AD-A264 091



(2)

Symposium on Applied Mathematical Programming and Modeling

APMOD93



Volume of Extended Abstracts

DTIC
ELECTE
APR 21 1993
S E D

93-08503



Budapest, Hungary
January 6-8, 1993

STRIBUTION STATEMENT
Approved for public release

Symposium on Applied Mathematical Programming and Modeling

APMOD93

Accesion For
NTIS CRA&I
DTIC TAB
Unannounced
Justification
By
Distribution /
Availability Codes
Dist | Avail and / or Special
A-1

Volume of Extended Abstracts

Editor: Prof. István Maros
Computer and Automation Institute,
Hungarian Academy of Sciences

Budapest, Hungary
January 6-8, 1993

Symposium on Applied Mathematical Programming and Modeling - APMOD93
Budapest, Hungary, January 6-8, 1993

ORGANIZED BY

Computer and Automation Institute,
Hungarian Academy of Sciences

SPONSORED BY

IFORS, International Federation of Operations Research Societies
RUTCOR, Rutgers Center for Operations Research
Hungarian Operations Research Society
Hungarian National Research Foundation (OTKA) #2587
US Army USARDSG-UK, AMXSN-UK-RI/(P)(70-1s)

ISBN 963 311 337 7

Hozott anyagról sokszorosítva
9220734 **AKAPRINT** Nyomdaipari Kft. Budapest. F. v.: dr. Héczey Lászlóné

Preface

You are holding the volume of extended abstracts of the symposium on Applied Mathematical Programming and Modeling (APMOD93) in your hand. The institutional organizer of the event is the Computer and Automation Institute of the Hungarian Academy of Sciences, venue: Budapest, Hungary, date: January 6-8, 1993.

The purpose of APMOD93, as a successor of APMOD91, held at Brunel University, London, UK, 1991, was to provide a continuing forum for new achievements in computational mathematical programming and modeling and their applications in solving large and difficult real-life problems.

The organization of APMOD93 was supported by many enthusiastic individuals and bodies. The bulk of the work in preparing the scientific program was done by the members of the International Program Committee.

This volume contains the extended abstracts of the papers accepted for presentation and received by December 1, 1992. The papers appear by the first authors, in alphabetic order.

It is our belief that the present volume will contribute to the rapid exchange of scientific information in the field of applied mathematical programming and modeling.

Budapest, December 1, 1992

István Maros
Chairman
International Program Committee
APMOD93

International Program Committee

István Maros - Chairman
(RUTCOR, Rutgers University, New Brunswick, NJ, USA & Hungarian Academy of Sciences)

Johannes Bisschop
(Twente University, Enschede, NL)
Endre Boros
(RUTCOR, Rutgers University, New Brunswick, NJ, USA)
Tibor Csendes
(József Attila University, Szeged, H)
Fred Glover
(University of Colorado, Boulder, USA)
Manfred Grauer
(University of Siegen, D)
Harvey Greenberg
(University of Colorado, Denver, USA)
Peter Hammer
(RUTCOR, Rutgers University, New Brunswick, NJ, USA)
Freerk Lootsma
(University of Delft, NL)
Gautam Mitra
(Brunel University, London, GB)
András Prékopa
(RUTCOR, Rutgers University, New Brunswick, NJ, USA & Eötvös Loránd University, Budapest, H)
Dave Shanno
(RUTCOR, Rutgers University, New Brunswick, NJ, USA)
Uwe Suhl
(Free University of Berlin, D)
Stavros Zenios
(University of Pennsylvania, Philadelphia, USA)
Margit Ziermann
(Representative of IFORS, H)

Contents

Badics, T., Boros, E.
Implementing a Maximum Flow Algorithm: Experiments with Dynamic Trees 11

Bajalinov, E.B.
On Decomposition of Dual Variables in Linear Programming and its Economic Interpretation 16

Bajalinov, E.B., Pannell, D.J.
GULF: A General, User-friendly Linear and Linear-Fractional Programming Package 21

Baker, T.E.
Graph-Based Modeling with MIMI/G 24

Barle, J., Grad, J.
LPRINT: LP Software Based on the Interior Point Method 27

Bennaceur, H., Plateau, G.
Impact of Quantitative Methods on Logical Inference Problem 31

Berland, N.J., Wallace, S.W.
Improving Bounds in Stochastic Programs by Partitioning the Support 34

Bianco, L., Blazewicz, J., Dell'Olmo, P., Drozdowski, M.
Scheduling Multiprocessor Tasks on a Dynamic Configuration of Dedicated Processors 36

Bierwirth, C., Kopfer, H., Mattfeld, D.C., Utecht, T.
PARNET: Distributed Realization of Genetic Algorithms in a Workstation Cluster 41

Biro, M., Bor, A., Knuth, E., Remzso, T., Szillery A.
Spreadsheet-Based Model Building and Multiple Criteria Group Decision Support 49

Black, J.A., Seyed-Hosseini, S.M.
Traffic Models for the Economic Evaluation of Private Sector Toll Roads and Tunnels in Australia 55

Boden, H., Grauer, M.
OpTiX-II: A Decision Support System for the Solution of Nonlinear Optimization Problems on Parallel Computers 71

Boros, E., Recski, A., Wettl, F.
Linear Time Algorithms for VLSI Routing 78

Chakrapani, J., Skorin-Kapov, J.
A Constructive Method to Improve Lower Bounds for the Quadratic Assignment Problem 84

Chinneck, J.W.
Finding Minimal Infeasible Sets of Constraints in Infeasible Mathematical Programs 92

Crama, Y., Oosten, M.
The Polytope of Block Diagonal Matrices 94

Csaki, P., Csiszar, L., Folsz, F., Keller, K., Meszaros, C., Rapcsak, T., Turchanyi, P.
A Flexible Framework for Group Decision Support: WINGDSS V3.0 102

Csendes, T., Zabinsky, Z.B., Kristinsdottir, B.P.
Global Optimal Solutions with Tolerances and Practical Composite Laminate Design 110

Darby-Dowman, K., Rangel do Socorro, M.
Preprocessing and Cutting Planes for a Class of Production Planning Problems 113

Darby-Dowman, K., Lucas C., Mitra, G., Smith, J.
Modelling Strategies in Integer Programming Applied to Scheduling a Fleet of Ships 121

Darby-Dowman, K., Kristjansson, B., Lucas C., Mitra, G., Moody, S.A.
Representing Procedural Knowledge within Mathematical Programming Modelling System (MPL) 128

Dell'Amico, M., Martello, S., Vigo, D.
Lower and Upper Bounds for the Single Machine Scheduling Problem with Eairliness and Flow Time Penalties 134

Dorndorf, U., Pesch, E.
Combining Genetic and Local Search for Solving the Job Shop Scheduling Problem 142

Drud, A.S.
Finding an Initial Feasible Point in the Large Scale GRG Code CONOPT 150

Eschenauer, H.A., Weinert, M.
Shape Optimization of Complex Shell Structures in a Parallel Computing Environment 158

Ferreira, C.E., Grotschel, M., Martin, A., Weismantel, R.
On Multidimensional Partitioning Problems: Facial Structure and Applications 166

Frandsen, P.E., Sorensen, S.B.
Contoured Beam Reflector Array Antenna Optimization with Dual Parameters Types 170

Freville, A., Hanafi, S.
A Lagrangean Relaxation for Optimizing a Discrete Production System in Manufacturing 178

Freville, A., Guignard, M.
Relaxations for Minimax Problems 185

Friedler, F., Fan, L.T.
Combinatorial Acceleration of the Branch and Bound Search for Process Network Synthesis 192

Fulop, J.
On a Special Class of Linear Programs with an Addititonal Reverse Convex Constraint 201

Gaivoronski, A.A., Messina, E., Sciomachen, A.
A Statistical Generalized Programming Algorithm for Stochastic Optimization Problems 209

Gassmann, H.I.
A Fast-Start Algorithm for Multistage Stochastic Programs 216
Giannessi, F.
Some Connections Between Integer Programming and Continuous Optimization 220
Glover, F., Babayev, D.
New Results for Aggregating Integer-Valued Equations 224
Glover, F., Skorin-Kapov, J.
Heuristic Advances in Optimization Integrating Tabu Search, Ejection Chains and Neural Networks 234
Greenberg, H.J.
An Overview of the Development of an Intelligent Mathematical Programming System 244
Greenberg, H.J., Murphy, F.H.
Equivalence of Mathematical Programs 246
Gurvich, V.A.
Extremal Integer Sequences with Forbidden Sums 247
Hajian, M., Levkovitz, R., Mitra, G.
A Branch and Bound Algorithm for Discrete Programming Using the Interior Point Method and the Simplex Method 257
Hertog den, D., Kaliski, J., Roos, C., Terlaky, T.
A Logarithmic Barrier Cutting Plane Method for Convex Programming 266
Hertog den, D., Jarre, F., Roos, C., Terlaky, T.
A Unifying Investigation of Interior-Point Methods for Convex Programming 274
Hurlimann, T.
IP, MIP and Logical Modeling Using LPL 282
Jansen, B., Roos, C., Terlaky, T.
On Vaidya's Volumetric Center Method for Convex Programming 289
Jarre, F.
An Efficient Line-search for Logarithmic Barrier Functions 296
Jonasson, K., Madsen, K.
A Projected Conjugate Gradient Method for Sparse Minimax Problems 304
Kall, P., Mayer, J.
Model Management for Stochastic Linear Programming 312
Keyser de, W.
Two Algorithms for Solving Linear Programs with Logical Constraints 318
Kort, P.M., Tapiero, C.S.
The Dynamic Theory of the Investing Polluting Firm and Pollution Insurance 324
Kovacs, L.B.
A Comparative Study of Modelling and Problem-Solving by Mathematical Programming and Logic Programming 329

Kristjansson, B., Lucas, C., Mitra, M., Moody, S.
Modelling Tools for Reformulating Logical Forms into Zero-One Mixed Integer Programs 337

Lasciak, A.
Madrid and Ramsay Models for Slovak Economy 345

Levkovitz, R., Mitra, G., Tamiz, M.
Experimental Investigations in Combining IPM and Simplex Based LP Solvers 353

Locatelli, M. Schoen, F.
An Adaptive Stochastic Global Optimization Algorithm for One-Dimensional Functions 374

Lootsma, F.A.
Scale Sensitivity and Rank Preservation in a Multiplicative AHP and SMART 382

Madsen, O.
Lagrangean Relaxation and Optimal Solutions to Time Window Constrained Vehicle Routing Problems 389

Mans, B., Mautor, T., Roucairol, C.
Recent Exact and Approximate Algorithms for the Quadratic Assignment Problem 395

Maros, I., Meszaros, C.
A Numerically Exact Implementation of the Simplex Method 403

Martello, S., Toth, P.
The Bottleneck Generalized Assignment Problem 410

Melamed, I.I.
Neural Network Algorithms for Combinatorial Optimization Problems: Theory and Experience 417

Meressoo, T., Vaarmann, O.
Numerical Solution of Certain Decomposition - Coordination Problems in Nonlinear Programming 422

Minoux, M.
Some Large Scale LP Relaxations for the Graph Partitioning Problem and Their Optimal Solutions 430

Neck, R.
Optimal Budgetary Policies under Uncertainty: A Stochastic Control Approach 434

Nygard, K.E., Ficek, R.K.
Genetic Search in Multi-Depot Routing 436

Padberg, M., Alevras, D.
Order Preserving Assignments 438

Pinar, M.C., Zenios, S.A.
Nonlinear Min-max Optimization via Smooth Penalty Functions 449

Pinter, J., Meeuwig Dirk J., Meeuwig Jay W., Fels, M., Lycon, D.S.
The ESIS Project: An Intelligent Decision Support System for Assisting Industrial Waste Management 455

Ponnambalam, K., Vanelli, A.
Forcing a Vertex Optimal Solution in Interior Point Method Using an Auxiliary Function 463
Prekopa, A. Li, W.
Solution of and Bounding in a Linearly Constrained Optimization Problem with Convex Polyhedral Objective Function 471
Proll, L..G., Salhi, A., Insua, D.R.
A Parallel Implementation of a Framework for Sensitivity Analysis in MCDM 475
Proll, L.G., Salhi, A., Insua, D.R., Jimenez, J.I.M.
A Comparison of Two Stochastic Algorithms for Global Optimisation 483
Queyranne, M., Spieksma, F., Tardella, F.
A General Class of Greedily Solvable Linear Programs 489
Rendl, F.
Eigenvalues and Graph Partitioning Bounds 499
Rote, G., Vogel, A.
New Heuristics for Decomposing Traffic Matrices in TDMA Satellite Communication 502
Rudolph, G.
Parallel Simulated Annealing and its Relation to Evolutionary Algorithms 508
Salhi, A., Lindfield, G.R., Proll, L.G.
The Role of Duality in the Efficient Use of a Variant of Karmarkar's Algorithm 516
Serin, Y., Avsar, Z.M.
Markov Decision Processes with Restricted Observations: Finite Horizon Model 521
Skorin-Kapov, D.
On the Core of the Minimum Cost Steiner Tree Games 529
Smith, J.M.G., Chikhale N.
Buffer Allocation for a Class of Nonlinear Stochastic Knapsack Problems 537
Sonnevend, G.
Construction and Implementation of a Central Path Following Algorithm for Semifinite Convex Programs in Matlab 541
Strayer, H.J., Colbourn, C.J.
Bounding Network Reliability Via Surface Duality 549
Sural, H., Koksalan, M., Kirca, O.
A Location-Distribution Model for a Beer Brewer 559

T. de Araujo, M.M.
Discrete Dynamic Programming in Environmental Optimisation 562
Tamiz, M., Jones, D.F., El-Darzi, E.
A Review of Goal Programming and its Applications 573
Terzi, A.T., Erkip, N.
Multi-Stage Economic Lot Scheduling Problem 580
Vial, J.P.
Computational Experience with a Primal-Dual Interior Point Method for Smooth Convex Programming 583
Vizvari, B., Demir, R.
It Is Difficult to Find a Difficult Problem for Scheduling of Identical Parallel Machines 587
Voss, S.
Concepts for Parallel Tabu Search 595
Willers, P., Proll, L., Wren, A.
A Dual Strategy for Solving the Linear Programming Relaxation of a Driver Scheduling System 605
Williams H.P.
Deriving the Dual of an Integer Programme: Its Interpretations and Uses 611
Yarrow, L.-A.
Obtaining Minimum-Correlation Latin Hypercube Sampling Plans Using Discrete Optimization Techniques 613

Late Papers 621

Implementing a Maximum Flow Algorithm: Experiments with Dynamic Trees (Extended Abstract)

T. Badics and E. Boros *

November 19, 1992

1 Introduction

In this paper we report on an implementation of a maximum flow algorithm by Cheriyan and Hagerup [1]. Our aim was to test the behavior of this algorithm in practice, concerning it's good theoretical worst case bound. We were particularly interested in the effect of using theoretically well behaving data structures such as dynamic trees [11], and Fibonacci heaps [13]. We also made comparisons to two preflow-push based algorithm by Goldberg and Tarjan [10], and to an implementation of pushes along several edges without using dynamic trees.

2 Basic notions and the PLED algorithm

We assume that the reader is familiar with the generic maximum flow algorithm in [10] and refer to [10] for definitions of the terms *network*, *source* s, *sink* t, *edge capacity* $c(v,w)$, *flow*, *maximum flow*, *preflow* f, *flow excess* $e(v)$ *of a vertex* v, *residual graph*, *residual capacity* $rescap(v,w)$ *of the edge* (v,w), *valid labeling* d, *active vertex*, *push*, *saturating push*, and *nonsaturating push*. Let $G=(V,E)$ denote the digraph (assumed symmetric) corresponding to the network. Let $N=|V|$, $M=|E|$.

Our implementation is based on the algorithm developed by Cheriyan and Hagerup (see [1]). Following [1] we shall refer to this algorithm as PLED (shorthand for Prudent Linking and Excess Diminishing). This algorithm is an instance (with one minor exception) of the generic preflow algorithm by Goldberg and Tarjan [10]. PLED also uses an idea introduced by Ahuja and Orlin [4]

*The second author is supported in part by the Office of Naval Research (Grants N00014-92-J1375 and N00014-92-J4083).

of scaling the volume of the pushes. The scaling factor plays here, however, a slightly different role: the limits imposed on the volume of a push are not the same as in [4]. A third idea in PLED is randomization: after each relabeling of a vertex v, the edgelist of v is permuted randomly. This random permutation ensures a better theoretical running time. Noga Alon showed in [3] that this randomization can be replaced by a deterministic procedure. The worst case running time of PLED, using the randomized procedure is $O(NM+N^2(logN)^2)$ with high probability (see [2]). Using Alon's derandomization, the deterministic worst case bound improves to $O(NM + N^{8/3}(logN))$. The worst case bound without randomization is $O(NMlog(N))$.

Three main data structures are essential for PLED.

- **Ordinary heap** that contains vertices which have big excesses and which are ordered by their distance labels. This structure supports the easy selection of a vertex for a push. (Select a vertex with the minimal distance label among the vertices having large enough excesses).
- **Fibonacci heap**(see [13]) contains the rest of vertices, ordered by their (small) excesses. This supports constant (amortized) time decrease key operation and fast update of the scaling factor.
- **Dynamic trees structure** (see [11]) to maintain a spanning forest F of G containing a subset of the current edges, where the value associated with an edge in F is its residual capacity. This structure is able to send flow value along a path of length L in (amortized) time $O(logL)$

3 Implementation

Since the algorithm requires the above data structures to achieve the theoretically best performance, we decided to implement all of them.

Beside the above data structures, we implemented a routine for randomly permuting the edge lists of the vertices. Although the deterministic permutation of Alon derandomizes the algorithm, the overhead of such a permutation generating procedure is so large that we did not expect much improvement by implementing such a deterministic procedure. Moreover since our instances were mostly randomly generated, after some preliminary experiments we used the PLED algorithm without random permutations.

For comparison reasons we coded Goldberg's simple preflow-push algorithm using simple FIFO queue for selecting the active vertices, and the dynamic trees version of this algorithm described in the same paper ([10]). Later we refer to these codes as GOLD and GOLDYN respectively. In the code GOLDYN we

did not control the size of the dynamic trees. Thus we got a code which has theoretical worst case bound $O(NMlogN)$ instead of $O(NMlog(N^2/M)$. The reason for this choice was mostly lack of time and the expected overhead of such a control mechanism. Besides, in the range of examples we tested the codes, the benefit from such size control, even without the overhead, probably is not significant.

For testing the performance of the dynamic trees data structure, we implemented its operations (*find-min, add-value, find-root, link, cut, etc.*) with storing the trees explicitly and executing these operations in the obvious (linear time) way. Hence we avoided the overhead of handling Splay trees and complicated updates. Later in this paper the codes using these "non-dynamic tree" operations are called NPLED and NGOLDYN. Note that the number of elementary operations for PLED and NPLED (or GOLDYN and NGOLDYN) are the same on the same instance, only the way of handling the tree operations are different. Therefore the difference in running time shows exactly the impact of the dynamic trees structure.

In our implementations of all the codes we employed an idea, mentioned in ([10]), the so called "global-" or "big-relabeling". Our early experiments showed clearly, that in PLED just like in GOLD or GOLDYN, the running times of the variant which uses global-relabeling were much smaller (orders of magnitude) than the one which does not use it. Therefore we built in some heuristic parameters controlling the calling frequency of big-relabeling and affecting thus the running time.

A "big-relabeling" step consist of two breadth-first-searches, one starting from the sink and working on the sink side of the residual graph, and another one for the source side, starting from the source. In these breadth-first-searches the shortest distance is calculated from each vertex to the sink on the sink side, or to the source on the source side, respectively. Unfortunately breadth-first-search is a relatively expensive operation (it takes O(M) steps), so the calling frequency of big-relabel is very important and can be a subject of later studies.

We implemented another mechanism to achieve better running times in all three codes. Namely at initialization the algorithm calculates an upper bound U on the maximum flow value by taking the minimum of capacities of some cuts. Then it creates a new source by adding an artificial vertex S and a new arc (S, s) with capacity U to the network, where s was the old source. The new problem is obviously equivalent with the old one, and the extra cost of its implementation is negligible. The advantage of doing this is that we do not let the algorithm push too much excess into the network, reducing in this way the runtime of the second phase. We have found instances showing that without this procedure the running time was significantly bigger due to the long second phase.

4 Experimental results

For the experiments, we used the DIMACS suggested problems, and the generators GENRMF, WASHINGTON, and AC-MAX [6,5,7]. (See the DIMACS document "The Core Experiments"). The families of networks we report on include the ones suggested by "The Benchmark Experiments", and two classes of problems made intentionally very difficult for Goldberg's preflow-push algorithm.

All the experiments were carried out on a Sun Sparc 1+ Workstation under UNIX operations system.

5 Conclusions

Summarizing our work, we can conclude that although the PLED algorithm has a very good theoretical worst case bound, in practice Goldberg's simple preflow-push algorithm outperforms it on most of the examples of this study.

Our study shows that the structure of the networks is the most important factor in ranking the algorithms. One such parameter to be considered, reflecting the structure of the network, could be the relative distance between the source and the sink.

In this study we were particularly interested in the effectiveness of dynamic trees. Our experiments show clearly that there are families of problems for which dynamic trees improved the performance of our code at a small cost. To determine the properties of network classes on which the algorithms GOLDYN or PLED are the best would be an interesting topic of later works.

Let us remark finally that Fibonacci heaps did not help much in these examples. They did not improve neither the running time nor the number of selection steps of PLED.

References

[1] J. Cheriyan and T. Hagerup: "A randomized maximum-flow algorithm". in Proc.IEEE FOCS (1989), 118-123.

[2] J. Cheriyan and T. Hagerup: "A randomized maximum-flow algorithm". Tech. Report 988, O.R.I.E, Cornell Univ., Ithaca, NY, Oct. 1991.

[3] N. Alon: "Generating pseudo-random permutations and maximum flow algorithms" *Information Processing Letters* 35 (1990), 201-204. North-Holland

[4] R. K. Ahuja and J. B. Orlin: "A Fast and Simple Algorithm for the Maximum Flow Problem". *Operations Research* 37 (1989), 748-759.

[5] R. Anderson, et al.: Program washington.c for creating maximum flow problem instances. Available through Dimacs.

[6] T. Badics: Program genrmf.c for creating maximum flow problem instances. Available through Dimacs.

[7] G. Waissi: Program ac-max.c for creating maximum flow problem instances. Available through Dimacs.

[8] U. Derigs and W. Meier: "Implementing Goldberg's max-flow-algorithm — A computational investigation". *ZOR - Methods and Models of Operations Research* **33** (1989), 383-403.

[9] A. V. Goldberg: "A New Max-Flow Algorithm". Tech. Rep. MIT/LCS/TM-291, Laboratory for Computer Science, Massachusetts Institute of Technology, Cambridge, MA, 1985.

[10] A. V. Goldberg and R. E. Tarjan: "A New Approach to the Maximum-Flow Problem". J. ACM 35 (1988), 921-940.

[11] D. D. Sleator and R. E. Tarjan: "A Data Structure for Dynamic Trees". *Journal of Computer and System Sciences* **26** (1983), 362-391.

[12] R. E. Tarjan: *Data Structures and Network Algorithms.* SIAM publications, Philadelphia, PA. 1983.

[13] M. L. Fredman and R. E. Tarjan: "Fibonacci Heaps and Their Uses in Improved Network Optimization Algorithms". *Journal of ACM* **34** (1987), 596-615.

[14] Gy. Novotny: "Comparison of Different Data Structures". *Thesis.* Eötvös Loránd University, Budapest, 1989.

On decomposition of dual variables in linear programming and its economic interpretation

Erik B. Bajalinov

Department of Mathematics, University of L.Kossuth, 4010 Debrecen, Pf.12, Hungary

Consider the following linear programming (LP) and linear fractional programming (LFP) problems

$$\max_{x \in S} P(x), \tag{1}$$

$$\max_{x \in S} D(x), \tag{2}$$

$$\max_{x \in S} Q(x), \tag{3}$$

where $Q(x) = P(x)/D(x)$, $P(x) = \sum_{j=1}^{n} p_j x_j + p_0$, $D(x) = \sum_{j=1}^{n} d_j x_j + d_0 > 0$ for all $x \in S = \{x \in R^n : Ax \leq b,\ x \geq 0\}$, A is $m \times n$ matrix, i.e. $A = \|a_{ij}\|_{m \times n}$, $x = (x_1, x_2, \cdots, x_n)^T$, $b = (b_1, b_2, \cdots, b_m)^T$; a_{ij}, b_i, p_j, d_j are scalar constants and T denotes the transpose of a vector. Assume that the feasible set S is non-empty and $P(x)$, $D(x)$ and $Q(x)$ are not constant on S.

Let the basis feasible solution which maximizes the objective function $D(x)$ be vector $x^* = (x_1^*, x_2^*, \cdots, x_m^*, 0, 0, \cdots, 0)^T$. Our aim now is to show that for any optimal solution of problem (2) we can find such vector $p = (p_0, p_1, \cdots, p_n)$ that x^* is optimal solution of LP problem (1) and LFP problem (3). Further, let $B = (A_1, A_2, \cdots, A_m)$ be the optimal basis associated with the positive variables, where $A_j = (a_{1j}, a_{2j}, \cdots, a_{mj})^T$ is j-th column vector of matrix A. Because the basis vectors are linearly independent we have

$$A_j = \sum_{i=1}^{m} A_i x_{ij}, \quad j = 1, 2, \ldots, n,$$

and we use these coefficients x_{ij} to define the following

$$\left.\begin{aligned} \Delta_j' &= \textstyle\sum_{i=1}^{m} p_i x_{ij} - p_j \\ \Delta_j'' &= \textstyle\sum_{i=1}^{m} d_i x_{ij} - d_j \\ \Delta_j(x^*) &= D(x^*)\Delta_j' - P(x^*)\Delta_j'' \end{aligned}\right\} \quad j = 1, 2, \ldots, n. \tag{4}$$

Further, the values $\Delta_j(x^*)$ can also be put in the form

$$\Delta_j(x^*) = \sum_{i=1}^{m} p_i R_{ij} - p_j D(x^*) - p_0 \Delta_j'', \quad j = 1, 2, \ldots, n,$$

where $R_{ij} = D(x^*)x_{ij} - \Delta_j'' x_i^*, \ i = 1, 2, \cdots, m, \ j = 1, 2, \cdots, n.$

Because the vector x^* is an optimal solution of (2) we have, [1].

$$\Delta_j'' \begin{cases} = 0, & j = 1, 2, \ldots, m, \\ \geq 0, & j = m+1, m+2, \ldots, n, \end{cases} \tag{5}$$

As in [1] and [2] the basis of LP problem (1) and LFP problem (3) is optimal in original form if $\Delta_j' \geq 0$ for all j and $\Delta_j(x^*) \geq 0$ for all j respectively but we require only to consider $j = m+1, m+2, \cdots, n$ because

$$\Delta_j' = \Delta_j(x^*) = 0 \quad j = 1, 2, \cdots, m.$$

The corectness of the following assertion is obvious.

Theorem 1 *If vector $p = (p_0, p_1, \cdots, p_n)$ satisfies the conditions*

$$\left.\begin{array}{l} \sum_{i=1}^{m} p_i x_{ij} - p_j \geq 0, \\ \sum_{i=1}^{m} p_i R_{ij} - p_j D(x^*) - p_0 \Delta_j'' \geq 0, \end{array}\right\} \quad j = m+1, m+2 \ldots, n, \tag{6}$$

then x^* is an optimal solution of LP problem (1) and LFP problem (3).

We denote the set of vectors p which satisfy the inequalities (6) by H. It is obvious that $H \neq \emptyset$. Indeed, if $p = \lambda d$ where $\lambda \geq 0$ and $d = (d_0, d_1, \cdots, d_n)$, then using (4) and (5) we get

$$\left.\begin{array}{l} \Delta_j' = \lambda \Delta_j'' \geq 0 \\ \Delta_j(x^*) = D(x^*)(\lambda \Delta_j'') - (\lambda D(x^*))\Delta_j'' = 0 \end{array}\right\} \quad j = m+1, m+2 \ldots, n.$$

It means that set H contains at least the ray λd where $\lambda \geq 0$.

Let us now consider the dual problems corresponding to primal problems (1), (2) and (3) respectively, [1] and [3],

$$\left.\begin{array}{lll} \textit{Minimize} & \varphi(u) = \sum_{i=1}^{m} b_i u_i + p_0 & \\ \textit{subject to} & \sum_{i=1}^{m} a_{ij} u_i \geq p_j, & j = 1, 2, \cdots, n \\ & u_i \geq 0, & i = 1, 2, \cdots, m \end{array}\right\} \tag{7}$$

$$\left.\begin{array}{lll} \textit{Minimize} & \phi(v) = \sum_{i=1}^{m} b_i v_i + d_0 & \\ \textit{subject to} & \sum_{i=1}^{m} a_{ij} v_i \geq d_j, & j = 1, 2, \cdots, n \\ & v_i \geq 0, & i = 1, 2, \cdots, m \end{array}\right\} \tag{8}$$

$$\left.\begin{array}{lll} \textit{Minimize} & \psi(y) = y_0 & \\ \textit{subject to} & d_0 y_0 - \sum_{i=1}^{m} b_i y_i \geq p_0, & \\ & d_j y_0 + \sum_{i=1}^{m} a_{ij} y_i \geq p_j, & j = 1, 2, \cdots, n \\ & y_i \geq 0, & i = 1, 2, \cdots, m \end{array}\right\} \tag{9}$$

The next theorem indicates an important relationship between the optimal solutions of these problems.

Theorem 2 *If LP problems (1), (2) and LFP problem (3) have at least one common non-degenerate optimal solution x^*, then the following decomposition takes place*

$$u_i^* = y_i^* + Q(x^*) v_i^*, \quad i = 1, 2, \ldots, m, \tag{10}$$

where $u^* = (u_1^*, u_2^*, \ldots, u_m^*)$, $v^* = (v_1^*, v_2^*, \ldots, v_m^*)$, $y^* = (y_1^*, y_2^*, \ldots, y_m^*)$ are optimal solutions of dual problems (7), (8) and (9) respectively.

Proof: Suppose that vector x^* is a common non-degenerate optimal solution of (1), (2) and (3). Let us replace the k-th element b_k of vector b by $b_k + \varepsilon$. Here and in what follows this replacement is claimed to effect no change in the basis of the optimal solution. In accordance with LP theory [1], for the new optimal solution $x' = (x_1', x_2', \cdots, x_m', 0, 0, \cdots, 0)^T$ we have

$$P(x') = P(x^*) + \varepsilon u_k^*, \tag{11}$$

$$D(x') = D(x^*) + \varepsilon v_k^*. \tag{12}$$

Analogously, in accordance with [4] for LFP problem (3) we get

$$Q(x') = Q(x^*) + \frac{\varepsilon y_k^*}{D(x')}.$$

This equation can be written as

$$P(x') = Q(x^*)D(x') + \varepsilon y_k^*.$$

A comparison of the latter with (11) makes us infer that

$$P(x^*) + \varepsilon u_k^* = Q(x^*)D(x') + \varepsilon y_k^*.$$

Making use of equation (12) in the latter we find

$$\varepsilon u_k^* = \varepsilon y_k^* + Q(x^*)\varepsilon v_k^*.$$

It means that the decomposition (10) is correct.

Let us now focus on the economic interpretation of the results described above. Let a certain company manufacture n differend kinds of a certain scarce product. Further, let p_j be the profit gained by the company from a unit of the j-th kind of the product, p_0 be some constant profit gained whose magnitude is independent of the output volume, b_i be the volume of some resource i available to the company and a_{ij} be the expenditure quota of the i-th resource for manufacturing a unit of j-th kind of the product. Denote the unknown output volume of some j-th kind of the product by x_j. If $D(x)$ is a total output of the product, then problem (2) corresponds to the economic interests of the consumers. If the company's aim is maximization of its profit $P(x)$ and/or production efficiency $Q(x)$ calculated as a profit gained from a unit of output, then problems (1) and (3) correspond to the company's economic interests. Suppose that vector x^* maximizes an output function $D(x)$ on the feasible set S, i.e. x^* is the best output plan from the customers' point of view. If the profit vector p satisfies the conditions (6), then vector x^* maximizes the company's profit $P(x)$ as well as production efficiency $Q(x)$. It means that to maximize its profit and/or production efficiency the company ought to organize its manufacturing in accordance with an output plan x^* which conforms to the economic interests of the consumers in the best way. In this case we will say that the economic interests of the company conform to the economic interests of the consumers.

Further, let the economic interests of the company conform to those of the consumers and x^* be the optimal solution of (1), (2) and (3). In accordance with Theorem 2 in

this case decomposition (10) takes place. It is obvious that (10) may be interpreted in the following way: if the volume of resource i increases by one unit, the profit of the company rises by u_i^* units. Furthermore, y_i^* units of them are created by more intensive production, whereas $Q(x^*)v_i^*$ units by more extensive production, where v_i^* is the output increase.

This decomposition may prove to be useful if scarce resources are distributed among producers in a centralized way. Indeed, let us suppose that the company has made a request to be allocated certain extra units of the i-th resource. From the customers point of view it would be reasonable to satisfy the request if and only if $v_i^* > 0$ because it is the very case when the use of an additional volume of the i-th resource brings about an extra output of the scarce product.

Another way of using (10) is to use $Q(x^*)v_i^*$ as extra charge for an extra unit of the i-th resource. Indeed, in this case if the use of an extra unit of the i-th resource does not lead to an increase in efficiency and $y_i^* = 0$ then the extra profit of the company is equal to zero, too. It means that these extra charges will create an interest in increasing the use primarily of a resource, whose index i_0 is defined from the equation

$$i_0 = ind \max_{1 \le i \le m} y_i^*$$

since in this case the extra profit is the largest. So if these extra charges have been introduced into practice they will be favourable for the intensification of production.

References

[1] Gass,S.I., *"Linear programming"*, McGrow-Hill, New-York, 1958.

[2] Martos,B., *"Hyperbolic programming"*, Publ. Math. Instt. Hungarian Academy of Sciences, 5, Sec.B, 1960.

[3] Bajalinov,E.B., *"On duality in linear fractional programming"*, Izvestiya Akad. Nauk Kirg. SSR, 2, 1981, 10-20 (Russian).

[4] Bajalinov,E.B., *"On the economic sence of dual variables of linear fractional programming"*, Ekonom.-matem. metody, 3, v.24, 1988, 558-561 (Russian).

GULF : A General, User-friendly Linear and linear-Fractional programming package

Erik B. BAJALINOV
University of L.Kossuth, Institute of Mathematics, 4010 Debrecen, Pf.12, Hungary

David J. PANNELL
The University of Western Australia, School of Agriculture, Nedlands W.A. 6009 Australia

GULF is a simple to use but powerful, menu driven linear-fractional programming (LFP) and linear programming (LP) package for IBM compatible MS-DOS micro-computers with minimum of 256K RAM and one floppy disk drive.

The LFP problem solvable by the program may be written as follows

$$Q(x) = \frac{P(x)}{D(x)} = \frac{\sum_{j=1}^{200} p_j x_j + p_0}{\sum_{j=1}^{200} d_j x_j + d_0} \rightarrow \max(\min), \tag{1}$$

subject to

$$\sum_{j=1}^{200} a_{ij} x_j \leq (\geq)(=) b_i, \quad i = 1, 2, \ldots, 150, \tag{2}$$

$$x_j \geq 0, \quad j = 1, 2, \ldots, 200, \tag{3}$$

where denominator $D(x) \neq 0$ for all $x \in S$. S is feasible set defined by the constraints (2) and (3).

GULF is centered around a spreadsheed styled editor which is used to enter or edit an existing problem. It operates similarly to an electronic spreadsheet program, such as Lotus 1-2-3, Quatro Pro or Excel. The commands are available through the slash (/) key. The user may use several commands, among which the Calculate command is to calculate the optimal solution. After calculating, GULF prints the optimal solution, retrieves the data file and returns to the editor.

The program is called by typing GULF [Return]. A spreadsheet represantation immediately appears, in the following format :

Gulf		Limit	Col 1	Col 2	Col 3	Col 4	Col5	Col 6
Obj.Numer	N							
Obj.Denom	N	1.00						
Row 1	L							
Row 2	L							
Row 3	L							
Row 4	L							
Row 5	L							
Row 6	L							
Row 7	L							
Row 8	L							
Row 9	L							
Row 10	L							
Row 11	L							
Row 12	L							
Row 13	L							
Row 14	L							
Row 15	L							
Row 16	L							
Row 17	L							
Row 18	L							
Row 19	L							

GULF v2.2 Row=-1 Col=0 Aim=MAX File=C:\DEFAULT.GLF

Type < / > for commands

The upper-left position is reserved for the problem name, which you may modify at will, as well as any of the spreadsheet position. The number 1.00 in the "Limit" column of the "Obj.Denom" row and 0.00 in the other columns of the row (zeros are blanked) are the default values of the objective function denominator's constant term and coefficients respectively. If you retain these default values, GULF solves a standart LP problem using the objective function coefficients in the "Obj.Numer" row. To solve a LFP problem, the "Limit" value of the "Obj.Denom" row must be changed to a value other than 1.00 and/or other coefficients of the row must be changed to values other than zero.

A spreadsheet styled data editor includes a full range of editing functions, is menu driven, has a help facility and gives informative error messages. There is a first level with the Calculate (solves LFP or LP problem) command, the Help (it leads the user to 4 help screens), the Alter command for modifying the default parameters, the File command for file operations, the Print command, the Quit command, and a [Tab] command (which can also be reached directly from the editor) which leads

to matrix operations (Delete row or column, Insert, Copy, etc.).

To solve a LP or LFP problem GULF uses well known simplex algorithm [1], [2]. The user has the choice between two solution methods : the simple *steepest ascent* and the *highest step* [3] pivot selection. The second method involves longer iterations, but may result in less steps. The package includes an ideal feature for those who are learning about simplex method : the facility to drop back into the editor to view the matrix after each iteration.

The optimal solution and/or problem matrix can be printed on the screen, on a printer or into a text file on disk. Output includes levels, slacks, shadow costs and prices and range analysis, each of which can optionally be suppressed. After printing output, GULF returns to the editor and you can continue making any changes to the matrix.

Standart MPS data format is used, so data can be exchanged with other LP packages on mainframe or micro. It is possible to write your own data entry program which interfaces directly with GULF's solving algorithm, bypassing the data editor.

With a 256K RAM memory, the maximum size is 120 nonnegative variables and 80 constraints. This maximum size increases to 200 nonnegative variables and 150 constraints if you have 384K or more RAM. There is no minimum disk size required as GULF takes up only about 100K of disk space. GULF is not copy protected.

References

[1] Gass,S.I., *"Linear programming"*, Third edition. McGrow-Hill, New-York, 1969.

[2] Martos,B., *"Hyperbolic programming"*, Publ. Math. Instt. Hungarian Academy of Sciences, 5, Sec.B, 1960.

[3] Heesterman,A.R.G., *"Matrices and simplex algorithms"*, D.Reidel Publishing Company, 1983.

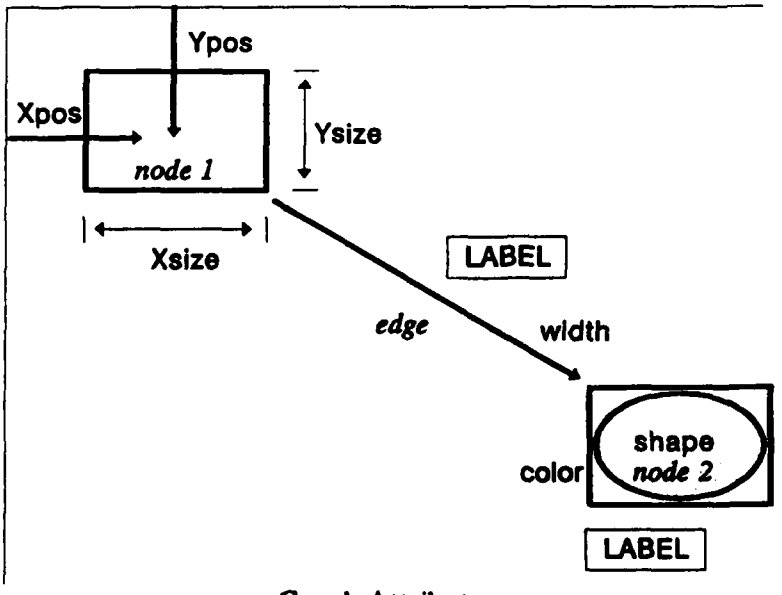
Graph-Based Modeling With MIMI/G

Thomas E. Baker
Chesapeake Decision Sciences, Inc.
200 South Street
New Providence, NJ 07974

We describe the results of a two-year research and development project in the area of graph-based modeling. The project was funded by a consortium of six industrial sponsors and was carried out by Chesapeake Decision Sciences, a small U.S. company that specializes in the development of state-of-the-art software in the area of planning and scheduling.

The graph-based modeling developments served as an extension of the existing MIMI (Manager for Interactive Modeling Interfaces) system which provides operations research, expert system, interactive graphics, and database capabilities for the solution of complex industrial planning and scheduling problems. Thus, the graph-based modeling features (MIMI/G) are intended to provide support for all aspects of the MIMI system including mathematical programming model management and solution analysis. As opposed to programming specialized windowed interfaces, we set out to provide a generic graph-based modeling language to enable end users to create new interfaces in a few seconds.

Graph-based modeling in MIMI/G uses a node/edge paradigm in which graph attributes are associated directly with structures and data in the MIMI database. The MIMI database consists of sets (ordered lists) and tables (defined on sets) and supports both hierarchical and relational data models. Graph nodes are generally associated with objects or entries in MIMI sets. Graph edges are usually associated with relations or entries in MIMI tables. Graph attributes (e.g., node position, node size, node color, edge width, edge color, etc.) are associated with values in MIMI tables.



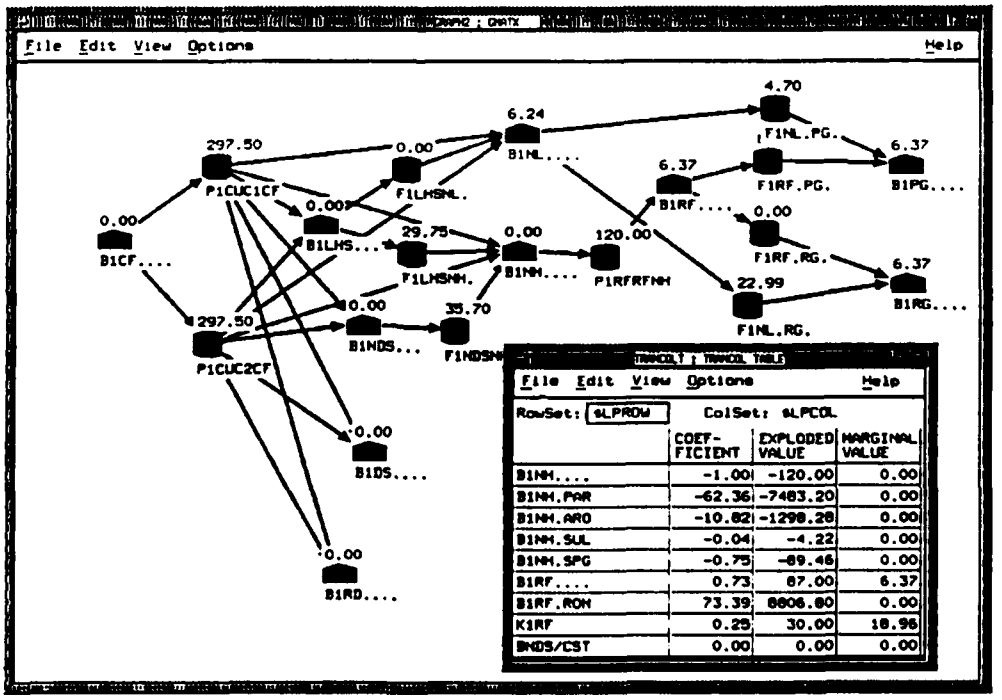
Graph Attributes

Relations defined on special MIMI sets (called graph sets) are used to specify the mapping of database elements into graph attributes. The GRAPH command, operating on graph sets, generates the graph in the X Window environment. Once generated, graphs represent a one-to-one relationship with the underlying data; a change to the graph changes the data and vice versa. Adding or deleting nodes in a graph add or delete the set entries in the MIMI database. Adding or deleting edges add or delete entries in MIMI tables. Thus, the graphs themselves become a natural user interface.

Each node and edge in a graph has a domain defined by the sets and set entries associated with the node or edge. If the nodes in a graph were generated as the entries in a particular set, then the domain of each node can be as simple as a tuplet listing the defining set and the entry associated with each node. However, the nodes in many graphs are defined on complex domains represented by several tuplets of defining sets and set entries.

In general, graphs are generated for a large variety of domains but displayed selectively only for a few domains. The manipulation of the domains that define which portions of the graph to display and which to hide is called graph navigation. Since the graphs in industrial applications are generally quite large (with hundreds or even thousands of nodes and edges), efficient graph navigation is key to the success of MIMI's graph-based modeling development efforts.

For example, we might choose to generate a graph of a large portion of an LP matrix with nodes defined as matrix columns from set MAC and matrix rows from set MAR. Edges would represent nonzero matrix elements from the sparse table MATX(MAR,MAC). Each matrix column also has a domain associated with the meaning of the column in physical terms–blending activity, time period 1, product PA, location BR, etc. These domains are also associated with the node representing the matrix column so that we can navigate the matrix graph by specifying a filter of partial domains for display.



Graph nodes are treated as objects in the object-oriented programming sense. When the user selects a node with the left or right mouse button, the graph tells the MIMI database the domain (of the node) the user has selected. MIMI macros or rules can be linked to nodes so that they will be run or fired upon mouse selection. Nodes can also be associated with additional data structures called frames which will pop up editable windows focused on the current domain of interest with selection with the right-mouse button.

The MIMI database supports inheritance and so does the frame feature associated with the right mouse button. Any text selected by the right mouse button is referenced through MIMI's database structures to present a window with the correct data (perhaps inherited) focused on the active domain of interest.

Node and edge shapes can be selected from a list of standard shapes or from external pixmaps supplied by the user. Thus, graphs can also be used to create icon-style interfaces for mouse selection. Pixmaps can also be imported as background (e.g. maps, plant layouts) for superimposing graph structures related to data.

Quite often, the x,y positions of graph nodes are related to data in the MIMI database; however, in some cases we would like the x,y positions of the nodes to be controlled by edge relationships. For these graphs, MIMI/G contains six layout routines which can be specified as part of the graph set definition.

Node/edge relationships in graphs often reveal structure at a glance. However, they also provide a natural interface for delving into the MIMI database along the lines indicated by the mouse selection of the user in an intuitive form.

The visualization of relations and data structures provides insight in many novel forms. Obviously, many physical problems have natural graphical interpretations, and we would expect that it would be easy to represent the data structures describing these problems in a graph-based modeling form. In all cases in which we found the specification of a graph set difficult for a natural problem, upon examination we found that the data structure representing the problem to be inefficient or unnatural–an observation which had escaped us prior to graphical visualization.

The graph-based modeling language in MIMI/G is quite simple to learn and to use. Many graphs can be specified with a few lines that relate graph attributes to database structures. MIMI/G facilitates the modeling process by allowing classes of graphs to be defined and by allowing other graphs to inherit their properties. Thus, neophyte users are able to specify new graphs on the basis of a collection of example graphs without completely understanding the process. Nevertheless, the ability to develop novel applications of graph-based modeling seems to be limited to a few–probably the same small subset of people who are good at modeling in general.

Since the MIMI/G development is new, our observation of users is limited. However, initial results indicate that there is a high degree of acceptance of the graph-based interfaces among model developers and end users alike.

LPINT: LP SOFTWARE BASED ON THE INTERIOR POINT METHOD

Janez Barle, Janez Grad*

Linearni ekonomski modeli d.o.o.
Kogojeva 1, 61117 Ljubljana, Slovenia

*Ekonomska fakulteta
Kardeljeva ploščad 17, 61109 Ljubljana, Slovenia

Karmarkar's algorithm (Karmarkar 1984) and other interior point methods are now regarded as a competitive methods for solving linear programming (LP) problems. It is therefore worth-while to undertake development of a professional LP software based on some particular interior point method. We describe design and implementation aspects of LPINT, an LP software package which is based on the primal-dual interior point algorithm. Its main characteristics can be stated as follows:

- high performance is assured by using state of the art algorithms (Lustig et al. 1991, Mehrotra 1991, Altman, Gondzio 1992) and recent results in sparse matrix research (George, Liu 1981, Duff et al. 1989).
- the overall system design is influenced by proven systems which are based on the simplex method (Suhl 1989). It is also very important to follow the methods and principles of contemporary software engineering. For example, it is desirable to attain a high degree of portability. We also emphasize the need for modularity, ease of use and other software qualities. Our goal was to obtain these qualities

without abandoning the usage of standard form PC user interface (pop-up and pull-down menus, windows etc.)

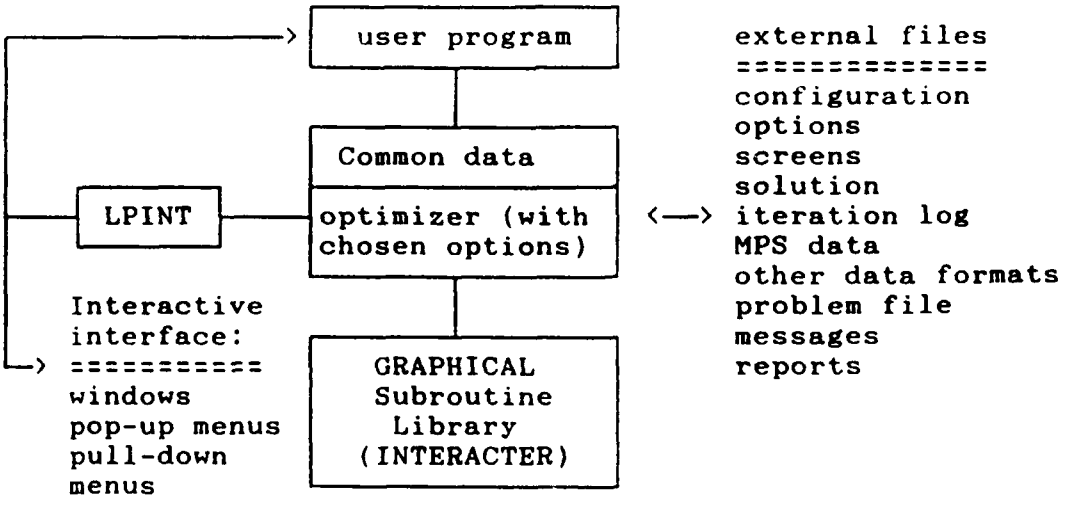
- two levels of use are provided: 1. interactive menu driven use,

  2. as a library of fortran subroutines which are driven by the user program

- LPINT was extensively tested using the so called NETLIB library of LP test problems (Gay 1985).

At each step of the primal-dual interior point it is necessary to solve the presumably sparse linear least square problem. It is generally accepted that, if problem dimension is not very big, the normal equations approach using sparse Cholesky factorization (George, Liu 1981) is an adequate method. The rows and columns of the normal equations matrix must be preordered in order to exploit sparsity. We have implemented the following methods for doing this: 1. Minimal degree algorithm (usually the most efficient method), 2. Nested dissection method (uses the same data structure as minimal degree algorithm, but it is less efficient in exploitation of sparsity when LP problems are considered), 3. Reverse Cuthill-McKee algorithm (a standard profile method which proved to be more efficient than minimum degree algorithm on some rare cases, but otherwise its performance is poor), 4. Modified Levy algorithm (an alternative profile method (Billionnet, Breteau 1989)). Among other algorithmic techniques implemented within LPINT we can mention splitting

dense columns (Gondzio 1991). In general, our goal was not to invent new algorithms but to enable making different comparisons as a starting point for further investigation of the algorithms and LP matrices. Perhaps the most distinguished components of LPINT are different tools for graphical display of LP constraint matrices and corresponding normal equation matrices. A visualization of these matrices could be particularly helpful when one must decide about efficient solution strategy and possible decomposition of the LP model. Our ultimate goal, which is not yet fully achieved is to create an open software environment for handling and analysing different sparse matrices (Alvarado 1990) which is to be specialized for LP matrices. We believe that all mentioned features make LPINT a valuable tool for postgraduate education and research in the field of LP.



LPINT system architecture

LPINT is currently implemented only on the PC, but the usage of portable graphical subroutine library (Interacter 1991) makes possible its porting to other software and hardware platforms.

References

Altman A., Gondzio J. (1992) An efficient implementation of a higher order primal-dual interior point method for large sparse linear programs. Technical Report ZTSW-1/A214/92, System Research Institute, Polish Academy of Sciences. Warszawa

Alvarado F.L. (1990) Manipulation and visualization of sparse matrices. ORSA Journal on Computing 2: 187-207

Billionnet, A.; Breteau, J.F. (1989) A Comparison of Three Algorithms for Reducing the Profile of a Sparse Matrix. Recherche Opérationnelle 23, 289-302

Duff I.S., Erisman A.M., Reid J.K. (1989) Direct Methods for Sparse Matrices. Clarendon Press. Oxford

Gay D.M. (1985) Electronic Mail Distribution of Linear Programming Test Problems. Mathematical Programming Society COAL Newsletter 13: 10-12

George J.A., Liu J.W. (1981) Computer Solution of Large Sparse Positive Definite Systems. Prentice Hall. Englewood Cliffs

Gondzio J. (1991) Splitting dense columns of constraint matrix in interior point methods for large scale linear programming. Technical Report No 105, LAMSADE, University of Paris Dauphine. Paris

INTERACTER: The Portable Fortran 77 User-interface Development System (1991). User Guide. Interactive Software Services Ltd. Stafford (United Kingdom)

Karmarkar N.K. (1984) A New Polynomial-Time Algorithm for Linear Programming. Combinatorica 4: 373-395

Lustig I.J., Marsten R.E., Shanno D.F. (1991) Computational Experience With a Primal-Dual Interior Point Method For Linear Programming. Linear Algebra and its Applications 152: 191-222

Mehrotra S. (1991) Higher order methods and their performance. Technical Report 90-16R1, Department of Industrial Engineering and Management Sciences, Northwestern University. Evanston

Suhl U.H. (1989) Design and Implementation of a Fast and Portable LP-Optimizer. Operations Research Proceedings 1988. Springer Verlag. Berlin Heideberg: 165-173

Symposium on
Applied Mathematical Programming and Modeling
APMOD'93

INVITED PAPER IN THE SESSION
INTERFACE BETWEEN AI AND INTEGER PROGRAMMING
organized by
Monique GUIGNARD

IMPACT OF QUANTITATIVE METHODS ON LOGICAL INFERENCE PROBLEM

Hachemi BENNACEUR and Gérard PLATEAU
Université Paris-Nord
LIPN, URA CNRS 1507
Institut Galilée, Avenue J. B. Clément
93430 Villetaneuse, France

In previous works we designed a general method for linear diophantine constraint satisfaction problem -denoted by FAST (Fast Algorithm for the constraint Satisfaction Testing)- which allows to prove the existence or not of a solution for a system of constraints over a finite domain.

Namely, the system has the following canonical type :

(S) $Ax \leq b$; $x \in D$, D discrete and finite

(all the components of the vector b and each coefficient of the matrix A are integer numbers).

This problem arises in several applications in computer science, namely in Artificial Intelligence area (such that : logical inference and SAT problems, regular problems (pigeon, queen, puzzle....., and for constraint logic programming), and in automatic vectorization of programs.

The main characteristics of our method is the solving of a sequence (very short in practice) of integer programming problems. Each generic problem of this sequence has an appropriate objective function and a constraint system size lower than the initial system one (very much lower in practice).

The algorithm starts from an initial vector x^0 in D. Let us denote

L the subset of the m constraints of (S) already satisfied (e.g. $A_i x^0 \leq b_i \;\; i \in L$)

and

G the subset of the other constraints (e.g. $A_i x^0 > b_i \;\; i \in G$).

From this starting point x^0, algorithm FAST generates a finite sequence of k integer vectors x^1, x^2, x^3,..., x^k in D, until either x^k satisfies the system of constraints (S), or the associated domain ($F(S)=\{x \in D \mid Ax \leq b\}$) is proved to be empty.

Namely, given an element x^h of the sequence which is not a solution of (S) (with $h \leq k-1$), by denoting again

L the subset of the m constraints of (S) already satisfied (e.g. $A_i x^h \leq b_i \;\; i \in L$)

and

G the subset of the other constraints (e. g. $A_i x^h > b_i \ i \in G$),
the next integer vector x^* (e.g. x^{h+1}) is obtained by solving (or partially solving) the following integer linear programming problem

(P)	min $f(x)$
	s.t. $A_i x \leq b_i \quad i \in L$
	$x \in D$

where f(x) is a positive linear combination of the $(A_i x)_{i \in G}$:

$$f(x) = \sum_{i \in G} \alpha_i A_i x \ , \ \alpha_i \geq 0 \quad \text{for all i in G.}$$

The number of iterations of our method is bounded by the number of contraints of the initial system, but at each iteration a NP-complete integer programming problem is solved exactly or approximately.

We propose to describe a specific version of algorithm FAST -denoted BFAST- devoted to the exact solution of linear boolean constraint satisfaction problems., e.g. with a system of this type

$$(BS) \qquad Ax \geq b;\ x \in \{0,1\}^n$$

As a matter of fact, it is well known that a propositional logic clause can be written as a 0-1 linear inequality in the following way :

the clause

$t_1 \vee \neg t_2 \vee \neg t_3 \vee t_4$

is equivalent to the diophantine constraint

$x_1+(1-x_2)+(1-x_3)+x_4 \geq 1$, with x_i in $\{0,1\}$, i=1,...,4.

This means that each x_j is a mathematical variable rather than a proposition and is interpreted as having the numerical value 1 when the proposition t_j is true and 0 when t_j is false. The numerical inequality asserts that at least one of the fourth literals is true.

Thus a set of clauses can be written as a system (BS) corresponding to a generalized covering problem : all the components of the right-hand side b of the constraints are integer numbers, and each coefficient of the left-hand side A of the constraints belongs to {-1, 0, 1}), each row of which corresponds to a clause.

Important typical applications are the inference problem in propositional logic, and deductive databases.

Although this constraint satisfaction problem associated with (BS) is NP-complete, it is possible to design efficient exact methods for several class of such instances. Recently Hooker has obtained good results using 0-1programming tools : his method consists in adding an objective function to the constraint system (BS) in order to solve an equivalent O-1 programming problem by a branch and cut method .

The associated computational experiments show clearly that his algorithm is largely as fast as the classical previous symbolics methods (set of support resolution, Davis-Putnam's procedure, ...) for logical inference problems.

Our BFAST algorithm solves a sequence of 0-1 programming problems (obtained by adding an objective function to subsystems of (S)). The solution of each generic 0-1 problem is obtained by a branch and bound method including heuristics, relaxations and reduction procedures.

The associated C code has been implemented on a SUN 3/160 computer with a lot of instances with a generalized covering type

$$Ax \geq b;\ x \in \{0,1\}^n$$

with $A \in \{0,1,-1\}^{mxn}$ and $b \in Z^m$

randomly generated with the Purdom and Brown model. Each clause is randomly and independently generated: each literal has the same probability; each clause includes distinct literals.

The computational experiments show the efficiency of our method BFAST.

Improving bounds in stochastic programs by partitioning the support

Nils Jacob Berland
Stein W. Wallace
Department of Economics
Norwegian Institute of Technology
University of Trondheim
N-7034 Trondheim
Norway

e-mail: njb@iok.unit.no sww@iok.unit.no

Abstract: The problem of bounding the expected value of the objective function in a stochastic program can be of interest in its own right (for example finding the expected project duration time in a stochastic PERT network) or it can be a part of a larger setting such as for example a two-stage stochastic program. We consider a general LP of the form: Find the expected value of Q, where Q is given by

$$Q = \sum_i \min\{qy \mid Wy = \omega_i,\ y \geq 0\}p_i$$

where we view ω_i as the i'th realization of a random variable $\tilde{\omega}$, with p_i being the probability that $\tilde{\omega} = \omega_i$. Finding the exact value of Q is hard expect for very small problems. However, for general LPs there exist different approaches for bounding Q, such as the Jensen lower bound and the Edmundson-Madansky upper bound. Whichever bound is used, one will often experience that the bounds are not tight enough according to some chosen rule. A natural possibility is then to partition the support of $\tilde{\omega}$ and then find conditional bounds on each *cell* of the partition.

In this paper we discuss different ways of partitioning the support. It is fairly obvious that one will always partition the cells(s) with the largest error (where error is measured as the difference between the upper and

lower bound multiplied by the probability associated with the cell). However, given a cell, one must decide how to do it. Due to the difficulty of finding conditional expectations over anything but rectangles we immediately decided to consider only partitions that affect one random variable at a time. Also, after some preliminary testing, we decided to split a cell in the middle, i.e. as close to the midpoint between the minimal and maximal value as possible. Of course, one could also have chosen the mean or median. Our computations indicate that that is less useful, but that the difference is not substantial.

However, our main issue is to understand better which dimension (random variable) to partition on. Our results indicate very clearly that picking the correct dimension is crucial. This is perhaps best understood if we for a moment assume that we introduce a random variable that does not show up anywhere in the LP. If we chose to split on this random variable, the bounds will remain unchanged, but we now have two cells, each as difficult as the first one. Hence, we must bring both cells down to an acceptable error and this is basically twice as hard as bringing the error associated with the original cell down. In other words, picking an incorrect random variable has doubled our workload. Of course, we never have such random variables in a problem. But we will often have random variables that are totally uniteresting (for example the duration of an activity in a PERT network which is such that irrespective of the value taken by this random variable the activity is *never* critical). The problem with an incorrect choice is that we never recover from it. With a totally useless partition, the remaining workload associated with the cell in question basically doubles, and that cannot be offset later on.

Given this important observation we discuss a number of basic approaches to the question of how to pick the right random variable. These basic approaches are combined with different shortcuts. For example, if we foresee that a given partition will cause one of the resulting new cells to immediately satisfy the error bounds, we chose this partition without further testing.

The talk will outline these approaches and present numerical results. We demonstrate that bad choices can increase the workload several orders of magnitude.

Scheduling multiprocessor tasks on a dynamic configuration of dedicated processors

L.Bianco * J.Błażewicz †, P.Dell'Olmo*, M.Drozdowski†

Extended abstract

One of the commonly imposed assumptions in the classical sheduling theory is that any task is processed by one processor at a time [10, 1]. With the development of technology, parallel systems and parallel algorithms this assumption is not so obvious. For example consider a fault tolerant system in which several processors test each other [13] or a testing system in which one processor stimulates the tested object and the other processor is analysing its output. Another range of applications appears in the field of new parallel algorithms and corresponding tasks systems.

In recent years several papers dealt with a problem in which a task requires more than one processor simultaneously. Two groups of models have been distinguished. In the first group of models it is assumed that any task can be executed on any set of processors under the condition that a fixed *number* of processors is assigned to the task [6, 11, 7, 9, 15]. There are three models in this group [16]: in the model called "$size_j$" a task requires a fixed number of processors simultaneously [6, 7]; in the model "$cube_j$" a task requires a number of processors which is a power of 2 (eg. either 1 or 2 or 4 etc. processors) [9, 15]; in the model "*any*" each task can be executed on any subset of the processors but the execution speed depends on the number of processors processing the task [11, 17].

In the second group of models it is assumed that the number of processors is not important, but the *set* of processors processing a task [14, 3, 5]. This

*Instituto di Analisi dei Sistemi ed Informatica del C.N.R., Rome,Italy.

†Instytut Informatyki Politechniki Poznańskiej, Poznań, Poland. The research was partially supported by the grant KBN-302279101 and by project CRIT.

problem is similar to classical scheduling with additional resources [8] and can be expressed in terms of weighted graph colouring [14]. There are two models of in this group. These are: model "fix_j" where a task can be executed by a fixed set of processors [14, 3, 5] and model "set_j" in which each task has a set of alternative sets of processors by which it can be processed.

In this paper we will concentrate on the model "set_j" which is a generalization of the model "fix_j". Before presenting results we will set up the problem more formally.

We are given set $\mathcal{T}$ of n tasks and set $\mathcal{P}$ of m dedicated processors. Each task T_j requires for its processing a set of processors D_l simultaneously from a set S_j of such sets (ie. $S_j = \bigcup_{l=1}^{|S_j|} D_l$). We will call these D_l sets *processing modes* or *processing configurations* of task T_j.

A processing time of a task may depend on the set of processors processing it. We assume that processing times of tasks are given in the matrix:

X=$\{t_j^{D_i} : t_j^{D_i} \in R^+$ is a processing time of task T_j in processing mode i requiring a set of processors D_i; if T_j cannot be scheduled in this mode then $t_j^{D_i} = +\infty$ $\}$

Tasks are independent. We will analyze preemptable and nonpreemptable task cases. In case of preemptable tasks any task can be at no cost interrupted and restarted later probably in different processing mode. In this case we also assume that processing percentages of tasks processed in various processing modes are "additive" or in other words can be accumulated. For example if some task has been processed 1 second in processing mode A while the total processing time for this task in this mode is 10 seconds, then the task is processed in 10%. If next, this task has been processed in additional 20% in some other processing mode then it is processed in 30%. After restarting in the processing mode A this task will occupy processors appropriate in this mode in 7 seconds. This approach is similar to the case of scheduling on unrelated machines or scheduling under resource requirements [4].

An optimality criterion is schedule length (C_{max}).

To denote analyzed problems we will use an extended version of the scheme proposed by Graham, Lawler, Lenstra and Rinnoy Kan [12] with later extensions [8, 16]. In this notation a scheduling problem is described by three fields. The first field describes processor system. In this work it will

be letter P optionally followed by positive integer which denotes the number of processors. If there is no constant after P then the number of processors is not fixed and is given in the current instance of the problem. The second field describes the task system. Word "*pmtn*" is used to denote that tasks are preemptable, if this word is absent tasks are nonpreemptable. Word "set_j" denotes simultaneous requirement of multiple processors by tasks. Moreover in general any task can be processed by more than one such a set of processors. The last field denotes the optimality criterion, it is C_{max}.

In the paper we will present a dynamic programming based procedure to solve optimally simple cases of the nonpreemptive version of the problem. This will result in pseudopolynomial algorithms. For a general case of the nonpreemptive scheduling a heuristic will be proposed and its worst case behavior will be analyzed. The preemptive version of the problem will be solved via linear programming. The organization of the paper is as follows. In section 2 the case of nonpreemptive scheduling is considered. In section 3 the preemptive version of the problem is solved.

Nonpreemptive Scheduling

In general the problem P| set_j | C_{max} is NP-hard. This can be easily shown by a reduction from the set partition problem to problem P2| set_j | C_{max}. For three processors and tasks requiring processors from only one set the problem is NP-hard in the strong sense [5]. Thus, it is unlikely to propose an algorithm solving these problems in polynomial time. Moreover, for more than two processors it is hard to expect pseudo-polynomial time algorithm.

In this section we will present pseudo-polynomial time algorithms for problems P2| set_j | C_{max} and a restricted version of the problem P3| set_j | C_{max}, respectively. Then a simple heuristic for the problem P| set_j | C_{max} with the worst case behavior bound equal to m, will be presented.

Preemptive Scheduling

In this section we will analyze the problem P| $set_j, pmtn$ | C_{max}. In general (when the number of processors is unbounded), the problem in question is NP-hard in the strong sense [2]. For a limited number of processors however, this problem can be solved in polynomial time using linear programming procedure.

References

[1] K.R.Baker, Introduction to Sequencing and Scheduling, John Willey & Sons, New York, 1974.

[2] L.Bianco, J.Błażewicz, P.Dell'Olmo, M.Drozdowski, Scheduling preemptive multiprocessor tasks on dedicated processors, paper in preparation.

[3] G.Bozoki, J.P.Richard, A branch-and-bound algorithm for continuous-process task shop scheduling problem, AIIE Trans. 2, 1970, 246-252

[4] J.Błażewicz, W.Cellary, R.Słowiński, J.Węglarz, Scheduling under Resource Constraints: Deterministic Models, J.Baltzer, Basel, 1986,

[5] J.Błażewicz, P.Dell'Olmo, M.Drozdowski, M.G.Speranza, Scheduling multiprocessor tasks on three dedicated processors, IPL 41, 1992, 275-280.

[6] J.Błażewicz, M.Drabowski, J.Węglarz, Scheduling multiprocesor tasks to minimize schedule length, IEEE Trans. Comput., C-35, 1986, 389-393.

[7] J.Błażewicz, M.Drozdowski, G.Schmidt, D.de Werra, Scheduling independent two processor tasks on a uniform duo-processor system, Discrete Applied Mathematics 28, 1990, 11-20.

[8] J.Błażewicz, J.K.Lenstra, A.H.G.Rinnoy Kan, Scheduling subject to resource constraints: classification and complexity, Discrete Applied Mathematics 5, 1983, 11-24.

[9] G.I.Chen, T.H.Lai, Preemptive scheduling of independent jobs on a hypercube, IPL 28, 1988, 201-206

[10] E.G.Coffman Jr. (ed.), Computer and Job-Shop Scheduling Theory, John Willey & Sons, New York, 1976.

[11] J.Du, J.Y-T.Leung, Complexity of scheduling parallel task systems, SIAM J.Discrete math. 2, 1989, 473-487

[12] R.I.Graham, E.L.Lawler, J.K.Lenstra, A.H.G.Rinnoy Kan, Optimization and approximation in deterministic sequencing and scheduling: a survey, Ann. Discrete Math. 5, 1979, 287-326

[13] H.Krawczyk, M.Kubale, An approximation algorithm for diagnostic test scheduling in multicomputer systems, IEEE Trans. Comput. C-34, 1985, 869-872

[14] M.Kubale, The complexity of scheduling independent two-processor tasks on dedicated processors, IPL 24, 1987, 141-147.

[15] J.Plehn, Preemptive scheduling of independent jobs with release times and dead-lines on a hypercube, IPL 34, 1990, pp. 161-166

[16] B.Veltmann, B.J.Lageweg, J.K.Lenstra, Multiprocessor scheduling with communication delays, Parallel Computing 16, 1990, pp. 173-182

[17] Q.Wang, K.H.Cheng, A heuristic of scheduling parallel tasks and its analysis, SIAM J. on Comput., 1992, vol 21, No.2, pp.281-294.

PARNET*:Distributed Realization of Genetic Algorithms in a Workstation Cluster

Christian Bierwirth, Herbert Kopfer, Dirk C. Mattfeld, Thomas Utecht

FB. Wirtschaftswissenschaften,
University of Bremen, FRG, Nov. 1992

Abstract

The population of parallel genetic algorithms (PGAs) can easily be split up to match the needs of a coarse grained parallelism. A cluster of interconnected workstations, seen as an MIMD-architecture, is the chosen hardware to express this kind of parallelism. A PGA implementation, as any other parallel algorithm, is bound to an execution environment, giving sustained support for its realization. Our execution environment, called PARNET, is constructed as a distributed server referred to as base layer. The next higher level of abstraction is provided by the object oriented interface layer, allowing to construct the PGA layer on top of both. We introduce the PARNET conception, leading from the single machine operating environment to the distributed realization of a PGA.

1 Introduction

Genetic algorithms belong to a class of optimization strategies, which can be implemented most efficiently on MIMD-architectures. A PGA population matches the needs of a coarse grained parallelism for efficient, almost asynchronous processing [1].

The realization in a cluster of interconnected workstations must be supported by an execution environment, providing the necessary abstraction of the interconnecting network. While distributed operating systems like Helios provide and prescribe a specialized support for distributed computing in general, and therefore allow a kind of abstraction from single processing nodes, this support is missed on a network of general purpose workstations. In this context facilities like remote procedure calls and remote execution can only be seen as a basic access to the potential summable computational power of a workstation cluster.

The PARNET computation environment is an approach to provide better access to the overall computation power in distributed environments. Needed administration for distributed objects can be done on a per application base, instead of the operating

*This research project is supported by the Deutsche Forschungs-Gemeinschaft

system level. So the PARNET functionality is a subset of what could be expected from a distributed operating system. Typical problems of these systems can be avoided or at least alleviated and implemented more efficiently.

2 PARNET

2.1 Overall Structure

The overall structure of PARNET is a per application distributed server. The parts of this server are constructed following the layer–model shown in figure 1. In each processing node, there is at least one part of the server running. The base layer handles main topics of abstraction from the network environment and provides low level message passing and semaphore based synchronization crossing the machine boundaries. The application programmer can use the base layer interface directly or rely on the next abstraction level, provided by the interface layer. This layer provides more complex synchronization features in an object oriented fashion. Build on top of this layer some framework for implementing parallel algorithms is provided. Here we show a PGA-layer matching the needs for coarse grained parallel processing of genetc algorithms. But it may be for example exchanged by another layer, e. g. supporting a general framework for event driven simulation.

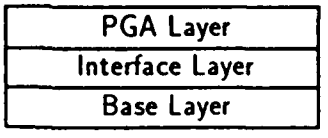


Figure 1: PARNET layer model

The implementation is built on the widely accepted programming model of multi-threaded tasks, available on state of the art operating systems (i.e. Mach, Solaris 2.x, Helios) together with a high reliable communication facility (TCP/IP, Helios message passing) [5]. The thread programming model must at least provide a *fork()* call to start threads and binary semaphore operations (*P(), V() or Signal(), Wait()*).

2.2 Base-Layer

Providing a flexible and extensible base environment leads to the main functionality of the base layer:

- initial booting from the local workstation,
- crash detection for remote computing nodes,
- hierarchical organized naming scheme for several kinds of objects,
- efficient message transfers by building optimal sized data packets,
- dynamic creation of name addressable communication ports.

To operate on a per application base, the PARNET environment is bound to the application itself and spreads out beginning from the local machine. Because of this mechanism an errorness application can only crash down itself, whereas a system service like implementation may affect further applications directly or indirectly.

A minimal necessary support for distributed applications is a crash detection facility. Any desirable crash recovery is at least somehow coupled with the application, or leads to a specialized parallelization paradigm, for which the problem of recalling a past state can be solved efficiently. Because we will not concentrate on a specialized parallelization method in the base layer, we support crash detection and following shutdown of the application.

For naming and addressing of distributed objects in PARNET, we provide a hierarchical name space. Again we benefit from the per application approach in PARNET. It is obvious, that an application can trust itself, and therefore no access control or authorization is needed. The object name–tree is distributed between the server parts, allowing local interpretation per context. A caching strategy is used to avoid unnecessary communication requests in searching objects (*locate()*). Figure 2 shows a snapshot of a typical name space. Terms in quotation marks denote user created objects, which may be functions, classes and their methods, mail-ports and further object names.

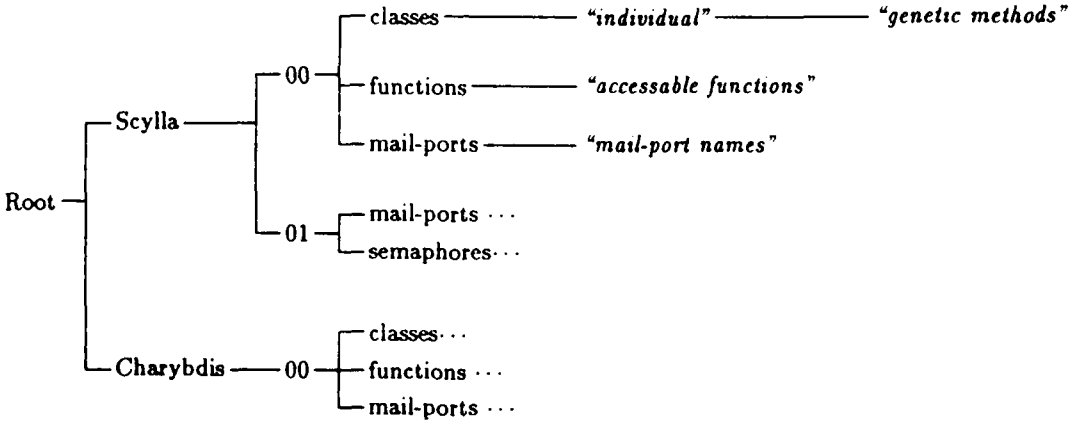


Figure 2: PARNET name space

The addressing of any object in PARNET is based on this naming-scheme. As an example, if we address data to "/scylla/01/functions/beep", on host "scylla", server-part "01", a function beep is called.

Addressing data to "mail-port/incoming_data" will search for a mail-port with the name "incoming_data" first in the local context and second in a remote context. The first occurrence of the name is the address the data will be delivered to.

Although communication shows to be still the bottleneck in most distributed applications, some afford can be done, to alleviate this fact [8].In our multithreaded implemen-

tation there is a sending thread for data communication to each remote server-part. So sending data occurs only if this thread is actually executing. It is a quite natural approach to pack together as many messages as currently available in order to have a good chance to achieve efficient transfer rates.

Traditionally only static structures for communication are realized in distributed applications. This fact is normally imposed by the underlying support-tool (Interface Description Language (RPC), or Component Distribution Language – Helios). Hence dynamic creation of communication ports is an unusual feature, allowing to send data to mail-ports which are not created so far and assuming, that the addressed mail-port will be created in near future. There are several situations where this may simplify the application code and avoids explicit synchronization, before starting to communicate.

2.3 Interface Layer

Object oriented programming has proved to be a worthy software engineering approach. An object is described by a class defining a data type for which access to data is restricted to a specific set of access functions. The PARNET base layer is designed to handle a wide range of parallel distributed applications. Hence access at the abstraction level of the base layer is indispensable. Interface classes are used to provide access to the PARNET base layer at a higher level of well known concepts of parallel programming.

The PARNET interface layer uses the model of invoking objects remotely to provide access to remote data and control of remote threads. Application data can be declared as a class and can be handled as an object thereafter. Threads of coarse grained parallelity are defined as a set of member functions of a class. Access to remote objects is provided via numeric object identifiers.

The idea of using classes leads to the object oriented model of inheritance, which is extensively used in the PARNET interface. A subclass can be derived from its base class, while the properties of the base class are inherited to the subclass. Inheritance is used to gain access to the base layer at different levels of well known programming concepts. The level of abstraction grows analogous to class tree inheritance. Access is permitted at any level of abstraction. Calling a meaningful operation on application level forces the processing of a unique chain of arbitrary operations along the directed graph of the interface class hierarchy.

Let us consider a simple example. At a low level of abstraction a locking primitive is provided by the PARNET base layer. The class *Monitor* is derived from *Locking* and provides a Hoare monitor using the inherited properties from *Locking*. In a next step *Monitor* inherits its functionality to *Semaphore*, which builds a counting semaphore using *Monitor* methods. In a last step *Fifo Semaphore* is derived from *Semaphore*, using a couple of semaphores to construct the Fifo behavior.

Figure 3 introduces the simplified PARNET interface hierarchy of derived classes. Base classes (appear in roman characters) are wrappers around the PARNET base layer.

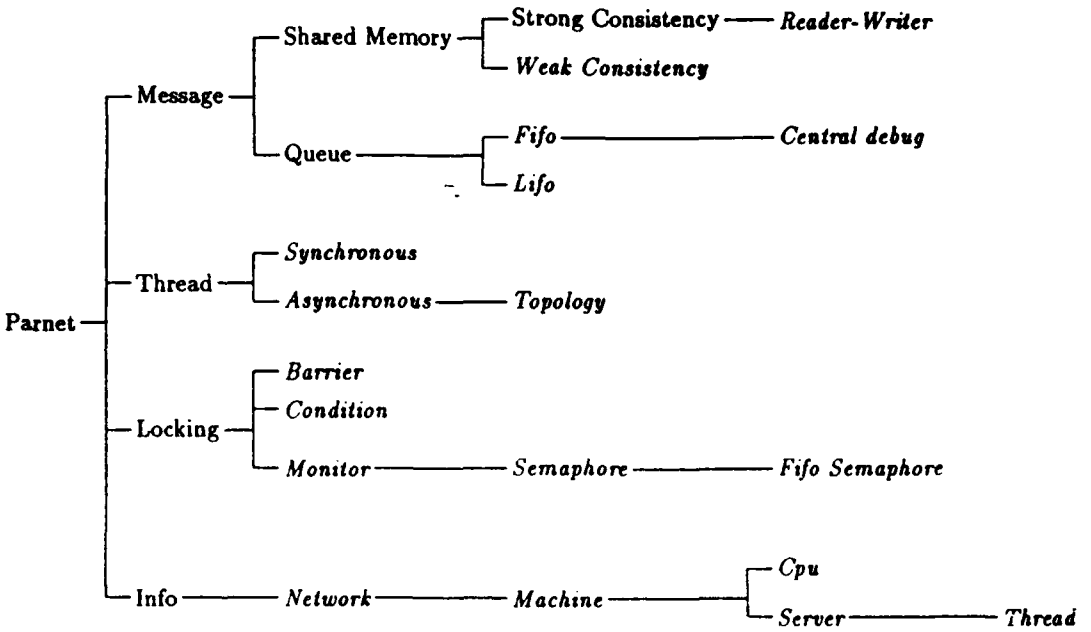


Figure 3: The PARNET interface hierarchy

These abstract classes cannot be used by the application programmer. They are base classes from which user classes at a higher level of abstraction are derived (appear in emphasized characters).

Four base classes represent the functionality of the PARNET base layer. The *Message* class derives two more base classes, which provide different communication interfaces to the programmer. Different kinds of *Queues* are provided as well as a distributed virtual *Shared Memory*. Above the latter class well known communication patterns like *single-writer-multiple-reader* can be built using the *Semaphore* class. Threads may be spawned in a *Synchronous* or in an *Asynchronous* way; a kind of processor group facility is provided by the *Topology* class. The *Locking* base class inherits its functionality to *Barrier*, *Condition*, and *Monitor*. These classes are used to synchronize threads. To enable threads to react on the network computing environment, an *Info* class is provided.

The program development of parallel applications is simplified by the usage of shared memory in comparison to message passing. A drawback using shared memory in a distributed environment is the relatively slow communication medium (e. g. ethernet). The PARNET interface provides features to take decisions on the distribution, protection and consistency of shared data to minimize the base layer network traffic.

Using this interface non trivial distributed parallel applications can be built, whose complexity regarding control and communication patterns surpasses usual remote pro-

cedure calls. Once invoked, threads may have "a life of their own" communicating with other threads without the need of a central instance. This is an important demand for using the interface by Parallel Genetic Algorithms.

2.4 A PGA Layer

A PGA application can be characterized by the following issues: All evolved threads perform the same non trivial procedure of iterations. No update of systemwide information has to be done frequently. Instead, global information is either readonly or rare and of weak consistency condition. Most steps of execution can be done using local data, which is private to each thread.

```
class individual derived from Parnet      // Class uses PARNET interface
    shared  data my_solution              // My own solution
    shared  data all_neighbors            // Other solutions I know

    virtual function terminate            // Members to be declared in
        ...                               // derived classes
    virtual function accept

    function main                         // The predefined GA interface
        while not terminate ()
            selected_neighbor = select    (all_neighbors)
            temp_solution     = crossover (my_solution, selected_neighbor)
            temp_solution     = mutation  (temp_solution)
            temp_solution     = local-opt (temp_solution)
            if accept (temp_solution, all_neighbors)
                 my_solution  = temp_solution
        end while
    end function
end class
```

Figure 4: An abstract PGA interface class using virtual functions

In the following the pseudo code of a GA thread class is presented as an example for the implementation of user interface classes. Figure 4 shows an abstract class *Individual*, which is derived from several PARNET interface classes to provide remote facilities. The shared data represents the current solution in the process of iteration and the access to the solutions of other executing *individuals*. Write access is permitted to *my_solution*, which may be updated by the individual. The access to *all_neighbors* is protected by a readonly constraint.

Apart from shared data, there is only one member function *main* defined. This is the loop called at invocation time by the thread class. It contains the skeleton of a generic PGA [6]. For a number of iterations defined by *terminate* the function *select* chooses a suitable partner for its recombination. The *crossover* operator recombines the genetic code of the current solution with the selected neighbor to a new (temporary) solution.

```
class my_individual derived from individual
    virtual function terminate
        body of function ...              // Define the implementation to be
    end function                          // executed in the interface method main
        ...
    virtual function accept
         body of function ...
    end function
end class
```

Figure 5: A user defined PGA class

Mutation and *local-opt* are optional operations on the new solution. The *accept* operation compares the recombined new data with the solutions of the neighborhood and indicates whether the old solution is to be overwritten by the new one. Otherwise the new solution could not dominate the older one and is discharged immediately. In the case of overwriting, the new solution is forwarded to the individuals being neighbors to the one we looked at.

No object of *individual* can be invoked. Instead, the abstract class inherits its interface to a user defined class. This user defined class is responsible for the implementation of the functions declared virtual in the base class. Figure 5 shows an implementation class derived from *individual*, defining the details of the PGA operations used by the interface class.

```
program
    topology<my_individual>  population   // Declare a topology of threads

    population.number(32)                 // Define the number of individuals
    population.exec()                     // asynchronous start of remote threads
        ...
    population.wait()                     // Wait for termination of all threads
end program
```

Figure 6: An program example using the PGA interface

Figure 6 gives a program example of the above described interface. A topology of individuals is defined within the *population* object. The member function *number* is called to determine the size of the population. The *exec* member invokes 32 remote individuals asynchronously. The *wait* member blocks until all individuals terminate. The called member functions are defined in base classes at different levels of abstraction.

The definition of the methods, introduced in figure 6, are derived from PARNET classes. A review to figure 3 may help making the class interdependencies more transparent. The method *exec* lives in class *Thread*, while *wait* is defined in class *Asynchronous* and *number* is a member of class *topology*. The resulting source code represents the structure of the program in a obvious way hiding the details of networking.

3 Conclusion

We presented the PARNET approach, leading from single machine computing environments to a realization of a distributed application environment in a workstation cluster. The object oriented approach fits best the coarse grained parallelism we wish to realize. Following these approaches a framework for parallel genetic algorithms has been introduced.

Well known network facilities, socket based communication, remote execution and remote procedure calls do not provide the desired programming support. A specialized distributed operating system is missing the desired robustness for carrying out everydays work [2]. The PARNET approach situated above the base operation system pursues a per application environment to avoid or at least alleviate typical problems of distributed computing.

The PARNET idea has not been the only approach until now. A programming tool, called PVM, with little different goals and a more different conception has recently become available [3]. Some other related work concentrate on the object oriented interfacing [2],[7],[4]. Some affords were made to hide nearly any distribution. Our work is more reflected by hiding hard to use details of networks and provide the ability to visibly express distribution.

References

[1] C. Bierwirth, D. C. Mattfeld, S. Stöppler, "Pseudo–Parallelity and Distributed Programming under UNIX–System V", Proc. PCMO, Siegen, in: Parallel Computation and Mathematical Optimization, Siegen 1991; M. Grauer, D. Pressmar (ed.), Springer 1991.

[2] S. Deshpande, P. Delisle, A. G. Daghi, "A Communication Facility for Distributed Object-Oriented Application", C++ Technical Conference, USENIX Association.

[3] G. A. Geist, V. S. Sunderam; "Network Based Concurrent Computing on the PVM System", Oak Ridge National Laboratory, TN 37831.

[4] D. Grunwald, "A Users Guide to AWESIME: An Object Oriented Parallel Programming and Simulation System", University of Colorado, 1991, technical report /CU-CS-552-91/.

[5] M. L. Powell et al, "Sun OS 5.0 Multithread Architecture", Sun Microsytems, Technical white paper, CA 1991.

[6] S. Stöppler, C. Bierwirth, "The Application of a Parallel Genetic Algorithm to the n/m/P/Cmax Flowshop Problem", in: New Directions for Operations Research in Manufactoring, G. Fandel, Th. Gulledge, A. Jones (ed.), Springer 1992.

[7] V. S. Sunderam, "Concurrent Programming with Shared Objects in Networked Environments", Emory University Atlanta, Oct. 1992, still unpublished.

[8] T. Utecht, H.U. Post, "Performance Analysis of Parallel Applications by Monitoring the Interprocess Communication", Proc. of EWPC'92, Barcelona, in: Parallel Computing: From Theory to Sound Practice, W. Joosen, E. Milgrom (ed.), IOS Press 1992.

Spreadsheet-Based Model Building and Multiple Criteria Group Decision Support*

Miklós Biró, Attila Bor, Előd Knuth, Tibor Remzső, András Szilléry

Computer and Automation Institute
Hungarian Academy of Sciences
Budapest, Kende u. 13-17. H-1111 Hungary

The purpose of this paper is the description of a system that, while fully exploiting and integrating the advanced features of the Microsoft Excel commercial spreadsheet software and Hewlett-Packard's NewWave office environment, extends them with new optimisation and multiple criteria group decision model building capabilities.

The new features include a meta-model building language which allows the automatic generation of a class of mathematical programming spreadsheet models dynamically linked to public and private databases. These models can immediately be used for both intuitive experimentation and optimisation. Figure 1 shows a sample model.

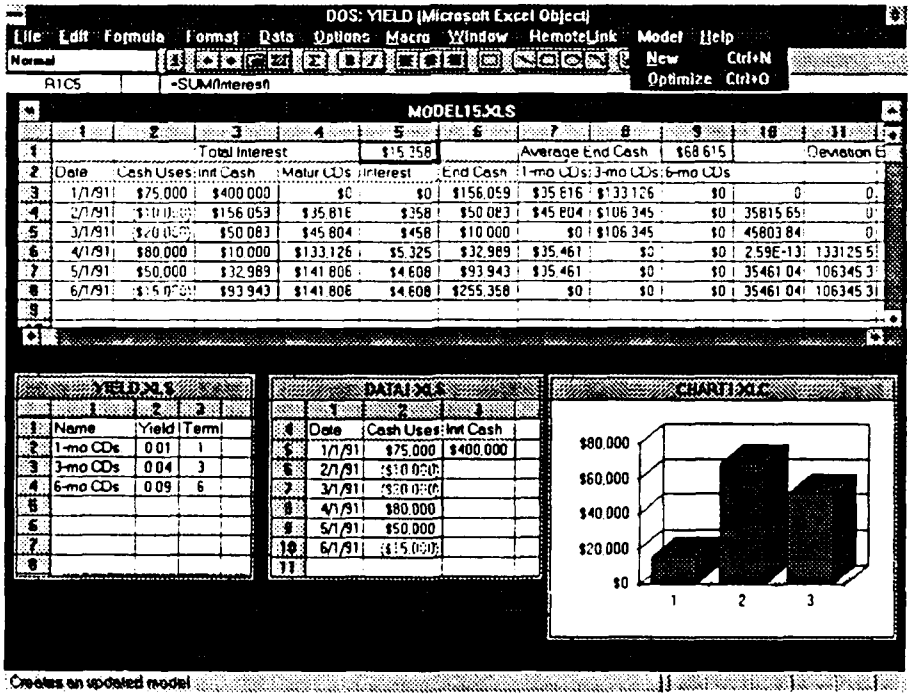


Figure 1

*Supported by OMFB 91-97-11-0005, OTKA #2567, #2571, et #2575.

An archiving tool allows the user to save and later retrieve any given state of the model together with a freely selectable set of characteristic indicators. The indicators belonging to different saved states of the model can be easily compared using graphical charts. Figure 2 shows the chart used to compare the saved states.

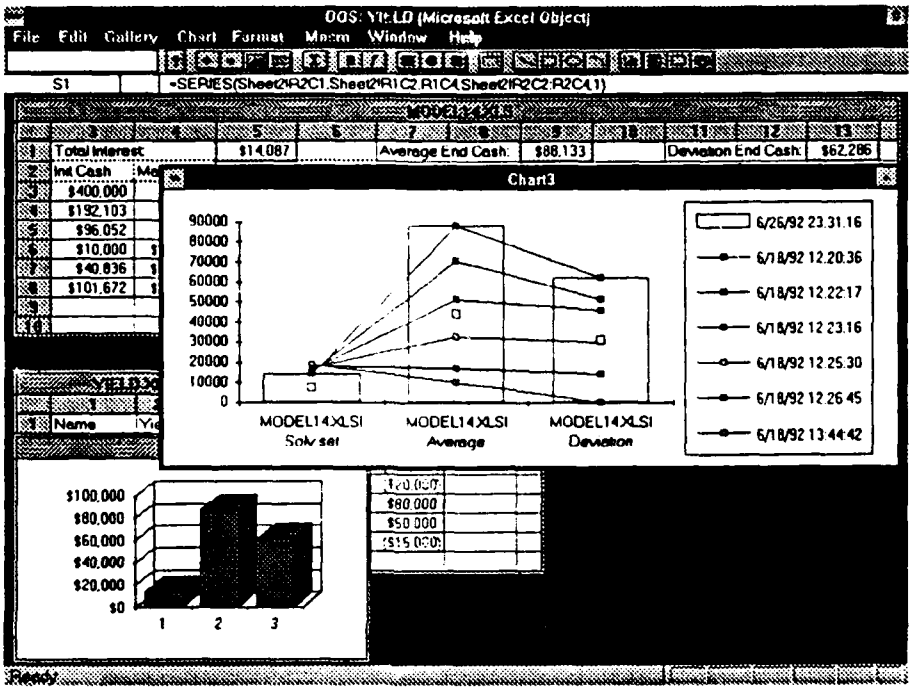


Figure 2

The system works in both the Apple Macintosh and the Microsoft Windows environments. Its functionality is significantly enhanced with inter application communication and dynamic data exchange tools that we extended beyond the built in capabilities of the commercial environments to computers connected by a network. The optimisation and multiple criteria decision making can in this way be performed in an environment where dynamically changing data originating from shared data bases or other members of the decision making group are permanently taken into account.

The decision makers' communication needs are supported at two different levels. The first level is the already mentioned dynamic data exchange. A hot link to data can be established without requiring the intervention of the user who is linked to the data. Figure 3 shows the corresponding interface under Microsoft Excel.

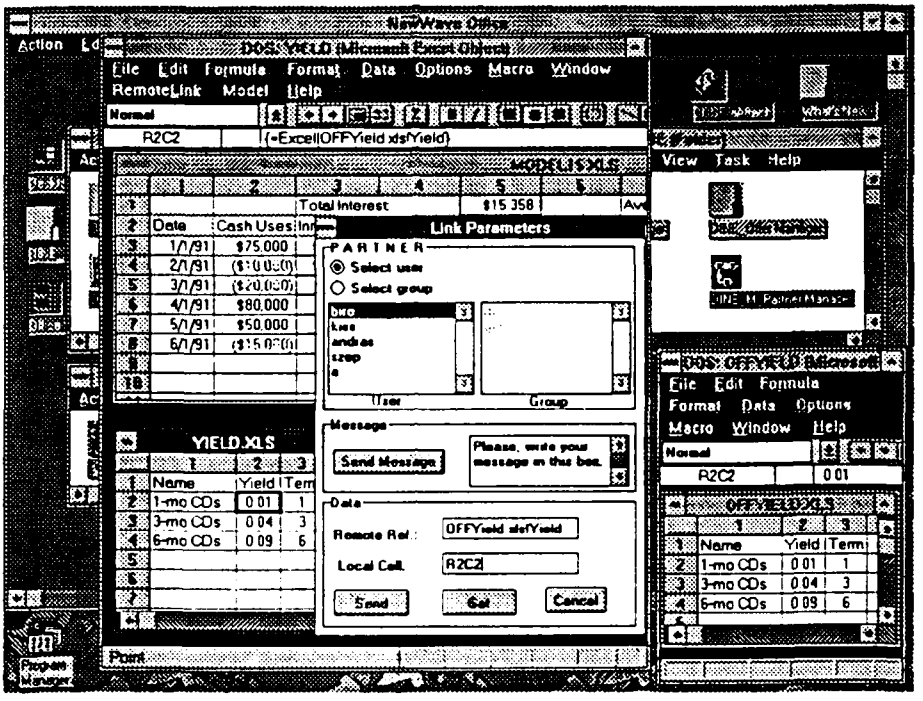


Figure 3

The second level is implemented under Hewlett-Packard NewWave. It lets the partners view all objects on each others' office desk. They can even copy or move objects to or from their partners' desk. Of course, permissions for viewing, copying, moving etc... can be selectively assigned to all types of objects. This feature complementing the functionality of NewWave is only implemented on a PC under Microsoft Windows. Figure 4 shows the view of the office desk of another partner.

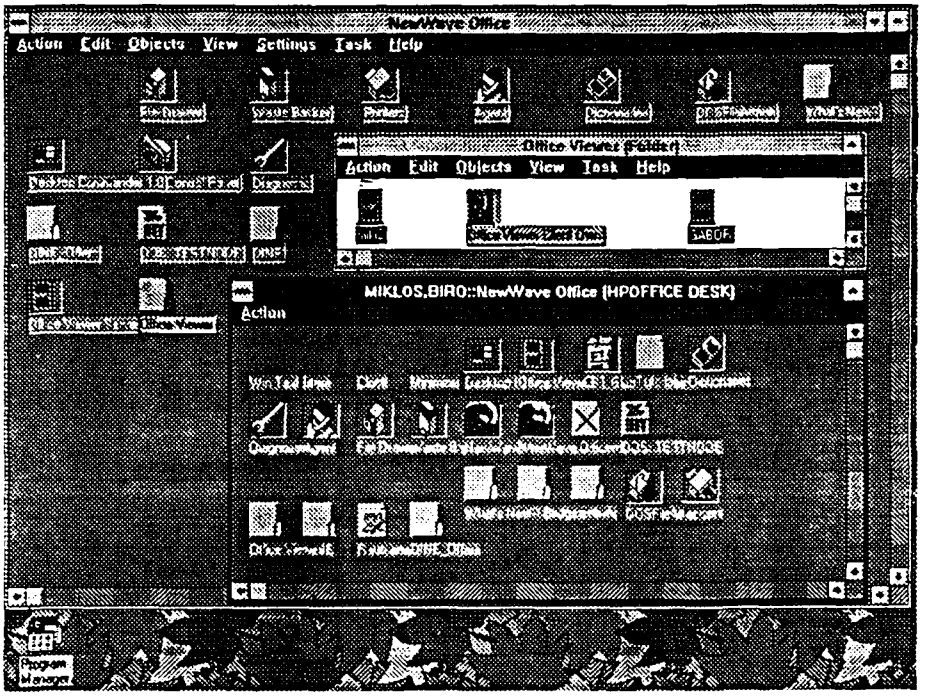


Figure 4

References

[Angehrn 1990] A.A.Angehrn, Supporting Multicriteria Decision Making: New Perspectives and New Systems, Proceedings of the First Conference of the International Society of Decision Support Systems, Austin, Texas (1990).

[Binbasioglu, Jarke 1986] M. Binbasioglu and M. Jarke, Domain Specific DSS Tools for Knowledge-Based Model Building, Decision Support Systems, 2(1986)213-223.

[Biró, Csáki, Vermes 1991] Biró,M.; Csáki,P.; Vermes,M. WINGDSS Group Decision Support System under MS-Windows. In: Proceedings of the Second Conference on Artificial Intelligence (ed. by I.Fekete and P.Koch). (John von Neumann Society for Computer Sciences, Budapest, Hungary, 1991) pp.263-274.

[Biró, Maros 1992] Biró,M.; Maros,I. The Use of Deep Knowledge from the Perspectives of Cooperative Problem Solving, Systems Modeling, and Cognitive Psychology. In: Shifting Paradigms in Software Engineering (ed. R. Mittermeir). (Springer-Verlag, Wien, New York, 1992) pp.56-67.

[Biró, Bodroghy, Bor, Knuth, Kovács 1992] Biró,M.; Bodroghy,E.; Bor,A.; Knuth,E.; Kovács,L. The Design of DINE: a DIstributed NEgotiation Support Sheil. In: Decision Support Systems: Experiences and Expectations, (ed. by T. Jelassi, M.R. Klein, W.M. Mayon-White). IFIP Transactions A-9 (North-Holland, 1992) pp.103-114.

[Biró, Jamrik, Janek, Knuth, Remzsô 1992] Biró,M.; Jamrik,F., Janek,G., Knuth,E.; Remzsô,T. Factors Influencing the Design of a Prototype Decision Support Application for a Distributed Object Management Project. In: Proceedings of the Fourth IFAC/IFIP Workshop on Experience with the Management of Software Projects, Schloss Seggau, Austria. (Pergamon Press, 1992) (to appear)

[Brans, Vincke, Marechal 1986] J.P. Brans, Ph. Vincke and B. Marechal, How to Select and how to rank projects: The PROMETHEE-method, European Journal of Operational Research, 24(1986)228-238.

[Gass 1990] S.I. Gass, Model World: Danger, Beware the User as Modeler, Interfaces, 20, no.3(1990)62-64.

[Geoffrion 1987] A.M. Geoffrion, An Introduction to Structured Modeling, Management Science, 33(5)(1987)547-588.

[Gerlach, Kuo 1990] J. Gerlach and F. Kuo, An Approach to Dialog Management for Presentation and Manipulation of Composite Models in Decision Support Systems, Decision Support Systems, 6(1990)227-242.

[Gupta 1989] S.Gupta, Modeling Integrative Multiple Issue Bargaining, Management Science, 35,7(1989)788-806.

[Jarke, Jelassi, Shakun 1987] M.Jarke, M.T.Jelassi, M.F.Shakun, MEDIATOR: Towards a Negotiation Support System, European Journal of Operational Research, 31(1987)314-334.

[Jelassi, Foroughi 1989] M.T.Jelassi and A.Foroughi, Negotiation Support Systems: An Overview of Design Issues and Existing Software, Decision Support Systems, 5(1989)167-181.

[Keeney, Raiffa 1976] R.Keeney and H.Raiffa, Decisions with Multiple-Objectives: Preferences and Value Tradeoffs (Wiley, New York, 1976)

[Kersten, Szapiro 1986] G.E.Kersten, T.Szapiro, Generalized Approach to Modeling Negotiations, European Journal of Operational Research, 26(1986)124-142.

[Ma, Murphy, Stohr 1989] P. Ma, F.H. Murphy and E.A. Stohr, Representing Knowledge about Linear Programming Formulation, Annals of Operations Research, 21(1989)149-172.

[Matwin, Szpakowicz, Koperczak, Kersten, Michalowski 1989] S.Matwin, S.Szpakowicz, Z.Koperczak, G.E.Kersten, W.Michalowski, Negoplan: An Expert System Shell for Negotiation Support, IEEE Expert, Winter(1989).

[Murphy, Stohr 1986] F.H. Murphy and E.A. Stohr, An Intelligent System for Formulating Linear Programs, Decision Support Systems, 2(1986)39-47.

[Murphy, Stohr, Asthana 1992] F.H. Murphy, E.A. Stohr and A. Asthana, Representation Schemes for Linear Programming Models, Management Science, 38, no.7(1992)964-991.

[Müller-Merbach 1983] H. Müller-Merbach, Model Design Based on the Systems Approach, J. Opl. Res Soc., 34, no.8(1983)739-751.

[Roy, Lasdon, Plane 1989] A. Roy, L. Lasdon and D. Plane, End-user optimization with spreadsheet models, European Journal of Operational Research, 39(1989)131-137.

[Roy, Vincke 1981] B. Roy and Ph. Vincke, Multicriteria analysis: Survey and new tendencies, European Journal of Operational Research, 8(1981)207-218.

[Saaty 1980] T.L. Saaty, The Analytic Hierarchy Process (McGraw Hill, 1980).

[Sprague, Carlson 1982] R.H.Sprague and E.D.Carlson, Building Effective Decision Support Systems (Prentice Hall, Englewood Cliffs, N.J. 1982).

جمهوری اسلامی ایران

IRAN UNIVERSITY OF SCIENCE & TECHNOLOGY

TRAFFIC MODELS FOR THE ECONOMIC EVALUATION OF PRIVATE SECTOR TOLL ROADS AND TUNNELS IN AUSTRALIA

J. A. Black

Department of Transportation Engineering, School of Civil Enginering, University of New South Wales, Kensington, NSW 2033 Australia.

S. M. Seyed-Hosseini

Department of Industry and Management, Iran University of Science and Technology, Narmak, Tehran-16, Iran.

Abstract - *The objectives of this paper are to provide the context for private sector tollways in Australia, to explain the traffic forecasting methods employed, and to identify the evaluation criteria. Attention is focused on the technical aspects of the traffic estimation and evaluation process which takes into account traffic forecasting techniques, traffic diversion and assignment, the temporal distribution of peak-hour traffic demands, vehicle operating costs and fuel consumption, monetary values of travel time and discount rates.*

1. INTRODUCTION

Traditionally, the main roads in Australia have been constructed by state governments through their Department of Main Roads. However, in the state of New South Wales in recent years, the policy has shifted to accommodate, and then encourage, private sector participation in the construction and operation of road facilities. The 400 million Sydney Harbour Tunnel, which has been under construction since early 1989, is an example where the state government has allowed the construction of a road facility funded by a private consortium. Furthermore, the Department of Main Roads (now Roads and Traffic Authority) of the state of New South Wales has called for expressions of interest for three privately funded toll roads - namely: the Buladelah Tollway, on the Pacific Highway, about 250km north of Sydney; the F4 toll road in the north western sector of the Sydney metropolitan area; and the F2 toll road in the western fringe of Sydney. Attracting private funds will allow the Roads and Traffic Authority to accelerate the construction of major road projects in accordance with the Roads 2000 Plan. The total funds available from both Commonwealth Government sources and State sources to the Department of Main Roads in 1986/87, for

ECONOMIC ANALYSIS OF CROSS HARBOUR TRANSPORT

Accuracy of the economic evaluation of toll roads depends on successful forecasting of the level of traffic using the transport facility. Figure 1 shows the methodology adopted by the senior author in the economic evaluation of the Sydney Harbour Tunnel project and other alternative cross harbour transport proposals. The four lane, 2.4 km long tunnel, now under construction, is, conceptually, a parallel facility to the existing bridge, which has 8 road lanes and two train tracks connecting the north and south banks of Port Jackson and the Parramatta River (the Sydney harbour). The investment and operating costs will be recovered by the toll levied on traffic crossing the harbour using either the (existing) bridge or the tunnel (when opened in late 1992), over a 35 year period commencing on May 1987.

The topmost cell in Figure 1 refers to traffic projections related to Average Annual Daily Traffic (AADT). In 1985, AADT on the existing Harbour Bridge was 178,180 vehicles (DMR, 1986a, b). DMR (1986a) provides the following traffic projections for the Sydney Harbour Bridge, based on past traffic trends.

$$\text{Max Growth } \hat{Y}_{max} = 178372 + 3358x$$

$$\text{Min Growth } \hat{Y}_{min} = \frac{200000}{1+0.14e^{-0.09x}}$$

where,

$\hat{Y}_{max}$ = estimate of AADT at maximum growth rate,

$\hat{Y}_{min}$ = estimate of AADT at minimum growth rate; and

x = number of years from the base year 1985.

Note that in these two formulae the maximum growth implies a constant annual increment of some 3360 AADT each year, whereas the minimum growth is based on a daily upper "saturation level" of 200,000 vehicles for AADT.

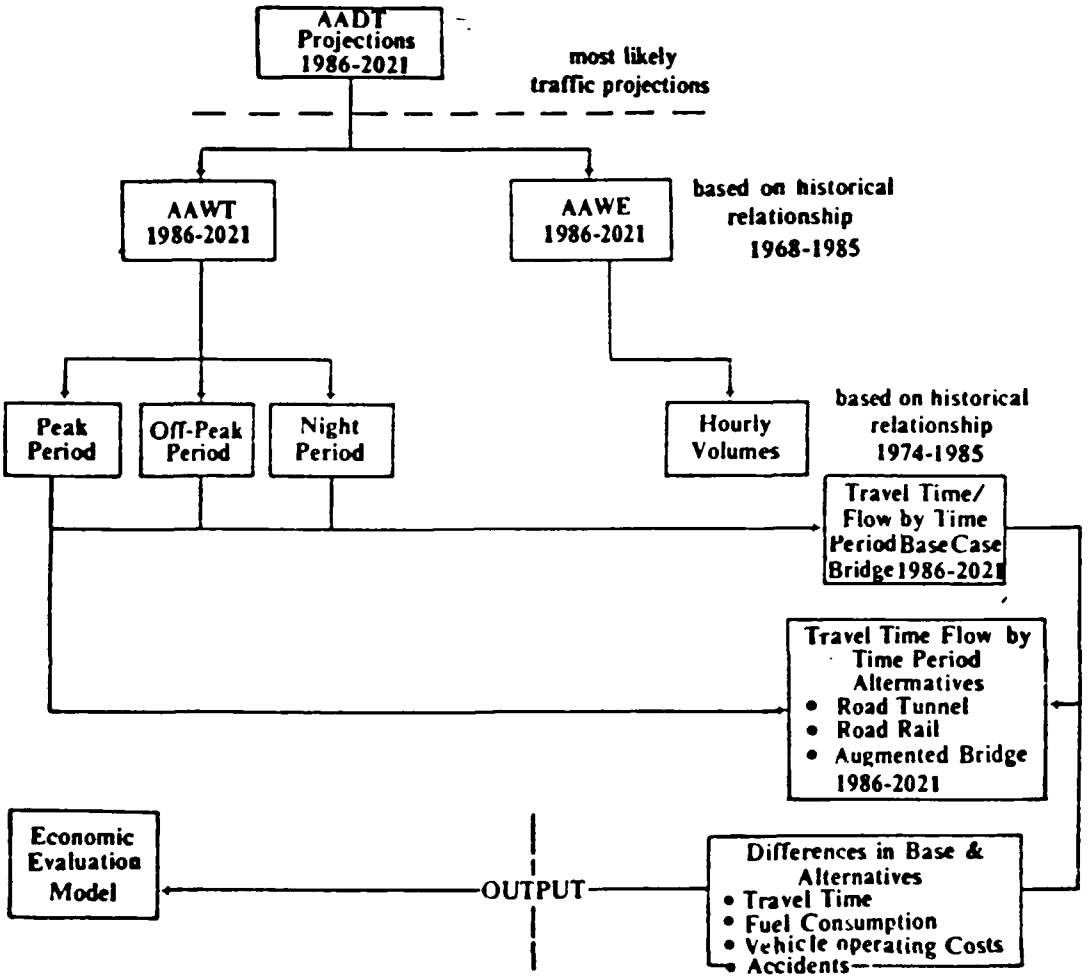


Fig. 1: Flow Diagram of Traffic Forecasting Methodology and Economic Evaluation Model for Sydney Cross-Harbour Transport

Gutteridge, Haskins and Davey (1986) - the traffic consultants to the Sydney Harbour Tunnel: Transfield-Kumagai Joint Venture - approached traffic projections from a growth in southbound traffic on the Sydney Harbour Bridge for Average Annual Weekday Traffic (AAWT). The shape of the mathematical function is a logistic curve based on "a strongly linear historic trend and a long term growth constraint, based on a maximum Service Volume, creating a mathematical asymptote for the growth curve (Gutteridge, Haskins and Davey, 1986). Traffic projections for Average Annual Weekday Traffic (AAWT) up to 2021 are illustrated by Cameron McNamara (1986a), the company responsible for preparing the tunnel environmental impact statement (EIS). These include "high", "most likely" and "low" projections. For example, the high projection is calculated from:

$$\hat{Y}_{WT} = \frac{135000}{1+10^{(0.6435 + 0.0281x)}}$$

where

$\hat{Y}_{WT}$ = estimate of AAWT southbound on bridge and tunnel; and

x = number of years from the tunnel openning year 1992.

The, "most likely" traffic projection assumed great importance in the final appraisal of the project because it formed the basis of the guaranteed revenue stream for the developers that was "underwritten" by government. An independent, expert review of the traffic forecasts and economic evaluations was sought from Unisearch Ltd (1987a).

It is at this point that our approach differs from that of the consultants to the Joint Venturers, and an original traffic model was developed. The preferred approach is to make projections of AADT and then partition them into average annual weekday traffic (AAWT) and Average Annual Weekend Traffic (AAWE). Based on time series data of traffic volumes on the Sydney Harbour Bridge from 1968 to 1985, regression analysis leads to the following relationship:

$$\hat{Y}_{WE} = 4650 = 0.766Y (r^2 = 0.99)$$

where,

$\hat{Y}_{WE}$ = estimate of average annual weekend traffic (AAWE); and

Y = average annual daily traffic (AADT).

Average Annual Daily Traffic (AADT) equals five-sevenths of Average Annual Weekday Taffic (AAWT) plus two sevenths of Average Annual Weekend Traffic (AAWE). By rearranging, and ignoring public holidays:

$$AAWT = 7/5(AADT - 2/7AAWE).$$

For the calculation of the costs and operational benefits associated with the tunnel proposal, three time periods for each weekday are defined:

Peak periods -	7 to 10 am and 4 to 7 pm;
off-peak periods -	10 am to 4 pm and 7 to 11 pm; and
night period -	11 pm to 7 am.

As travel times are flow dependent it was necessary to estimate typical hourly traffic flows for these three time periods over the life of the project (for evaluation purposes taken to be up to 2021). Based on historical data, regression analyses of the temporal distribution of traffic, as a function of Average Annual Weekday Taffic, leads to the following equations:

$$\hat{Y}_p = 6.593 Y_{WT}^{0.78} \quad (r^2 = 0.99)$$

$$\hat{Y}_{op} = 0.070 Y_{WT}^{1.16} \quad (r^2 = 1.00)$$

where,

$\hat{Y}_p$ = estimate of peak period traffic volumes;
$\hat{Y}_{op}$ = estimate of off-peak period traffic volumes; and
Y_{WT} = estimate of average weekday traffic volumes.

To ensure the temporal distribution of traffic is properly constrained by the total average daily traffic figure, the estimate of the night period traffic volumes $\hat{Y}_n$ becomes:

$$\hat{Y}_n = \hat{Y}_{WT} - \hat{Y}_P - \hat{Y}_{OP}.$$

Traffic assignment to a network requires the rate of demand to be established. The hourly traffic flows for the three time periods give the demand rate and the impact of projected traffic volumes on travel time were estimated using Davidson's (1966) travel time/traffic flow relationship. Thus, travel times at different times of the day are computed firstly for the bridge only situation and then for the bridge with tunnel alternative.

The bridge only base case and the tunnel alternative can be compared in the form of four different measures: travel time saving; fuel consumption; vehicle operating costs; and accident savings (Table 1.) As mentioned before, the travel time differences were based on Davidson's model. Differences in fuel consumption were computed using the model reported by Bowyer (et al., 1984, 1985). Differences in vehicle operating costs were computed using published data from New South Wales Road Freight Transport Industry Council, (1986) for trucks and Royal Auto (July, 1986) for motor vehicles. Abelson (1987) also provides comprehensive methods to compute vehicle operating costs. Accident costs, which average at 1 cent per vehicle kilometre, were based directly on Department of Main Roads data (Cameron McNamara, 1986b), and were applied to the distance saving of 800 metres via the tunnel made by 33 per cent of weekday and weekend traffic. Table 1 sets out some economic evaluation parameters: the

tunnel construction cost, its annual maintenance and operating cost, and the monetary items for travel time, vehicle operating cost, fuel saving and accident costs.

Table 1. Summary of Economic Evaluation Parameters for the Sydney Harbour Tunnel

Parameter	Value
Construction costs (limited clearance tunnel)	$395 millon
Annual operating / maintenance cost	$7.9 million
Weighted monetary value of time	$6.00 per peak hour (1992-1999) $7.70 per hour (1993 onwards)
Vehicle operating cost	$0.16 per veh / km
Vehicle occupancy	1.4 persons / vehicle
Vehicle accident costs	$0.01 per veh / km
Fuel savings	$0.55 per litre
Traffic estimation ("most likely")	$Y = \frac{135000}{1+10^{-(0.644+0.028x)}}$
for bridge and tunnel	where, Y = average annual weekday traffic southbound x = number of years from the tunnel openning year 1992
Benefit cost ratios	1.9 at 4 % p.a. discount rate 1.2 at 7% p.a. discount rate 0.8 at 10% p.a. discount rate

(Source: Cameron McNamara, 1986b)

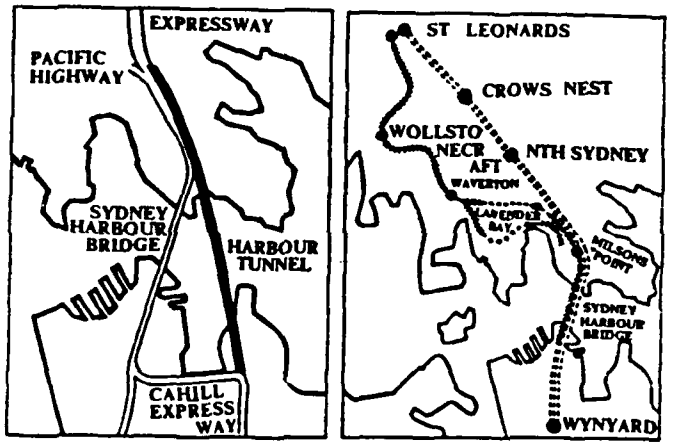
The basic economic evaluation parameters given in Table 1 were used both by the tunnel proponents and by Unisearch Ltd. The most likely traffic projection used by the consultants to the Joint Ventures is given in this table. From these inputs, the consultants undertook an economic evaluation of the tunnel proposal and estimated the benefit cost ratios as ranging from 1.9 to 0.8 depending on the discount rate adopted.

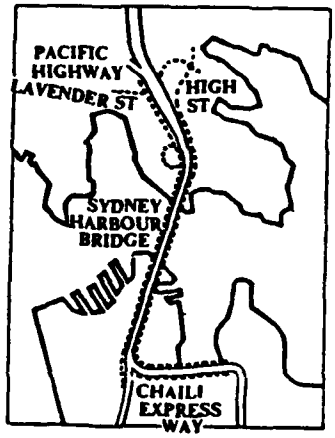
The independent economic evaluation by Unisearch Ltd (1987a) also used the values in Table 1 as inputs but applied the traffic model described above to give a more accurate representation of the temporal traffic flows over the bridge and tunnel. A micro-computer model was developed to facilitate sensitivity analyses for variations in costs and benefits and in the values of the economic evaluation parameters. This approach gave lower benefit cost ratios than those derived by the consultants to the joint ventures and given in Table 1 .

The additional advantage of the Unisearch Ltd approach was that it allowed a

ready comparison of other cross harbour transport proposals, such as an augmented Sydney Harbour Bridge and a cross harbour rail tunnel (Unisearch, 1987b). Figure 2 illustrates these alternatives together with the road tunnel proposal. The augmented bridge gave the highest benefit cost ratio, primarily because of its relatively low capital cost of $ 44 million. A rail tunnel, together with the extra two traffic lanes on the Sydney Harbour Bridge (replacing the existing rail tracks), gave a benefit cost ratio very similar to that of a road tunnel.

In May, 1987, the goverment decided to proceed with the tunnel project, despite widespread media criticism of its financial viability. The determination by the Commissioner for Main Roads used higher monetary values for travel time than those in Table 1 , and included a salvage value for the tunnel. The tunnel is under construction and is scheduled to open to road traffic in September, 1992. To date, traffic using the Sydney Harbour Bridge is below that forecast by the consultants to the joint Venturers.





AUGMENTED PROPOSAL BRIDGE.

Fig. 2: Sydney Harbour Tunnel and Alternative proposals.

ROUTE CHOICE MODELLING

Whereas route choice modelling was not necessary in the evaluation of the Sydney Harbour Tunnel because the traffic demands are split between parallel facilities according to Wardrop's principle of an equal travel time assignment, it was an important technical feature of traffic modelling of both rural tollways and metropolitan tollroads. The government's principle in the development of any toll road is that a "free" alternative route must be available for drivers. Therefore, a route choice model was developed by the authors to forecast future level of traffic on three separate toll road projects at various levels of toll charges. Price elasticity of demand for cars and trucks are available for number of US toll facilities, as shown in Table 2. In general, when the cost of using the toll facility is increased, the traffic volume is reduced.

Table 2. Price Elasticity of Demand for Cars and Commercial Vehicles on US Tollways

Location	Cars		Commercial Vehicles	
	Toll Increase (%)	Elasticity	Toll Increase (%)	Elasticity
1. ROADS				
Pennsylvania	24	-0.08	24	-0.06
New Jersey	20	-0.13	30	-0.17
Indiana	20	-0.31	30	-0.17
Massachusetts	30	-0.18	30	-0.17
Oklahoma 1	17	-0.21	33	-0.25
2	17	-0.30	33	-0.08
3	9	-0.25	22	-0.13
4	18	-0.25	56	-0.19
5	11	-0.31	44	-0.12
2. BRIDGES				
Delaware	20	-0.26	43	-0.25
Chesapeake Bay	15	-0.15	15	-0.26

(Source: based on Wuestfeld and Regan, 1981)

The authors applied a binary logit model to split the total traffic volume in a given corridor between the toll road and the non-toll road. Ben Akiva and Lerman (1979) show the applicability of logit formulation for choice modelling purposes. The formula for a binary choice logit model for route assignment is:

$$P(t) = \frac{1}{1 + \exp(\alpha + \beta_1 x_1 + \beta_2 x_2 + \beta_3 x_3)}$$

where,

$P(t)$ = probability of using the tollway;

x_1 = differences in route distances;

x_2 = differences in route travel times;
x_3 = differences in operating costs;
α = constant term; and
β_1,β_2,β_3 = coefficients.

However, because of the paucity of data the three explanatory variables are combined into one standard, composite, variable, called generalised cost - which is a combination of travel time and cost - as follows:

$$P(t) = \frac{1}{1 + \exp(\beta x)}$$

where x = differences in generalised cost.

The model was calibrated using the data obtained from Sydney - Wollongong Tollway (see Section 2) in 1987.

The above logit model was applied, for example, in traffic estimation and appraisal of a proposal for a tollway from Queanbeyan (near Canberra, Australia's capital city) to the South Coast of New South Wales. This particular application was part of the research and development of a consultancy project undertaken by all three authors on behalf of Unisearch Ltd. for a private - sector consortium. The essential features of the existing situation are described as follows. The distance from Canberra to Moruya (on the South Coast) via the Araluen Valley is 162 km. The Araluen Valley road is about 7 m wide and of gravel construction. It is a very steep mountain road with many tight curves and is presently not a feasible route for Canberra - South Coast traffic. For instance, field studies showed that driving from Moruya to Araluen on a Saturday morning in April (Autumn), only three cars and two motorcycles were observed. On the other hand, Canberra to Moruya, via the Kings Highway (Main Road 51) and Batemans Bay, is 152 km. From Braidwood to Batemans Bay the distance is 61 km, with a winding section of road through Clyde Mountain. During the weekdays, the traffic flow is light between Bungendore (25 km to the east to Queanbeyan) and Batemans Bay, because two thirds of traffic to and from Queanbeyan leaves Main road 51 at Bungendore, with destinations to and from Goulburn, located to the north.

Over a twenty - year period from 1967 to 1988, traffic counts by the Department of Main Roads, New South Wales, show that the number of vehicles using the Kings Highway has increased from about 1000 vehicles per day to about 3000 vehicles per day. However, the average traffic counts conceal both the seasonal, and weekend, characteristics of traffic. This general level of traffic activity noted above was confirmed by a survey undertaken by the local government autority, Tallaganda Shire Council in December, 1987.

Four different alignments proposed for the toll road, and considered in our analysis, are schematically shown in Figure 3. In the context of the wider land - use / transport system, we observe that the toll road concept is a sound one. It is consistent with current New South Wales Government policy on roads. An

alternative, free, road would be available to motorists, irrespective of any toll road alignment finally adopted. A toll road between Queanbeyan and Moruya would give genuine route distance saving to motorists (who currently use Kings Highway to gain access to and from the coast) from Goulburn, Canberra, and parts of country New South Wales (Fig. 4). The alignment would also make travel from Melbourne via Canberra to the South Coast almost as short as the currently favoured route via the Princes Highway (782 km compared with 742 km). Distance savings to road users represent genuine resource savings in fuel consumption and vehicle operation costs.

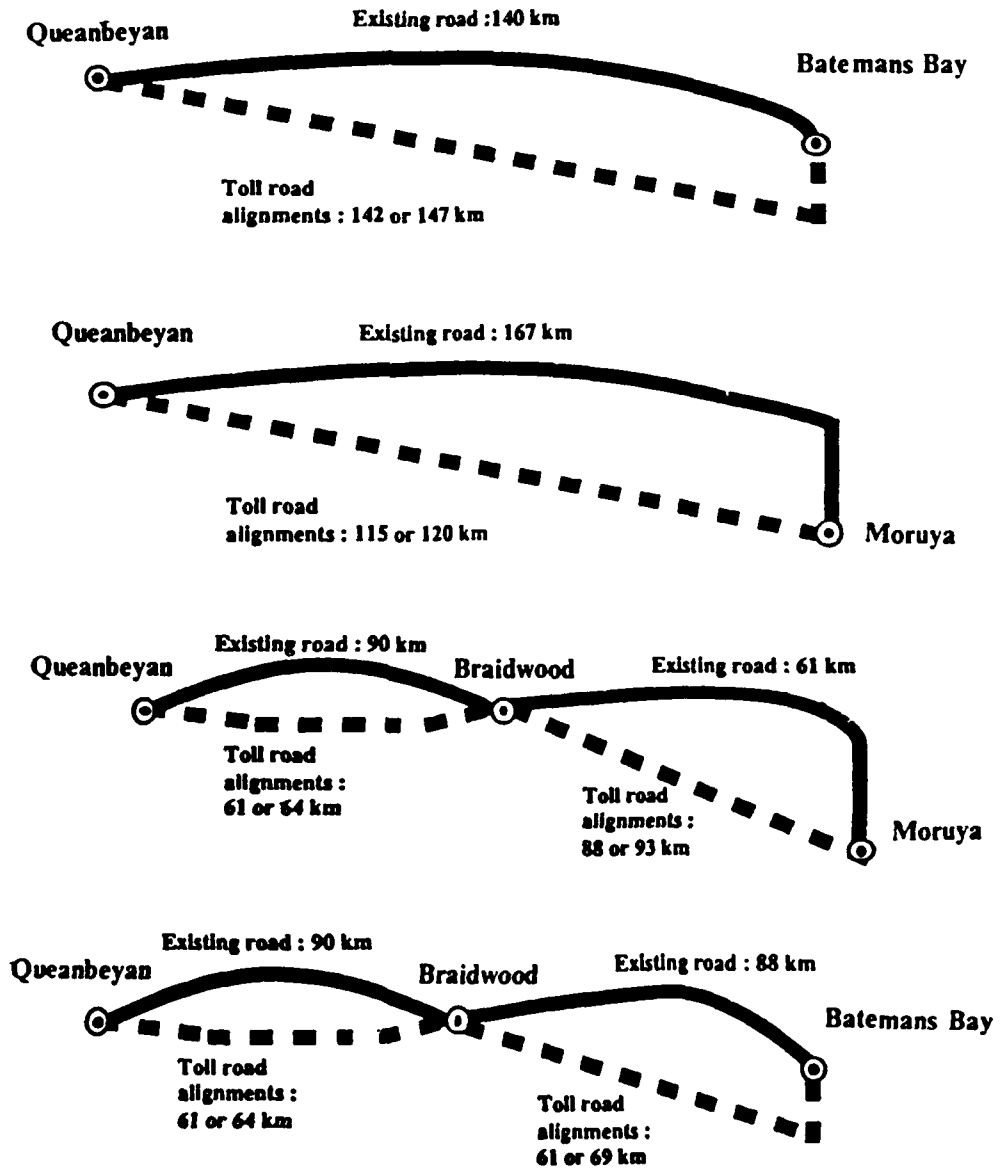


Fig. 3 Network Representation of Toll Road Alignments.

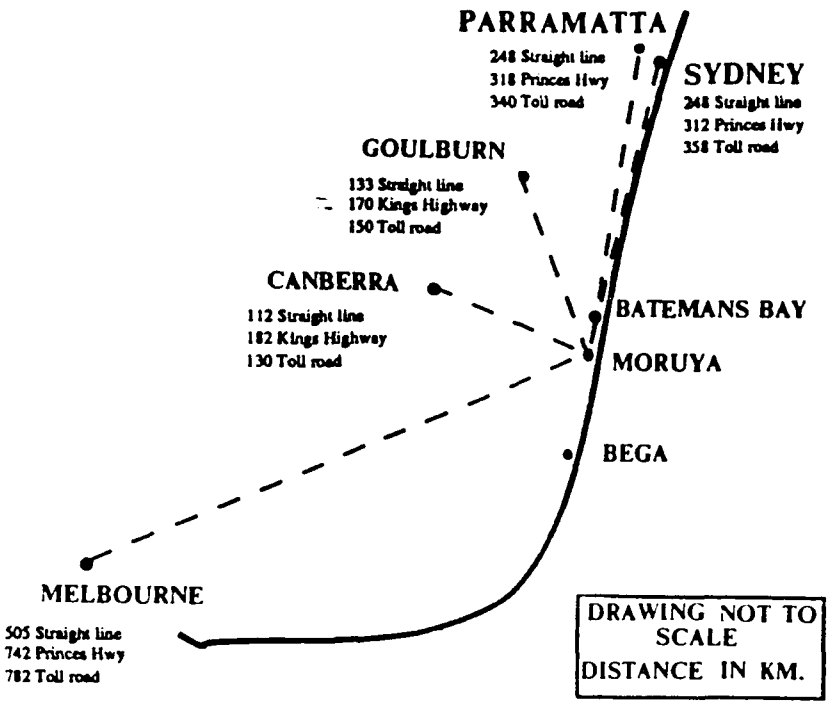


Fig 4. Effect of Toll Road on Distances to Moruya (South Coast).

The micro-computer-based traffic forecasting model that was developed allows a variety of assumptions about toll road characteristics to be analysed-length, tolls charged, whether tolls are indexed or unindexed, or whether drivers decide on their route choice because of total, or perceived, costs. Toll charged, value to travel time, and fuel consumption costs, are taken into account in the analysis based on perceived costs. In addition, vehicle operating cost (which includes depreciation and tyre wear) is included in the total costs based method. The level of traffic obtained from the total costs method is generally less than that obtained from the perceived costs method because of increased operating cost for toll road users. The results of this comprehensive analysis, using a logit model to take into account behavioural response of travellers for the two toll road alignments, have been presented elsewhere (Unisearch Ltd, 1989).

A brief summary of this comprehensive analysis makes the following points. Based on the historical growth of traffic on the Kings Highway, and traffic projection by trend extrapolation into the future, it can be shown that a toll road is not financially viable (given its construction, operation and maintenance costs). This is summarised in Fig 5 for the corridor between Queanbeyan and Batemans Bay and in Fig 6 for the corridor between Queanbeyan and Moruya. The graphs show the relative traffic levels for long and short alignments, for perceived and total costs, and for four toll levels ($5 for cars, $10 for trucks; $10 for cars, $20 for trucks; $20 for cars, $30 for trucks; and $30 for cars, $50 for trucks). In conclusion, the level of traffic on these toll road alignments, considerng the historical growth of traffic alone, is insufficient to justify investment.

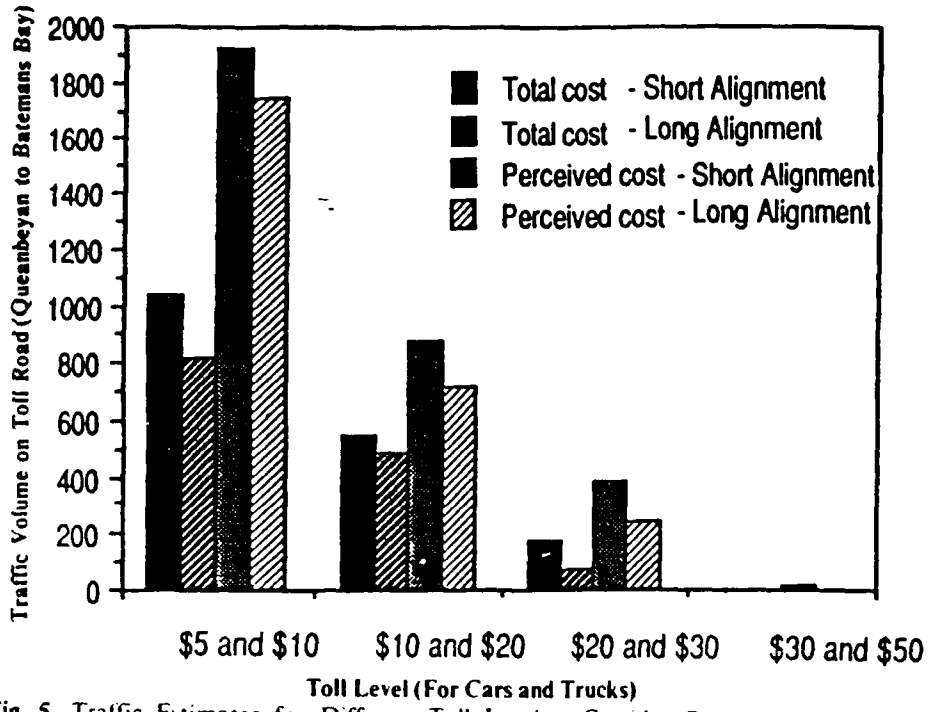


Fig 5. Traffic Estimates for Different Toll Levels - Corridor Between Queanbeyan Batemans Bay (South Coast)

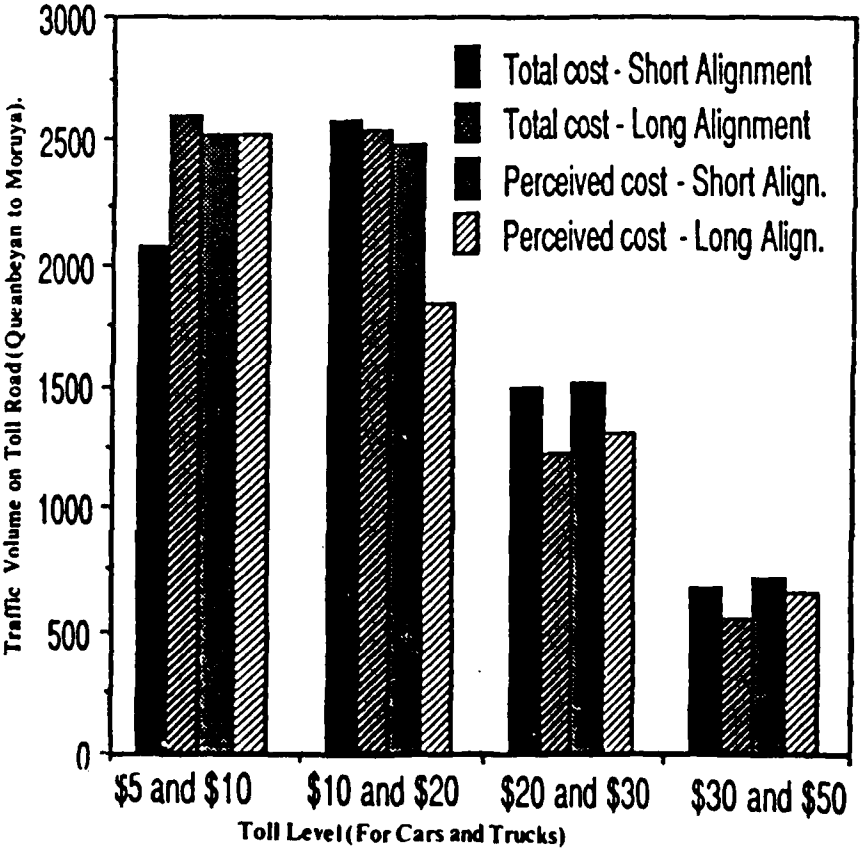


Fig 6. Traffic Estimates for Different Toll Levels - Corridor Between Queanbeyan and Moruya (South Coast)

However, trend extrapolation can be a misleading technique for long-term traffic forecasting, especially when there is substantial change in the land-use context. Therefore, the second stage of the analysis took a different approach. We asked the question: what level of traffic would be required to give a return on investment assuming construction costs and maintenance costs of the toll road were known? This is referred to as the break - even traffic analysis. Steps involved in this break - even analysis are shown in Fig 7.

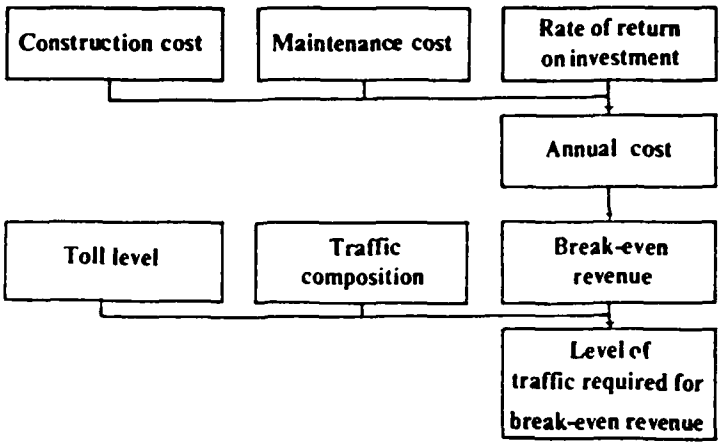


Fig 7. Conceptual Models for the Estimation of Break - Even Traffic Levels for Toll Road Investment.

The results are summarised in Fig. 8 . The graphs show the traffic volume required over a period of 30 years to make the proposal financially viable (8 percent return on investment) for three toll levels ($5 - car, $10 - trucks; $10 - cars, $ 20 - trucks; and $ 20 - cars, $30 - trucks). Assuming a toll level of $10 for cars, the traffic levels required initially would be in the order of 14000 vehicles per day.

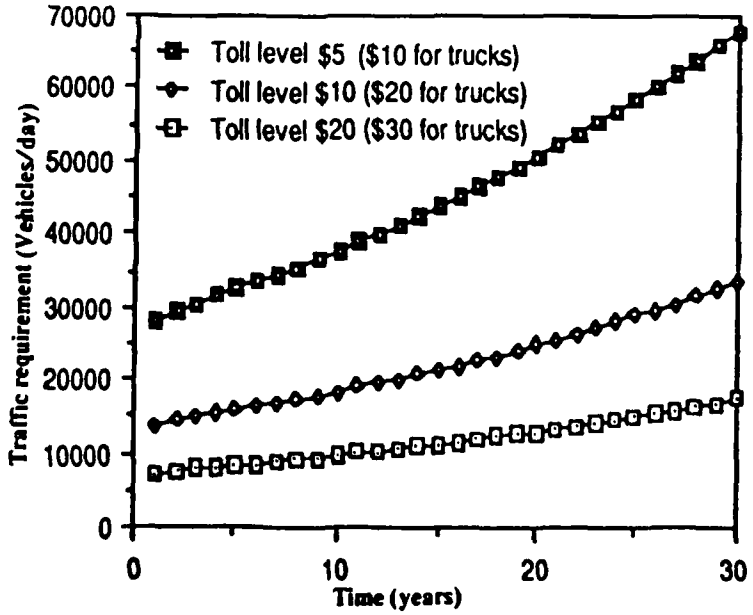


Fig 8. Average Annual Daily Traffic Over a 30-year Period Required to Financially Justify a Toll Road.

The significance of identifying the level of traffic required for a break - even revenue is to determine the difference between traffic forecasts based on historical trends, and the traffic required to justify, on economic grounds, investment in new roads. This traffic "shortfall" represents the amount of annual traffic that would have to be "generated" or "induced" by new land-use developments. In the context of the Queanbeyan - South Coast toll road proposal, these developments relate primarily to tourism and the attraction of the coast both for retired people (from Canberra) and for holiday home investment. Also considered was the relationship between the Very Fast Train (VFT) proposal linking Sydney, Canberra, and Melbourne (a fesibility study is due for completion in 1991) and its influence on tourist traffic, especially the role of the tollway as a "feeder service" to this new railway. the traffic results of these scenarios are beyond the scope of this paper, but nevertheless they formed an important part of traffic forecasting methodology that was developed for the client.

CONCLUSIONS

The methodology for economic evaluation and financial appraisal of transport facilities which aim to attract private-sector funding has been described using two case studies. The first case study, the $ 400 million Sydney Harbour Tunnel project, required the development of a methodology that took standard AADT traffic projections and separated them into weekday and weekend traffic and then into traffic by three periods of the day.. The traffic estimation procedure provided the necessary input to the application of Davidson's travel time/flow model to transform the traffic projections into level of service measures, such as travel time, fuel consumption, vehicle operating costs, and frequency of accidents. Standard economic evaluations (benefit cost ratio, internal rate of return, and net present value) were performed based on the predicted variations of the above measures under a number of alternative transport improvement strategies. A flexible micro - computer program allowed sensitivity analyses to be readily undertaken.

The second case study of a proposed toll road project in rural New South Wales was introduced to demonstrate the application of the logit model to account for the manner in which road users weight up the costs and benefits in choosing between alternative routes. The logit model was applied to estimate the potential traffic share on a toll facility, compared to the total traffic volume of the corridor, given assumptions about route lengths, speeds and toll charges. This particular case study was also used to demonstrate the break-even traffic analysis developed to estimate the annual level of traffic required over the project life-time to attract private investment.

This paper has explained the research and development underpinning major independent, expert consultancy advice to government and to the private sector by the University of New South Wales R & D Company, Unisearch Ltd. Although economic evaluation has become standard practice in road project appraisal there are nevertheless technical issues in the quantification of benefits (and also of environmental costs that have not been addressed in this paper). Quantification of

benefits requires accurate traffic estimation because the level of facility use will be an important factor in total user benefits and in the financial viability of any project financed on a user-pays principle. In Australia, as direct experience with tollways is limited, as outlined in Section 2, and our search through the Australian literature failed to discover any suitable methodology, it was necessary for the authors to develop original traffic models as one part of the overall economic and financial evaluation process, and these has been explained in technical detail in this paper. To date no methodological work on the private-sector toll road proposals in New South Wales has been published, and so this work is presented to stimulate discussion about the methodology to improve the accuracy of traffic modelling and forecasting exercises.

ACKNOWLEDGEMENTS

The administrative support of Unisearch Ltd (especially Ms Sandra Wood and Ms Pilar Helmers) is acknowledged in undertaking the research and development reported in this paper. The authors were assisted in their work by Messrs Villum Pipal and Lida Song, (Department of Transport Engineering) and Dr G.H. Kim (now Gaemyung University, South Korea). Dr. S.M. Sayed-Hosseini thanks the Iranian government and the Iran University of Science and Technology, Tehran, for support to undertake sabbatical leave at the University of New South Wales. The authors acknowledge the support from the clients who initiated this work, especially Messrs R.B. Symthe, M. Conroy and D.Roman and Ms C. de Marco (NSW Department of Environment and Planning). Mr F. Ciardi (SEPI, Australia) and Mr M. Podmore (Performancec Equity Limited). However, the views and interpretations in this paper are those of the authors.

REFERENCES

[1] Abelson, P. The Economic Evaluation of Roads in Australia, (Mosman, NSW: Australian Professional Publications), 1987.

[2] Ben - Akiva, M.E. and Lerman, S.R."Disaggregate Travel and Mobility Choice Models and Measures of Accessibility", in Behavioural Modelling by Hensher D.A.and Stopler P.R. (Eds) Croom Helm, London, 680-697, 1979.

[3] Black, J.A. and Blunden, W.R. "On Economic Evaluation of Private - Public Sector Transport Projects", 13th Australasian Transport Research Forum: Forum Papers 2,777-788 (1988).

[4] Bowyer, D.P., Akcelik, R. and Biggs, C.D. "Guide to Fuel Consumption Analysis for Urban Traffic Management", ARRB Internal Report, AIR 390-9 (Vermont, Victoria: Australian Road Research Board), 1984.

[5] Bowyer, F.P., R. Akcelik and C.D. Biggs "Guide to Fuel Consumption Analysis for Urban Traffic Management", AARB Special Report, No.32) (Vermont, Victoria: Australian Road Research Board), 1985.

[6] Cameron Mc Namara. Sydney Harbour Tunnel: Environmental Impact Statement prepard for Transfield - Kumagai Joint Venture by Cameron McNamara Consultants, November, 1986a.

[7] Cameron Mc Namara. "Sydney Harbour Tunnel: Economic Evaluation Working Paper", November, 1986 in Sydney Harbour Tunnel Working Papers 1986b.

[8] Clark, R.G. "The Behavioural Impact of the Sydney - Wollongong Tollway", unpublished

PhD Thesis, University of New England, 1977.

[9] Davidson, A.E. "The Gateway Arterial Road", queensland Roads, 23 No. 2, 21-24 (1987)

[10] Davidson, K.B. "A Flow Travel Time Relationship for Use in Transportation Planning", Proceedings of Australian Road Research Board, No. 1,3 183-194 (1966).

[11] Department of Main Roads, The Roadmakers: A History of Main Roads in New South Wales (Sydney: Department of Main Roads), 1976.

[12] Grigg, T. "Gateway Bridge Transport: Short Term Impacts of Industrial Activity", 12th Australian Transport Research Forum Papers, 2 583-604 (1987).

[13] Gutteridge, Haskins and Davey "Sydney Harbour Tunnel: Summary Report on key Traffic Issues", October, 1986 in Sydney Harbour Tunnel Working Papers, 1986.

[14] Loveday, P. and Morris, B. "Privatisation Opportunities for NSW Main Roads", 13th Australasian Transport Research Forum Papers, 2, 543-553 (1988).

[15] New South Wales, Department of Main Roads, "Turnpikes in Early New South Wales", Main Roads, No. 4, 16, 107-111 (1951).

[16] New South Wales, Department of Main Roads , Traffic Volumes and Supplementary Data 1986- South Coast Division, 1986a.

[17] New South Wales, Department of Main Roads, Traffic Volumes and Supplementary Data 1986 - Illawarra Division, 1986b.

[18] New South Wales, Road Freight Transport Industry Council, Economic and Industry Report No. 12, 1986.

[19] Symons, N.R., Standingford, J.A. and Jones, R.B., "The Sensitivity of Travel Demand to Toll Charges, Experience at West Gate Bridge", Proceedings 12th Australian Road Research Board Conference, Part 6, 12 154-159 (1986).

[20] Thompson, R.G. and Vincent, E.N., "Assessing the Traffic and Economic Impacts of the West Gate Bridge Toll Removal", Proceedings 14th Australian Road Research Board Conference, part 3, 14, 168-175 (1988).

[21] Unisearch Ltd, "The Proposed Sydney Hardour Tunnel: Some Comments on Vehicular Traffic Projections and Their Implications for Economic Evaluation", A report prepared on behalf of Unisearch Ltd by J A Blackand G H Kim for the Department of Environment and Planning, New South Wales (Kensington, NSW. Unisearch Ltd), February, 1987a.

[22] Unisearch Ltd, "Cross Harbour Transport Study - An Appraisal of the Sydney Harbour Tunnel and Some Alternatives - Phase I", A report prepared on behalf of Unisearch Ltd by J A Black for Save Our Sydney Group (Kensington, NSW: Unisearch Ltd), December, 1987b.

[23] Unisearch Ltd., "Traffic Estimation and Appraisal of a Proposal for a Queanbeyan - South Coast Toll Road", A report prepared on behalf of Unisearch Ltd by J A Black, S. M.Sayed-Hosseini, L. Song and U. Vandebona for SEPI Australia Pty Ltd, (Kensington, NSW: Unisearch Ltd), June, 1989.

[24] Wuestfeld, N. H. and Regan, E. J. "Impact of Rate Increases on Toll Facilities", Traffic Quarterly, No. 4, 35, 639-655 (1981).

OpTiX-II: A Decision Support System for the Solution of Nonlinear Optimization Problems on Parallel Computers

(to be presented at the Symposium on Applied Mathematical Programming and Modeling, Budapest, January 1993
Session 6: Parallel and Distributed Computing)

Harald Boden and Manfred Grauer
University of Siegen, Faculty of Economics
Information and Decision Sciences Institute
Hölderlinstr. 3, W-5900 Siegen, Germany
E-mail: hboden@fb5.uni-siegen.de

Abstract: The talk gives an introduction into the OpTiX-II DSS for the modelling and parallel solution of nonlinear optimization problems which arise especially in the fields of engineering design and production planning. The DSS supports all steps from the formulation of nonlinear optimization problems to the solution on parallel computers. Thereby OpTIX-II provides an engineer / decision maker with the knowledge of an optimization / computer expert in form of software. In order to reduce the overall computing time and to improve the quality of the solution obtained, much emphasis has been placed on decomposition principles and nonsequential solution approaches in mathematical optimization.

1. Introduction

OpTiX-II is an interactive decision support system for the solution of nonlinear optimization problems which arise especially in the fields of engineering design and production planning. The OpTiX-II software environment supports an engineer / decision maker with the knowledge of an optimization / computer expert in the form of software. It makes use of modern computer technology (e.g. computer networks, parallel computers) for faster and more reliable problem solutions and supplies an easy-to-use graphical interface for untrained computer users. In order to reduce the overall computing time and to improve the quality of the solutions obtained, much emphasis has been placed on decompositon principles and nonsequential solution approaches in mathematical optimization. Parallelism in the solution of nonlinear optimization problems can be exploited at several levels:

(I) First, the application of decomposition techniques and multi-level optimization strategies leads to 1st level optimization subproblems and a 2nd level coordination problem. These decomposed optimization problems are solved by primal decomposition methods (feasible method) or dual decomposition methods (non-feasible method). In both cases, the solution of the 1st level optimization subproblems is well suited for a *coarse grained* parallel computation. Therefore OpTiX-II may distribute these subsystem optimizations onto a network of heterogenous Unix-workstations or parallel MIMD-type computers.

(II) Secondly, parallel implementations of the classical algorithms based on the Lagrange- or Kuhn-Tucker theory, make use of *fine grained* parallelism. This leads to a high communication effort between the processors involved in the computation. Therefore, it is advisable to use strongly coupled parallel computers with a high communication bandwidth, e.g. shared memory multiprocessor-systems. Currently OpTiX-II supports some parallel implementations of classical algorithms on shared memory multiprocessor-systems using the Unix operating system.

(III) In their mathematical description, practical nonlinear optimization problems frequently consist of highly nonlinear objective functions and constraints. In these cases assumptions about unimodality, convexity, and smoothness of well-known solution methods in nonlinear optimization are mostly invalid. OpTiX-II allows to apply a simultaneous combination of different optimization algorithms to one optimization problem. Thereby, the controlled information exchange between the participating and parallel-running methods is the basis for a more reliable and, in some cases, even faster solution.

Work with OpTiX-II consists of three phases: During the problem formulation phase (I), the user defines the precise mathematical formulation of the optimization problem under analysis (see Section 2). This formulation is then translated (II) into a machine representation, which is suitable for parallel processing in heterogenous networks (see Section 3). The third phase, used for the solution of the optimization problem, is described in Section 4. Within this step (III), the user has to define a optimization strategy, by choosing a combination of optimization algorithms, and he has to start the optimization process.

2. Problem Formulation Phase

This phase is supported by the OpTiX-II Edit-Environment (fig. 1), which is being used for formulating the optimization problem, for controlling the generation of optimization servers for different platforms and for starting the execution environment.
The problem description is entered into a graphically controlled problem editor using the OpTiX-II problem description language which resembles the mathematical notation for nonlinear optimization problems (fig. 2). In many practical situations, complex optimization problems can not be described by simple mathematical notation. Therefore, OpTiX-II allows the inclusion of external functions written in the programming languages C or Fortran (fig. 1). Calls to these functions may

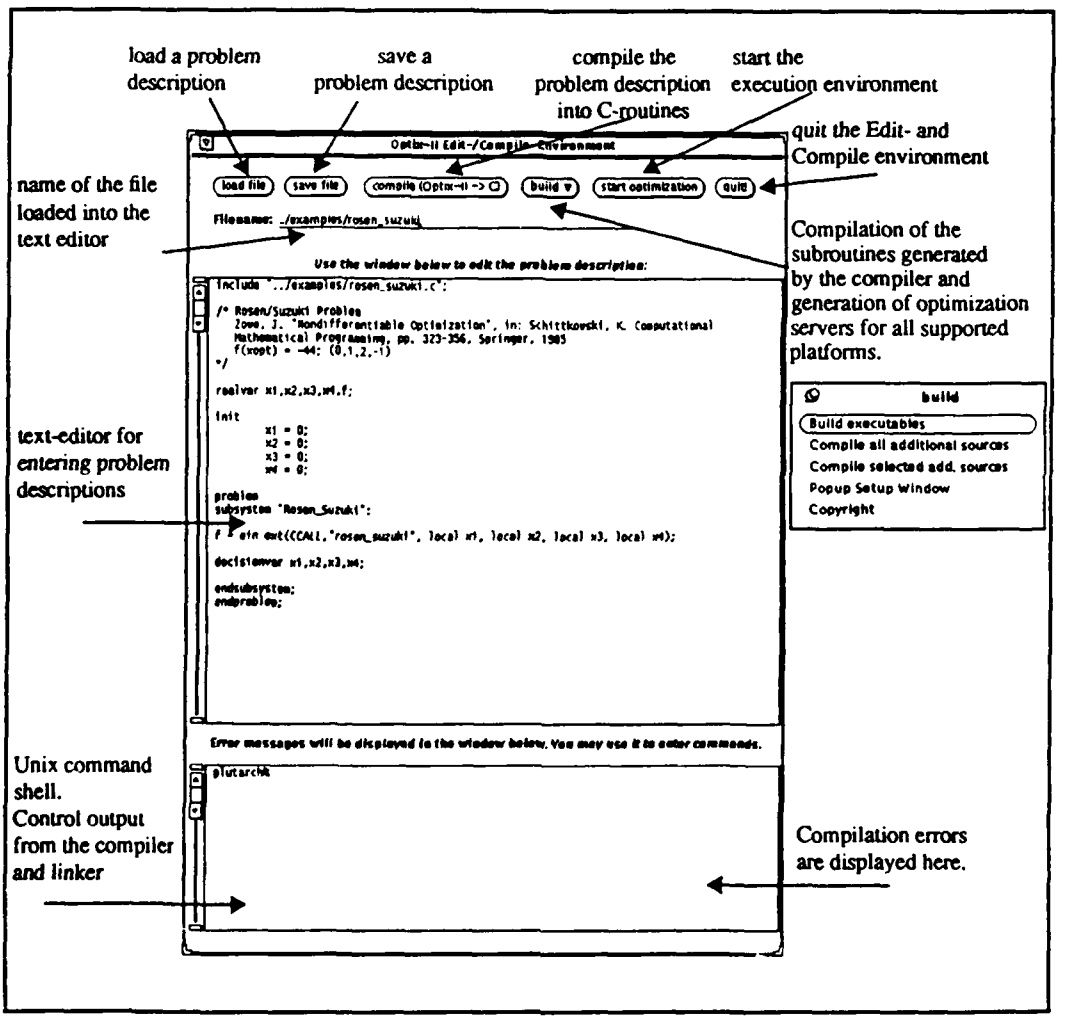


Fig. 1: The OpTiX-II Edit-Environment.

become a subexpression of an objective function or a constraint. Even commercially available simulation packages for the solution of complex mathematical models may be included in such a way. Other language constructs allow the formulation of decomposed optimization problems (fig. 2) so that multi-level optimization strategies may be applied.

3. Problem Translation Phase

This phase is controlled by the „Compile" and „Build executables" menus from the Edit-Environment. The selection of the „Compile" button starts the OpTiX-II problem compiler, which translates the problem description into a collection of functions written in the computer language C. Thereby, the compiler calculates symbolic first and second order derivatives for all objective functions and constraints. This ensemble of C-functions is then compiled and linked into optimization servers for different hardware/software platforms (e.g. SPARC/Solaris, MIPS/Ultrix, Transputer/Helios). These optimization servers are called from within the execution environment (in phase III). This approach allows parallel optimization in heterogeneous computer networks.

```
/* gear reducer decomposed */
realvar x1,x2,x3,x4,x5,x6,x7,f,f1,f2;

problem

subsystem "shaft_and_bearings_1" :
    f1 = min -1.508*x1*sqr(x6) + 7.477*x6^3 + 0.7854*x4*sqr(x6);
decisionvar x4,x6;
constraints
        /* g3 */ 1.93/x2/x3*x4^3/x6^4 <= 1;
        /* g5 */ sqrt(sqr(745*x4/x2/x3)+16.9E6)/0.1/x6^3 <= 1100;
        /* g24 */ (1.5*x6+1.9)/x4 <= 1;
bounds
        /* g16,g17 */ 7.3 <= x4 <= 8.3;
        /* g20,g21 */ 2.9 <= x6 <= 3.9;
endsubsystem;

subsystem "shaft_and_bearings_2" :
    f2 = min -1.508*x1*sqr(x7) + 7.477*x7^3
            + 0.7854*x5*x7^2;
decisionvar x5,x7;
constraints
        /* g4 */ 1.93/x2/x3*x5^3/x7^4 <= 1;
        /* g6 */ sqrt(sqr(745*x5/x2/x3)+157.5E6)
                    /0.1/x7^3 <= 850;
        /* g25 */ (1.1*x7+1.9)/x5 <= 1;
bounds
        /* g18,g19 */ 7.3 <= x5 <= 8.3;
        /* g22,g23 */ 5.0 <= x7 <= 5.5;
endsubsystem;

subsystem "Gear_Reducer_2nd_level":
f = min     -1.508*x1*sqr(x6) + 7.477*x6^3 + 0.7854*x4*sqr(x6)
            -1.508*x1*sqr(x7) + 7.477*x7^3 + 0.7854*x5*x7^2
            +0.7854*x1*sqr(x2)*(3.3333*sqr(x3)+14.9334*x3-43.0934);
decisionvar x1,x2,x3;
constraints
        /* g1 */ 27/x1/sqr(x2)/x3 <= 1;
        /* g2 */ 397.5/x1/sqr(x2)/sqr(x3) <= 1;
        /* g7 */ x2*x3 <= 40;
        /* g8 */ x1/x2 >= 5;
        /* g9 */ x1/x2 <= 12;
bounds
        /* g10,g11 */ 2.6 <= x1 <= 3.6;
        /* g12,g13 */ 0.7 <= x2 <= 0.8;
        /* g14,g15 */ 17 <= x3 <= 28;
endsubsystem;
endproblem;
```

Fig. 2: Example of an OpTiX-II problem description for a decomposed nonlinear optimization problem ([Azarm90], [Golinski70]).

Currently the problem translation phase generates code for:

(i) Unix workstations from different manufacturers.

(ii) MIMD computers with shared memory:
- multiprocessor Unix workstations/servers, e.g. Sun SPARCstation 10 and 600 series, Sun SPARCcenter 2000 series.

(iii) MIMD computers with distributed memory:
- Transputerclusters.
- Workstation networks, regarded as loosely coupled multiprocessor-system.

4. Problem Solution Phase

The OpTiX-II execution environment is used for controlling the ongoing optimization process. It distributes the computations onto NFS-based heterogeneous computer networks, multiprocessor workstations and transputer-based parallel computers. Furthermore, the Execution-Environment records the problem solution process and displays the results. The user interacts with the control module of the Execution-Environment (fig. 3). This user-interface corresponds to the Open Look standard and is completely interactive. In the simplest case of a non-decomposed optimization problem, the user selects an optimization algorithm from the algorithms list, a host for execution from the hosts panel, the „add" option from the „edit"-menu, and presses the start button. The optimization results are displayed in the control module window (fig. 3). After each computation, the user may select another algorithm and continue the optimization by pressing the continue button. For difficult or decomposed optimization problems, the user has to define a more complicated strategy script, defining the optimization steps that have to be taken. In each step, the user may combine the following strategies:

(i) In the case of decomposed optimization problems, all subsystem optimizations can be run in parallel, reducing the overall computational time effort. The user may choose this strategy by selecting a 3-tupel (subsystem, algorithm, host for computation) for each subsystem within the control module (fig. 3).

(ii) The user may apply a parallel optimization method (to a subsystem optimization), if a parallel computer with shared memory is available.

(iii) A simultaneous combination of different optimization algorithms to one optimization (sub)-system may be applied. In this situation, the controlled information exchange between the participating and parallel-running methods is the basis for a more reliable and, in some cases, even faster solution ([Boden91a], [Boden91b]). This approach is similar to hybrid optimization methods described in [Burdakov88] and [Kleinmichel92]. Their idea is to define tests for switching between a globally convergent method I and a locally superlinearly convergent method II in order to obtain a globally and locally superlinearly convergent method (fig. 4). The OpTiX-II user may apply both methods in parallel on different computing nodes. After a user-definable number of iterations, the best value of both strategies is selected and used as a basis for further computations.

The basic control unit in OpTiX-II is a block (fig. 5) that consists of a sequence of optimization steps, each using the strategies described in (i) to (iii). By the use of several blocks, in parallel, the calculations may be started from different initial points. Thereby, problems resulting from multimodality, nondifferentiability, and nonconvexity of the feasible domain may be overcome.

For the solution of decomposable nonlinear optimization problems the user may apply primal decomposition methods (resource division, principle of interaction prediction, feasible method) or dual decomposition methods (objective function modification, non-feasible method). In both cases, the user has to define first and second level (coordination) problems within the problem editor. He then defines a control strategy consisting of a block with two steps. In the first step, he selects at least one 2-tupel (algorithm, host for computation) for each first level subsystem. There-

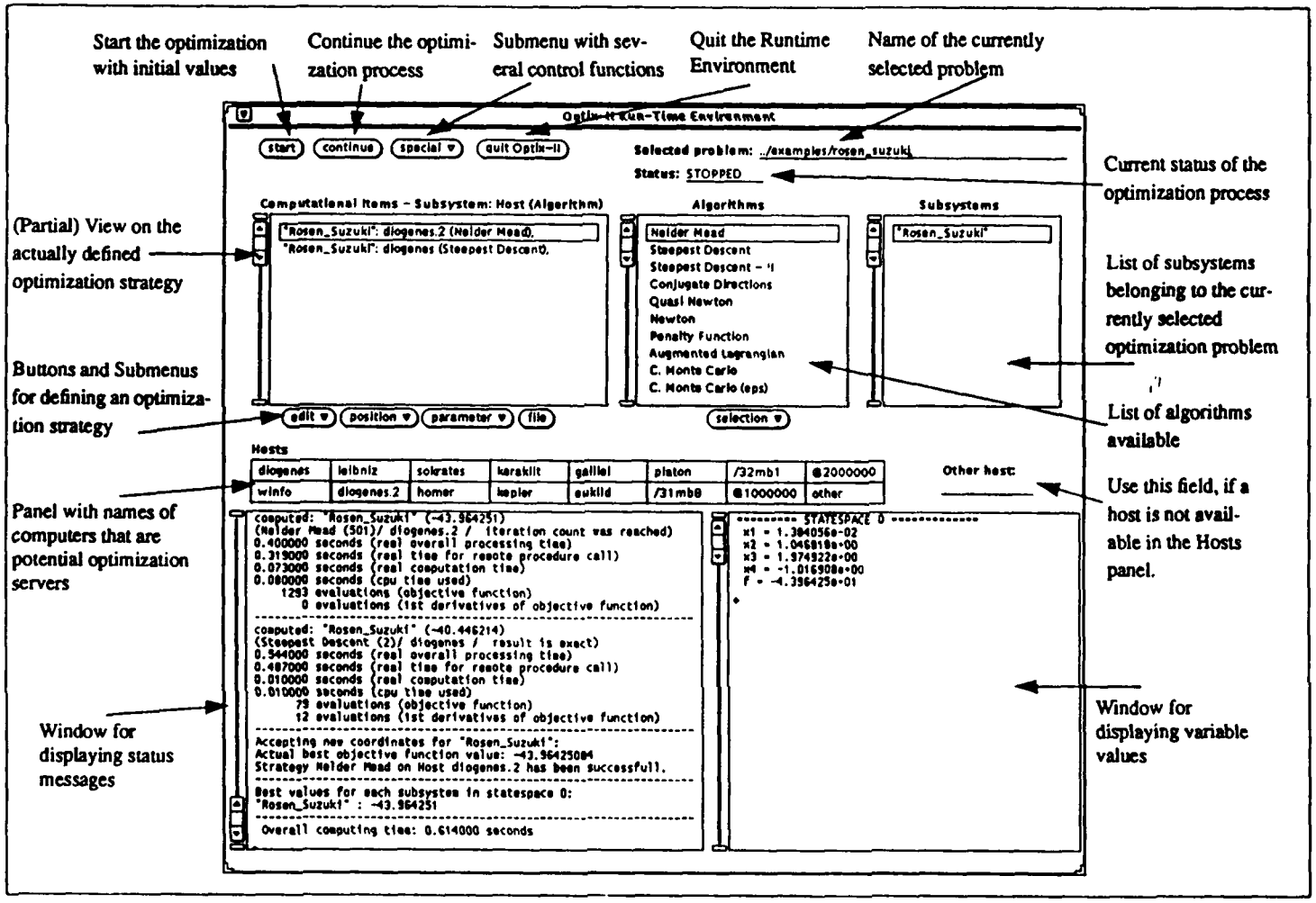


Fig. 3: The OpTiX-II Execution-Environment.

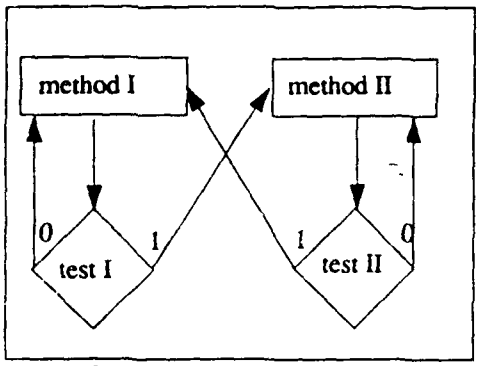


Fig. 4: The coupling-scheme within hybrid optimization methods [Kleinmichel92].

after, in the second step, he chooses at least one 2-tupel (algorithm, host for computation) for the second level (coordination) problem. This block is executed several times until convergence is obtained.

The user may set up experiments and adapt the program to an optimal sequence of (parallel running) algorithms for his type of nonlinear constrained optimization problem.

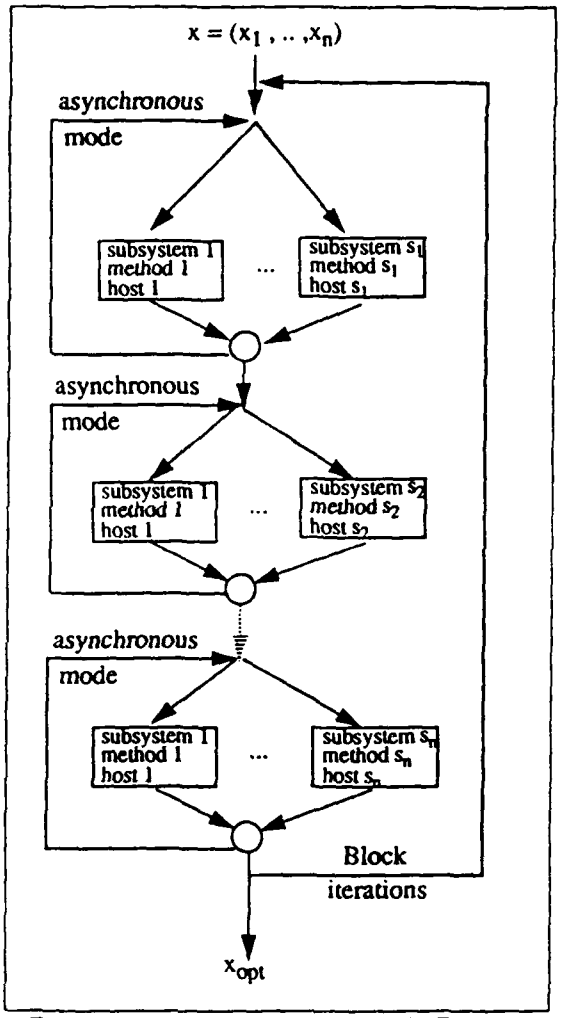


Fig. 5: Structure of a block in OpTiX-II.

5. References

[Azarm90] Azarm, S.; Li, W.C. : *Optimality and Constrained Derivatives in Two-Level Design Optimization*, in: Journal of Mechanical Design, Vol. 112, Dec. 1990.

[Bertsekas89] Bertsekas, D.P.; Tsitsiklis, J.N.: *Parallel and Distributed Computation*, Prentice Hall, 1989.

[Boden91a] Boden, H.; Gehne, R.; Grauer, M.: *Parallel Nonlinear Optimization on a Multiprocessor System with Distributed Memory*, pp. 65-78 in [Grauer91].

[Boden91b] Boden, H.; Grauer, M.: *Ein paralleler Lösungsansatz für nichtlineare Optimierungsprobleme*, p. 254-261 in Grebe, R.; Ziemann, C. (Eds.). Parallele Datenverarbeitung mit dem Transputer. Springer-Verlag, Berlin, 1991 (Informatik Fachberichte 272).

[Boden92] Boden, H.; Grauer, M.: *Funktional parallele Lösung von Dekompositionsproblemen der mathematischen Optimierung*, pp.139-146 in Grebe, R.; Baumann, M. (Eds.): Parallele Datenverarbeitung mit dem Transputer , Springer, Berlin, 1992 (Informatik aktuell).

[Burdakov88] Burdakov, O.; Richter, C.: *Parallel Hybrid Optimization Methods*, pp. 16-23 in Kurzhanski, A.; Neumann, K.; Pallaschke, D. (Eds.): Optimization, Parallel Processing and Applications. Springer-Verlag, Berlin, 1988.

[Dennis90] Dennis, J.E., Torczon, V.: *Direct Search Methods on Parallel Machines*, TR 90-19, Department of Mathem. Sc., Rice University, Houston, Texas, 1990.

[Golinski70] Golinski, J.: *Optimal Synthesis Problems Solved by Means of Nonlinear Programming and Random Methods*, pp. 287-309 in Journal of Mechanisms, Vol. 5, 1970.

[Grauer91] Grauer, M.; Pressmar, D.B. (Eds.): *Parallel Computing and Mathematical Optimization*. Springer-Verlag, Berlin, 1991.

[Grauer92] Grauer, M.; Boden, H.: *Opportunities on Parallel and Distributed Computation for Optimization and Decision Support*, pp. 197-207 in: Proceedings of the Tenth International Conference on Multiple Criteria Decision Making, Vol. 1, Taipei, July 19-24, 1992.

[Hoffmeister91] Hoffmeister, F.: Scalable Parallelism by Evolutionary Algorithms, pp. 175-196 in [Grauer91].

[Kleinmichel92] Kleinmichel, H.; Richter, C.; Schönefeld, K.: *On a Class of Hybrid Methods for Smooth Constrained Optimization*, pp. 465-499 in Journal of Optimization Theory and Applications: Vol. 73, No. 3, June 1992.

[Lootsma89] Lootsma, F.A.: *Parallel Nonlinear Optimization*, Delft University of Technoiogy, Report of the Fac. Math. and Informatics, No. 89-45.

Linear time algorithms for VLSI routing

Endre Boros (Rutgers Center for Operations Research, Rutgers University, New Brunswick, NJ, USA), András Recski (Dept. Math., Fac. El. Eng. and Informatics, Technical University of Budapest, Hungary) and Ferenc Wettl (Dept. Math., Fac. Transp. Eng., Technical University of Budapest, Hungary)

Extended abstract

Consider the gradually more and more complex problems of single row routing, channel routing and switchbox routing on the one hand; and the gradually less and less restrictive models (1-layer, Manhattan, unconstrained 2-layer, multilayer) on the other hand. The single row routing problem can always be solved in the Manhattan model, and the channel routing problem can always be solved in the unconstrained 2-layer model, in fact, both in linear time. We show that the switchbox routing problem is solvable, even in linear time, in the multilayer model.

I.

A switchbox is a rectangular grid G of horizontal tracks (numbered from 0 to w+1) and vertical columns (numbered from 0 to h+1), where w and h are the width and the height of the switchbox. The boundary points of G are called

- Northern if their coordinates are of form (i, w+1) with $i=1,2,\ldots,h$;
- Southern with form (i,0) where $i=1,2,\ldots,h$;
- Eastern with form (h+1,j) where $j=1,2,\ldots,w$; and
- Western with form (0,j) where $j=1,2,\ldots,w$.

For example, Figure 1 is a switchbox with width 4 and height 5, where Northern, Southern, Eastern and Western boundary points are denoted by x's, plus signs, empty and solid dots, respectively. The "corners" of G will not be considered as boundary points,

A net is a collection of boundary points. A switchbox routing problem (SRP) is a set of pairwise disjoint nets. If every boundary point of every net is Southern, the SRP is

called single row routing problem, if they are all Southern or Northern, the SRP is called channel routing problem (CRP). We shall also use the expression open box routing problem (OBRP) if there is no Eastern boundary point.

A CRP is called bipartite if every net consists of one Northern and one Southern boundary point. An SRP is called 4-partite if every net consists of one of each four types of boundary points. Finally, we call an OBRP 3-partite if every net consists of one Northern, one Southern and one Western boundary points (hence no Eastern boundary point is contained in any net).

The solution of a routing problem in the single layer model (SLM) is the realization of the nets as pairwise vertex disjoint subgraphs (usually Steiner trees) of the planar grid graph G so that each subgraph connects the boundary points of the net. The edge-disjoint single-layer model (EDM) is defined in the same way except that the subgraphs must be pairwise edge disjoint only. For example, Figure 2 shows the solution of an SRP in the SLM, while the two SRP's of Figure 3 cannot be solved in the SLM, only in the EDM.

The unconstrained k-layer model (UkM) requires pairwise vertex disjoint subgraphs of the k-layer grid graph G_k. Edges of these subgraphs joining adjacent points of two distinct layers are called vias. In case of k=2 vias are also called via holes but one should not imagine them as holes if k>3 since situations like that of Figure 4 are also possible (segmented or stacked vias, see Mueller and Mlynski, 1988 or Lengauer, 1990, respectively).

The multilayer models may be constrained. If we have two layers and one of them is restricted to horizontal wire segments and the other is restricted to vertical ones then we obtain the Manhattan model (MM). For example, both SRP's of Figure 3 can be solved in the U2M but only the second one can be solved in the MM.

Finally let us emphasize that the "corners" of the grid graph G (or any copy of them in G_k) must not be used in the routing. Similarly, the solution of a single row routing problem must not use Eastern, Western or Southern boundary points, that of a CRP must not use Eastern and Western ones, and that of an OBRP must not use Eastern ones.

II.

Every single row routing problem can be solved in the MM, in fact, in linear time. This observation is probably due to T. Gallai and it belongs to the engineering folklore since decades. Similarly, every CRP can be solved in the U2M (Marek-Sadowska and Kuh, 1983) and even a linear time algorithm is known (Recski-Strzyzewski, 1990). However, - while Gallai's algorithm realizes the problem with minimum width, our algorithm does not, and the computational complexity of deciding whether a CRP can be solved in the U2M with a given width seems to be open, see Johnson, 1984 and Recski, 1992 as well.

Hence a natural question arises: can we solve every OBRP or every SRP in the UkM with a sufficiently large k? The answer is negative, as shown by the SRP of Figure 5, essentially due to Hambrusch, 1985. If the pairs of identical numbers are the nets then the congestion of the dotted line is h+n. In case of ℓ layers this clearly means $\ell w \geq h+w$, leading to the lower bound $\frac{h}{w}+1$ for the number of layers. Thus the number of necessary layers can be arbitrarily high if we allow very thin or very wide rectangulars.

However, suppose that the quantity

$$m = \max\left(\frac{h}{w}, \frac{w}{h}\right)$$

is fixed (essentially, bounded from above). Let s denote the number of those sides of the board which contain terminals at all, i.e. let s=1 for the single row routing problem, s=2,3 and 4 for the CRP, OBRP and SRP respectively.

Theorem 1 There is a function $\ell_o = \ell_o(m,s)$ such that any problem characterized with m and s can be solved in linear time in the UℓM for $\ell \geq \ell_o$.

In particular, we conjecture $\ell_o(1,s)=s$ if s>1. Right now we can prove the following very special case:

Theorem 2 $\ell_o(1,s)=s$ in the s-partite case (s>1).

Details of the proofs and algorithms will be published in the full paper. Routing examples are shown in Figures 6 and 7 (m=1, s=2 and m=1, s=4, respectively). Wires in the four layers (in this order) are shown by heavy, thin, broken and dotted lines, respectively.

III.

Our main conjecture, $\ell_o(1,4)=4$ seems to be unrelated to

the result of Brady and Brown (1984), apart from its title "Four layers suffice" because there the authors show that a realization of the SRP in the EDM can always be transformed to the U4M but such an EDM solution need not exist. Since deciding, whether such a transformation from EDM to U3M is possible, is known to be NP-complete (Lipski, 1984), it is reasonable to conjecture that the problem to decide if an SRP with h=w can be realized in the U3M, is also NP-complete.

Acknowledgement Research funds of the Hungarian Academy of Sciences (Grant No. OTKA 2118 for A.Recski and OTKA 2505 for F. Wettl) are gratefully acknowledged.

References

Brady, M. L and D.J. Brown, 1984. VLSI routing: Four layers suffice, in F.P. Preparata, ed., Advances in Computing Research, Vol.2: VLSI Theory, 245-258. JAI Press,Greenwich,CT.

Hambrusch, S.E., 1985. Channel routing in overlap models, IEEE Trans. on Computer-Aided Design of Integrated Circuits and Systems, CAD-4, 1, 23-30.

Johnson, D.S., 1984. The NP-completeness column: an ongoing guide, Journal of Algorithms, 5, 147-160.

Lengauer, T., 1990. Combinatorial algorithms for integrated circuit layout, Teubner, Stuttgart - Wiley, Chicester.

Lipski, W., jr., 1984. On the structure of three-layer wireable layouts, in F.P. Preparata, ed., Advances in Computing Research, Vol.2: VLSI Theory, 231-244, JAI Press, Greenwich, CT.

Marek-Sadowska, M. and E. Kuh, 1983. General channel-routing algorithm, Proc. IEE (GB), 130, G, 3, 83-88.

Mueller, L. and D.A. Mlynski, 1988. Automatic multilayer routing for surface mounted technology. Proc. IEEE Internat. Symp. Circuits and Systems, Espoo, Finland, 2189-2192.

Recski, A., 1992. Minimax results and polynomial algorithms in VLSI routing, in M. Fiedler and J. Nešetřil, eds., Combinatorics, Graphs, Complexity, Prachatice, 261-273.

Recski, A. and F. Strzyzewski, 1990. Vertex-disjoint channel routing on two layers, in R. Kannan and W. Pulleyblank, eds., Integer Programming and Combinatorial Optimization, Waterloo, 397-405.

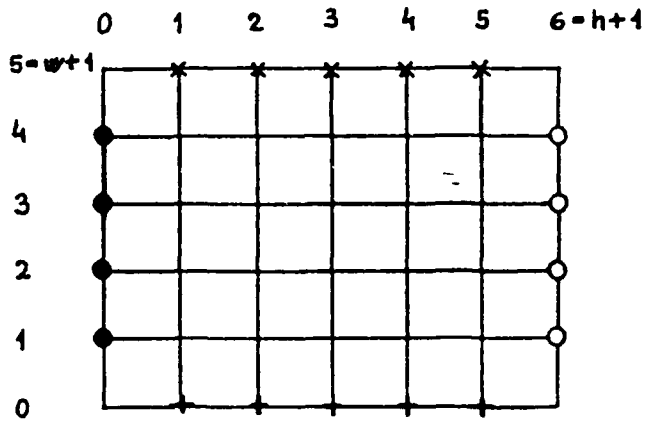


Figure 1

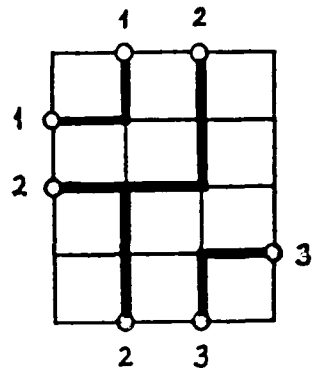


Figure 2

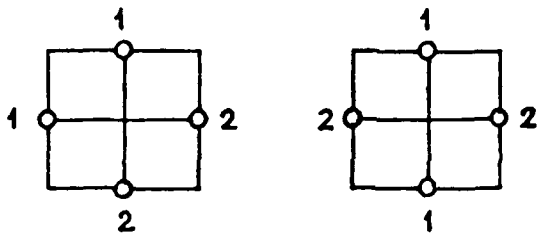


Figure 3

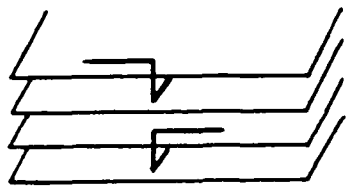


Figure 4

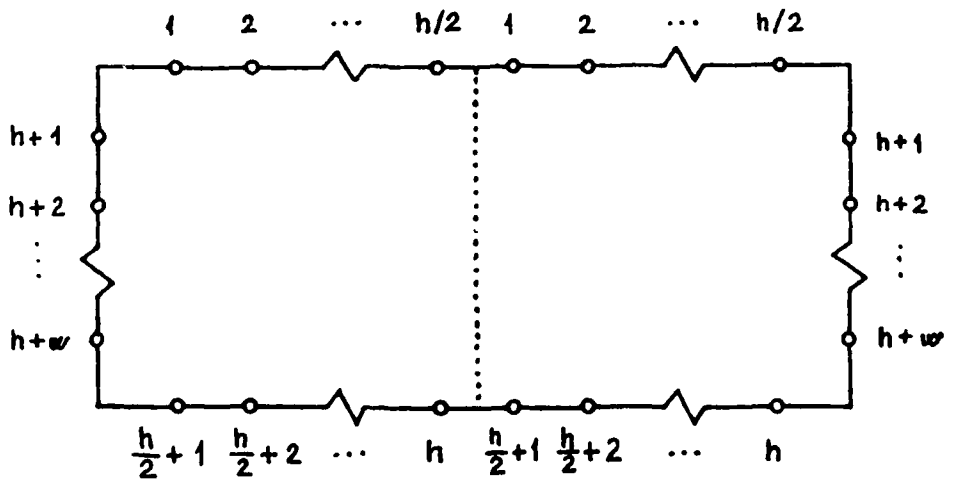


Figure 5

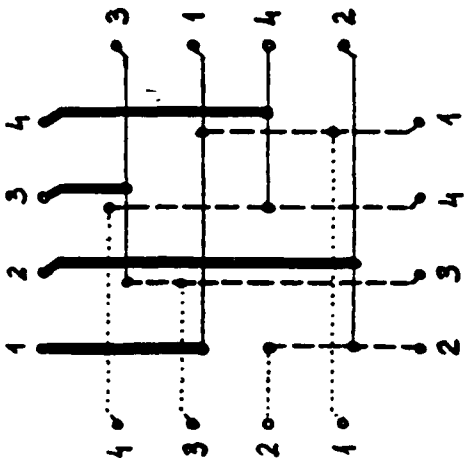


Figure 7

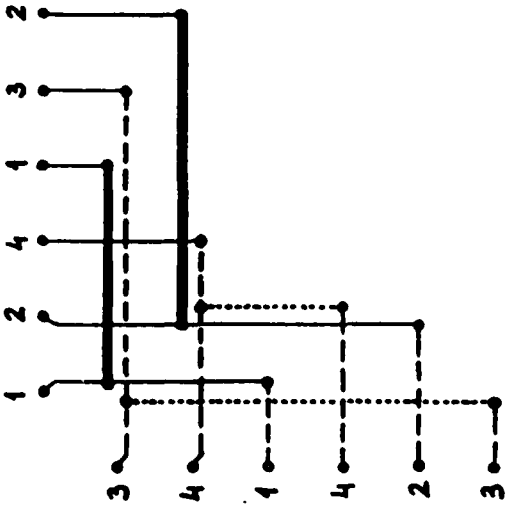


Figure 6

A Constructive Method to Improve Lower Bounds for the Quadratic Assignment Problem

Jaishankar Chakrapani* Jadranka Skorin-Kapov†

November 16, 1992

Abstract

We present a new approach to evaluating lower bounds for a class of quadratic assignment problems (QAP). An instance of a QAP of size n is specified by two $n \times n$ matrices D and F and we denote such an instance by QAP(D, F). Our approach is applicable to problems where the matrix D is derived as rectilinear distances between points on a regular grid. We construct two matrices F_{opt} and F_{res} such that $F = F_{opt} + F_{res}$ and the optimal solution to QAP(D, F_{opt}) is known. Any existing lower bound can then be applied to QAP(D, F_{res}), which in sum with the optimal value for QAP(D, F_{opt}) provides a valid lower bound to QAP(D, F). This approach results with improved lower bounds for some QAPs from the literature.

1. Introduction A quadratic assignment problem (QAP) of size n is specified by two $n \times n$ matrices D and F. Denoting by Π the set of all permutations of $\{1, 2, \ldots, n\}$, the problem can be defined as $\min_{\pi \in \Pi} C(\pi) = \sum_{i=1}^{n} \sum_{j=1}^{n} d_{\pi(i)\pi(j)} f_{ij}$. QAPs have numerous applications including facility location, backboard wiring and, scheduling. For a comprehensive survey of QAPs the reader is referred to a paper by Finke et al. [1]. In the context of facility location, the matrix D is thought of as the matrix of distances between locations, and the matrix F is thought of as the matrix of flow or interaction between facilities.

Two of the main existing lower bounds for the QAP are Gilmore-Lawlor bounds (GLB) [2, 4] and eigenvalue bounds [3, 8]. We propose a novel approach to the problem of computing lower bounds for QAPs. Our approach is applicable to QAPs whose matrix D is composed as rectilinear distances between points on a regular grid. All of Nugent's problems [6] of sizes between 5 and 30, one problem of size 36 due to Steinberg [11] and problems of sizes between 42 and 100 due to Skorin-Kapov [10, 9] fall under this category.

Our approach for calculating lower bounds for the QAP starts with an initial identification

*Department of Applied Mathematics and Statistics, State University of New York at Stony Brook, Stony Brook, NY 11794 email: chakrapa@ams.sunsyb.edu

†Harriman School for Management and Policy, State University of New York at Stony Brook, Stony Brook, NY 11794 email: jskorin@ccvm.sunysb.edu. All correspondences to be addressed to this author.

of F_{opt}^{init} and F_{res}^{init} such that $F = F_{opt}^{init} + F_{res}^{init}$, and the optimal solution to F_{opt}^{init} is known. Transformations preserving optimal permutation are then applied to F_{opt}^{init} to obtain F_{opt} and F_{res} such that $F = F_{opt} + F_{res}$. One of the transformations we use is similar to the one proposed by Palubetskes [7] to generate QAPs with rectilinear distance matrix and known optimal solution. For a QAP specified by matrices A and B, define by $opt(A, B)$ (resp., by $lb(A, B)$) the minimal objective function value (resp., the lower bound on the optimal objective function value). Clearly then, $opt(D, F) >= opt(D, F_{opt}) + opt(D, F_{res}) >= opt(D, F_{opt}) + lb(D, F_{res})$, and the last expression is a valid lower bound for the initial QAP. Any of the existing bounds from the literature can be applied to obtain $lb(D, F_{res})$, and therefore our method could serve as a preprocessing step to possibly tighten existing bounds. We formulate the construction of F_{opt} and F_{res} as a linear programming problem which we refer to in the sequel as $LPLB$. The sequel also states our results without proofs for the sake of brevity.

2. Constraints of $LPLB$ The first step to our bounding method is to generate a QAP instance with known optimal permutation. For convenience we use the identity permutation π_I as the optimal permutation.

QAP with π_I as optimal permutation: Consider a QAP where all the entries of one of the matrices, say F, equal a constant ξ. Denoting by $d_{sum} = \sum_{i=1}^{n} \sum_{j=1}^{n} d_{ij}$ the sum of all entries of the matrix D, it can be easily shown that for such a class of QAPs every permutation (including π_I) is optimal, and the objective function always evaluates to ξd_{sum}.

Transformations preserving optimal permutation: Let π_I be the optimal permutation to a QAP. We present two types of transformations that preserve the optimal permutation when applied to the flow matrix. Both transformations are elementary and the first is due to Palubetskes [7].

Let the rectilinear distance matrix D be formed from an $r \times c$ ($n = rc$) grid of points. Each location is then specified by its coordinates in the grid. Let $i = (r_i, c_i)$ and $j = (r_j, c_j)$ be two locations $1 \leq r_i, r_j \leq r$, $1 \leq c_i, c_j \leq c$. Let $k = (r_k, c_k)$ be another location. We say that k is in the path between i and j if r_k lies between r_i and r_j, and c_k lies between c_i and c_j. Note that if k is in the path between i and j, $d_{ij} = d_{ik} + d_{jk}$. The main result of Palubetskes [7] uses this property of rectilinear distances along with the validity of triangle inequality.

Definition 1 *Let α be a positive scalar. $\forall\ m \geq 1$, $\Delta^1(ik_1k_2\ldots k_mj,\alpha)$ is an $n \times n$ matrix such that the entries ik_1, k_1k_2, k_2k_3, $\ldots$, k_mj equal α, the entry ij equals $-\alpha$, and the rest of the entries are 0.*

Definition 2 *Let α be a positive scalar. $\Delta^2(ikj,\alpha)$ is an $n \times n$ matrix such that the entries ik, kj and ij equal α, and the rest of the entries are 0.*

Lemma 1 (Palubetskes) *Let D be the rectilinear distance matrix and let $\mathcal{F}$ be the flow matrix for which π_I is optimal. Let $i, k_1, k_2, \ldots, k_m, j$ be $m+2$ locations ($m \geq 1$) such that k_1 is in the path between i and j, and $\forall\ 2 \leq l \leq m$ k_l is in the path between k_{l-1} and j. Define $\mathcal{F}^1_{ik_1\ldots k_mj,\alpha} = \mathcal{F} + \Delta^1(ik_1\ldots k_mj,\alpha)$. π_I is still optimal for $\mathcal{F}^1_{ik_1\ldots k_mj,\alpha}$ with the optimal objective function value unchanged.*

Lemma 2 *Let D be the rectilinear distance matrix and let $\mathcal{F}$ be the flow matrix for which π_I is optimal. Let i, k, j be three locations such that $d_{ik} = d_{kj} = 1$ ($d_{ij} = 2$). Define $\mathcal{F}^2_{ikj,\alpha} = \mathcal{F} + \Delta^2_{ikj,\alpha}$ Then π_I is still optimal for $\mathcal{F}^2_{ikj,\alpha}$ with an optimal objective function value of $opt(D,\mathcal{F}) + 4\alpha$.*

Formally we define the transformations as below.

Definition 3 *$T1(ik_1\ldots k_mj,\alpha)$ is the transformation due to the addition of $\Delta^1(ik_1\ldots k_mj,\alpha)$ to the flow matrix, where the locations $i, k_1, \ldots, k_m, j$ satisfy the path criterion of Lemma 1.*

Definition 4 *$T2(ikj,\alpha)$ is the transformation due to the addition of $\Delta^2(ikj,\alpha)$ to the flow matrix, where the locations i, k, j are such that $d_{ik} = d_{kj} = 1$.*

Since the rectilinear distance matrix is symmetric, the flow matrix can be assumed to be symmetric without loss of generality. The resultant flow matrix after either of the transformations can be kept symmetric by performing both $T1(ik_1\ldots k_mj,\alpha)$ and $T1(jk_m\ldots k_1i,\alpha)$, or $T2(ikj,\alpha)$ and $T2(jki,\alpha)$ In the first case the optimal objective value is unchanged, and in the second case it is $opt(D,\mathcal{F}) + 8\alpha$.

Constraints: We first present constraints for the first type of transformations. Formulation of constraints involves identifying for each location i, the sets $M^i = \{j \mid j > i$ and j is involved in $T1(ik\ldots j)$ for some $k\}$ and $A^i = \{k \mid r_k \geq r_i$ and k is involved in $T1(ik\ldots j)$ for some $j\}$ such that $M^i \bigcap A^i = \emptyset$. Then for each location i, a single source single sink network flow graph is developed as follows. Node i is the source and each element in the set M^i has a directed edge to a common sink z^i. There are directed edges from i to each element k in the set A^i. For each $k \in A^i$ directed edges are added from k to each element

in the set A^k as follows. If $c_k < c_i$, add directed edges from k to elements in A^k whose columns are less than or equal to c_k; if $c_k > c_i$, add directed edges to elements in A_k whose columns are greater than or equal to c_k; and if $c_k = c_i$, add edges to all elements in A_k. This process is continued recursively for each element $k \in A^i$ and terminates upon reaching either the bottom left node $(r, 0)$, or the bottom right node (r, c), or both depending on whether c_k is less than, greater than, or equal to c_i respectively. Similarly, graphs are constructed for each location i. The result is a set of $n-1$ graphs, one for each location (except the n-th) as the source, and there is no edge between graphs corresponding to different locations. Consider the intermediate nodes which are neither the source nor the sink. For these nodes the balance constraints (flow-in equals flow-out) form a set of linear constraints. A flow from a source to a sink corresponds to a transformation as follows.

Definition 5 *Let i be the source and let $j \in M^i$ be one of the nodes with an edge to the sink z^i. Consider a positive flow of α along the path $i - k_1 - \ldots - k_m - j - z^i$. The transformation corresponding to the flow is $T1(ik_1 \ldots k_m j, \alpha)$*

We establish that the performance of the transformations must satisfy the constraints.

Theorem 1 *Let S_1 be the set of all feasible positive flows satisfying balance constraints, and let $\mathcal{T}^1$ be the set of all possible transformations of the first type with the sets A^i and M^i defined as above. Every $s_1 \in S_1$ corresponds to some subset of $\mathcal{T}^1$ and every subset of $\mathcal{T}^1$ corresponds to some $s_1 \in S_1$.*

A set of constraints for transformations of the second type can be realized in a similar fashion.

3. Objective Function of $LPLB$

We design (heuristically) a linear objective function to, possibly, tighten the Gilmore-Lawlor bounds (GLB). First, the rows and columns of the matrix F are permuted so that π_I achieves the best known objective function value for the QAP. Intuitively, this strategy should bring F_{opt} "closer" to F providing better bounds.

We start with an initial optimal part F_{opt}^{init} where all entries equal a constant $\xi > 0$ except the diagonal which are zeroes. The residual part F_{res}^{init} is defined so that $F = F_{opt}^{init} + F_{res}^{init}$. For the first type of transformations, we set $M^i = \{j \mid j > i \text{ and } [F_{res}^{init}]_{ij} < 0\}$ and $A^i = \{k \mid r_k \geq r_i \text{ and } [F_{res}^{init}]_{ik} > 0\}$. In other words, the transformation $T1$ adds (subtracts) to an entry of F_{opt}^{init} if the corresponding entry in F_{res}^{init} is greater (lesser) than zero.

Similarly, the sets are defined for $T2$. Note that $T2$ only adds to entries of F_{opt}^{init}. Denote by X_{ij}, the total amount added to or subtracted from $[F_{opt}^{init}]_{ij}$ due to all the transformations. We impose some additional constraints on X_{ij} as follows. $X_{ij} \leq -[F_{res}^{init}]_{ij}$ if $j \in M^i$, and $X_{ij} \leq [F_{res}^{init}]_{ij}$ otherwise. These constraints ensure that the transformations do not produce additional negative entries in the residual part of the flow matrix F.

Recall that π_I (the optimal permutation for F_{opt}) also achieves the best known objective function value for F. If π_I can also be established as an optimal permutation for F_{res}, its optimality for F is proven. Though this may not be possible in all cases, better bounds may be obtained in general if $lb(D, F_{res})$ is close to the value it achieves with π_I. Let $d_{max} = r + c - 2$ denote the maximum entry in the distance matrix, and let $d_{min} = 1$ denote the minimum entry. If π_I were optimal for F_{res}^{init}, in the evaluation of the objective function each entry $[F_{res}^{init}]_{ij}$ would be multiplied by d_{ij}. For the unknown optimal permutation let it be multiplied by some other distance matrix entry d_{pq}. The difference in objective function value due to a single entry ij is $[F_{res}^{init}]_{ij}(d_{ij} - d_{pq})$. If $[F_{res}^{init}]_{ij} > 0$, $(d_{ij} - d_{min})X_{ij}$ is the maximum gain in lower bound due to the entry ij. Similarly if $[F_{res}^{init}]_{ij} < 0$, $(d_{max} - d_{ij})X_{ij}$ is the maximum gain due to entry ij. Our objective function is to maximize $\sum_{i=1}^{n}\sum_{j=1}^{n} C_{ij}X_{ij}$ with the coefficients C_{ij} being either $d_{ij} - d_{min}$ or $d_{max} - d_{ij}$, depending on whether $[F_{res}^{init}]_{ij}$ is greater than or less than zero.

4. Computational Results

Computation of a lower bound involves three phases. In the first phase, an LP is generated depending on the initial optimal part and in the second phase, the LP is solved. In the third phase, the optimal part is constructed from the LP solution, its objective function evaluated, and a lower bound from the literature is applied to the residual part. The first and third phases of the computation were done on a Sun SPARC Station 1, and for the second phase the IBM 3090 version of LINDO was used. Recall that prior to generating the LP, the rows and columns of F are permuted (based on the best known heuristic solution) so that π_I, now, is the best known solution.

The constructive bounding method was tested on a number of problems from the literature viz., Nugent's problems [6] of sizes 5-30, one problem of size 36 due to Steinberg [11] and problems of size 42 and 49 due to Skorin-Kapov [10]. The initial optimal part was constructed by choosing constant entries ξ such that $\xi = bkv(D, F)/d_{sum}$, where $bkv(D, F)$ denotes the best known value for the QAP. Note that for this optimal part the optimal

objective value equals the best known value. The LP corresponding to this optimal part is generated and solved using LINDO. GLB is then applied to F_{res}. We present the results in Table 1. In the table BKV refers to the best known value for the QAP (which is optimal for problems up to Nug15), and CGLB refers to the lower bound for QAP(D, F) obtained by using GLB to obtain the lower bound for QAP(D, F_{res}).

Among the existing lower bounds in the literature, GLB provides the best bounds for Nugent's problems of size up to 8. However, for larger problems GLB does not perform as well as eigenvalue based bounds. We consider two eigenvalue based bounds from the literature: MEVB developed by Rendl and Wolkowicz [8], and IVB developed by Hadley et.al. [3]. MEVB provides better results than IVB for Nugent's problems of size up to 30. However, we do not have resuls from MEVB on problems of size greater than 30.

For problems of size greater than 15, we also performed another set of experiments by varying the starting entry for the optimal part. Recall from section 4 that if the starting entry ξ is 0, $M^i = \emptyset$ and no transformations of the first type are possible. As ξ increases, M^is]grow in size and A^is shrink until $A^i = \emptyset$ when ξ equals the maximum element in the matrix F. We tried a few values of ξ using IVB to compute the bounds for the residual part. The results are presented in Table 2.

From the tables it can be seen that when our constructive method is used as a preprocessing step, the bounds obtained (CGLB) are better than GLB for all the problems tested. Even for larger problems, where the eigenvalue bounds seem to provide better results, construction improves the bounds. Table 2 shows in bold the best bounds (CIVB) obtained by constructing F_{opt} and evaluating IVB(D, F_{res}). CIVB obtains better results than both MEVB, where applicable, and IVB for all the test problems.

Eigenvalue bounds seem to improve if the spectral radii of the matrices D and F are small [8]. Though there is no closed form equation to evaluate the spectral radius $sp(A)$ of a matrix A, it obeys the following inequality due to Mirsky [5] $sp(A) \leq m(A) = [2\sum_i \sum_j a_{ij}^2 - \frac{2}{n}(tr\ A)^2]^{\frac{1}{2}}$, where trA denotes the trace of the matrix A. Performing transformations to minimize $m(F_{res})$ can be posed directly as a quadratic programming problem with a convex objective function. Table 2 provides results with a linear objective function, and we suspect that a quadratic objective might improve results even further.

Table 1: Constructive GLB

Problem	BKV	MEVB(D,F)	GLB(D,F)	OPT(D,F_{opt})	GLB(D,F_{res})	CGLB(D,F)
Nug6	86	70	82	92.161	-9.604	84
Nug8	214	174	186	248.115	-42.112	206
Nug12	578	495	493	618.792	-92.682	528
Nug15	1150	989	963	1191.525	-149.254	1044
Nug20	2570	2229	2057	2687.272	-466.954	2222
Nug30	6124	5349	4539	6194.461	-974.015	5222
Ste36	9526	NA	7124	10795.224	-3316.368	7480
Sko42	15812	NA	11311	15836.092	-2698.571	13138

Table 2: Constructive IVB

Problem	ξ	MEVB(D,F)	IVB(D,F)	OPT(D,F_{opt})	IVB(D,F_{res})	CIVB(D,F)
Nug20	3	2229	2196	3480	-1275.052	2206
	4			4562	-2314.134	**2248**
	5			5700	-3457.714	2244
Nug30	3	5349	5265	9595	-4244.326	5352
	4			12778	-7402.730	**5376**
	5			15998	-10631.381	5368
Sko42	5	NA	13830	37350	-23241.681	14110
	6			44772	-30654.461	**14118**
	7			52234	-38152.217	14082
Sko49	6	NA	20716	65858	-45018.698	20840
	7			76838	-55974.705	**20864**
	8			87808	-67007.763	20802

5. Conclusions

We have proposed a new construction based approach to obtaining lower bounds for the QAP. Our approach is based on performing optimality preserving transformations to decompose the QAP into two problems: one for which an optimal solution is known, and another to which any existing lower bound can be applied. This provides a lower bound to the original QAP. Among existing lower bounds we considered GLB and IVB in our study.

We provide a set of linear constraints to perform the transformations. A linear programming problem $LPLB$ is formulated and solved to complete the construction. We have improved both GLB and IVB for all the problems tested. We conjecture that IVB may be improved directly by formulating a quadratic objective function, and solving the resulting optimization problem.

Though our method is developed for QAPs with rectilinear distance matrix, it can be extended to QAPs with distance matrices satisfying triangle inequality. Since our method constructs F_{opt} with known optimal solution such that $F = F_{opt} + F_{res}$, it has applications to sensitivity analysis and may also be useful in branch and bound techniques.

Acknowledgements We thank Franz Rendl for providing us the code to compute IVB. This work was supported in part by NSF grant DDM-8909206.

References

[1] G. Finke, R.E. Burkard, and F.Rendl. Quadratic assignment problems. *Annals of Discrete Mathematics*, 31:61–82, 1987.

[2] P. Gilmore. Optimal and suboptimal algorithms for the quadratic assignment problem. *Journal of SIAM*, 10:305–313, 1962.

[3] S.W. Hadley, F. Rendl, and H. Wolkowicz. A new lower bound via elimination for the quadratic assignment problem. Technical Report CORR 89-5, University of Waterloo, 1989.

[4] E. Lawlor. The quadratic assignment problem. *Management Science*, 9:586–599, 1963.

[5] L. Mirsky. The spread of a matrix. *Mathematica*, 3:127–130, 1956.

[6] C.E. Nugent, T.E. Vollmann, and J. Ruml. An experimental comparison of techniques for the assignment of facilities to locations. *Operations Research*, 16:150–173, 1968.

[7] G.S. Palubetskes. Generation of quadratic assignment test problems with known optimal solutions (in russian). *Zh. Vychisl. Mat. Mat. Fiz.*, 28(11):1740–1743, 1988.

[8] F. Rendl and H. Wolkowicz. Applications of parametric programming and eigenvalue maximization to the quadratic asignment problem. *Mathematical Programming*, 53(1):63–78, 1992.

[9] J. Skorin-Kapov. Extensions of a tabu search adaptation to the quadratic assignment problem. To appear in *Journal of Computers and Operations Research.*

[10] J. Skorin-Kapov. Tabu search applied to the quadratic assignment problem. *ORSA Journal on Computing*, 2(1):33–45, 1990.

[11] L. Steinberg. The backboard wiring problem: A placement algorithm. *SIAM Review*, 3:37–50, 1961.

FINDING MINIMAL INFEASIBLE SETS OF CONSTRAINTS IN INFEASIBLE MATHEMATICAL PROGRAMS
(extended abstract)

John W. Chinneck
Systems and Computer Engineering
Carleton University
Ottawa, Ontario K1S 5B6
CANADA

telephone: (613) 788-5733
email: chinneck@sce.carleton.ca

The major difficulty in mathematical programming is no longer the solution of large models, it is the correct formulation (or re-formulation) of the model. Large models are now solved routinely, but their very size complicates the determination of how to make repairs when the model is infeasible or otherwise nonfunctional. One useful approach is to localize or isolate the problem to a smaller portion of the whole model. This paper presents methods and case studies in the analysis of infeasible mathematical programs by isolating an Irreducibly Inconsistent Set (IIS) of constraints.

An IIS is a set of constraints which is infeasible, but which becomes feasible if any one member is removed. The IIS may consist of only a few constraints when the total constraint set is very large. The diagnosis of the problem in human-understandable terms often follows directly from examination of the IIS. At worst,

other algorithms or expert systems or humans need only operate on the IIS, typically a much reduced portion of the entire model, to arrive at a final diagnosis. This improves the overall efficiency of the diagnosis and repair process.

The paper presents the basic algorithms for IIS isolation for both linear and nonlinear programs, and their implementation in a modified version of MINOS 5.3, known as MINOS(IIS), developed at Carleton University. The algorithms are effective and quick in the linear case. The time to find the IIS is often a small fraction of the time to make the initial determination that the LP is infeasible.

A specialized procedure for networks, which incorporates the concept of nonviability analysis, is also presented. Nonviability is a structural property of a network which a priori forces some of the arc flows to zero, before the addition of flow bounds or extra side constraints. An ordered set of tests of an infeasible network, including nonviability and IIS analysis, provides an improved diagnosis. Unlike flow-balancing methods, the specialized procedure is applicable to advanced netforms such as processing networks.

The analysis of infeasible nonlinear programs is complicated by the inability of nonlinear optimizers to determine the feasibility of a nonlinear constraint set with 100% accuracy. However, useful information can still be extracted which can help in selecting a new initial point for the optimizer.

Case studies of analyses of infeasible linear programs, networks, and nonlinear programs are presented.

THE POLYTOPE OF BLOCK DIAGONAL MATRICES

by **Yves Crama** [1] and **Maarten Oosten** [2]
[1] Dept. of Quantitative Economics and [2] Dept. of Mathematics
University of Limburg, P.O. Box 616
6200 MD Maastricht, The Netherlands

1. INTRODUCTION

In this paper, we investigate the facial structure of the polytope whose extreme points are exactly the m×p 0-1 block diagonal matrices (m, p ∈ N). More precisely, we define a matrix X to be *block diagonal* if there exists a partition $R_1, \ldots, R_k, R_{k+1}$ of its row-set and a partition $C_1, \ldots, C_k, C_{k+1}$ of its column-set such that $x_{ij} \neq 0$ if and only if $i \in R_l$ and $j \in C_l$ for some $1 \leq l \leq k$ (notice that what we really mean is that X is block diagonal up to permutations of its rows and columns). We let

$$S_{mp} = \{ X \in \{0,1\}^{mp} \mid X \text{ is block diagonal } \},$$

and we denote by Q_{mp} the convex hull of S_{mp}. The goal of this paper is to provide a (partial) description of the polytope Q_{mp} by linear inequalities.

As explained in Crama and Oosten (1992), our interest for the polytope Q_{mp} mainly stems from its relation to the cell formation problem encountered in cellular manufacturing. The data for this problem are generally assumed to be summarized in the *machine-part incidence matrix* A, where $a_{ij} = 1$ if part j needs to be processed on machine i, and $a_{ij} = 0$ otherwise. Recall that a group technology cell consists of a number of machines (a machine-group) geared on the manufacturing of a number of similar parts (a part-family). The cell formation problem asks for a partition of the machines into machine-groups, a partition of the parts into part-families, and a matching between the machine-groups and the part-families which optimizes some measure of the inter- and intra-cell relationships. It can be abstracted into the following *block diagonalization problem*: given an m×p incidence matrix A and a function f(.,.), find an m×p block diagonal incidence matrix X which minimizes f(A,X) (the function f(.,.) gives an estimate of the distance, or dissimilarity, between the original incidence matrix A and the 'ideal' cellularized system represented by X). In Crama and Oosten (1992), we showed that, for many of the objective functions f(.,.) proposed in the literature, the cell formation problem can be reduced to the problem of minimizing a linear function of the variables x_{ij} (i = 1, ..., m; j = 1, ... p) over the polytope Q_{mp}. Similar block diagonalization problems also arise in the analysis of large data arrays (e.g. for marketing or archeology applications), in production planning for flexible manufacturing systems, in sparse matrix computations, etc (see Crama and Oosten 1992 for references).

In our presentation, we will often rely on a graph-theoretic interpretation of block diagonal matrices and of the polytope Q_{mp}. We follow the graph-theoretic terminology of Bondy and Murty (1976). Moreover, when B = (U,V,E) is a bipartite graph and G = (U,V,F) is a subgraph of B, we say that G is a *complete bipartite partitioning* of B if all connected components of G are complete bipartite (we look at isolated vertices as complete bipartite graphs). In particular, consider the complete bipartite graph $K_{mp} = (U_m, V_p, U_m \times V_p)$, where $U_m = \{u_1,\ldots,u_m\}$ and $V_p = \{v_1,\ldots,v_p\}$. We regard an arbitrary m×p 0-1

matrix X as the adjacency matrix of a subgraph G of K_{mp}, say $G = (U_m, V_p, F)$, where $(u_i, v_j) \in F$ if and only if $x_{ij} = 1$. It is easy to see that the matrix X is block diagonal if and only if its associated graph is a complete bipartite partitioning of K_{mp}.

This graph-theoretic interpretation stresses the analogy of the polytope Q_{mp} with the clique partitioning polytope P_n investigated by Faigle, Schrader and Suletzki (1986) and Grötschel and Wakabayashi (1989), and with the related multiway cut polytope studied by Chopra and Rao (1991). In fact, Q_{mp} can be viewed as the projection of P_{m+p} on some appropriate subspace. But this observation does not seem very useful in deriving a description of Q_{mp} from the results available about P_n.

In Section 2, some general properties of facet-defining inequalities for the polytope Q_{mp} are stated. In Section 3 specific families of facet-inducing inequalities are described. Section 4 contains some lifting theorems. Finally, in Section 5, a technique is presented to patch facet-defining inequalities into new valid inequalities which, under certain conditions, also define facets.

2. PROPERTIES OF FACET-DEFINING INEQUALITIES

We describe in this section some general properties of facet-defining inequalities for the polytope Q_{mp} : two 'lifting' results, relating facets of lower-dimensional polytopes to facets of higher-dimensional ones, and one proposition describing the 'graphical' structure of facet-defining inequalities.

In our discussion, it will be often convenient to consider the polytope associated with block diagonal submatrices of a given matrix, or equivalently, with complete bipartite partitionings of a given graph. To define these concepts more accurately, let $B = (U_m, V_p, E(B))$ be an arbitrary bipartite graph, where, as before, $U_m = \{u_1,...,u_m\}$ and $V_p = \{v_1,...,v_p\}$. The set of incidence matrices of complete bipartite partitionings of B is denoted by S_B, and the convex hull of S_B is denoted by Q_B. Clearly, if $B = K_{mp}$, then $S_B = S_{mp}$ and $Q_B = Q_{mp}$. In fact, the polyhedron Q_B can be viewed in the space $\mathbb{R}^{mp}$ as the face of Q_{mp} with the property that, for all $X \in Q_B$, $x_{ij} = 0$ when $(i,j) \notin E(B)$.

The dimension of Q_B is $|E(B)|$, since the subgraph of B containing no edges at all, as well as any subgraph containing only one edge of B, are complete bipartite partitionings of B. By the same reasoning, the *trivial inequalities* $x_e \geq 0$ and $x_e \leq 1$ are facet-defining for Q_B, for all $e \in E(B)$.

Suppose now that B_1 and B_2 are two bipartite graphs on the same vertex-set, and differing only in one edge (u_i, v_j), for example $E(B_2) = E(B_1) \cup \{(u_i, v_j)\}$. Our first result follows directly from the sequential lifting procedure described in Nemhauser and Wolsey (1988), combined with the observation that Q_{B1} is a facet of Q_{B2} :

Proposition 1. Consider the valid inequality $\prod X \leq \pi_0$ and assume that it defines a facet of Q_{B1}. Then, the inequality $\prod X + \pi_{ij} x_{ij} \leq \pi_0$ defines a facet of Q_{B2} iff

$$\pi_{ij} = \pi_0 - \max \{ \prod X \mid X \in S_{B2} \text{ and } x_{ij} = 1 \}.$$

This proposition guarantees that, when a facet-defining inequality is derived for a 'partial' polyhedron Q_B, this inequality can always be lifted to a facet-defining inequality of Q_{mp}. The following proposition shows that an inequality defining a facet of Q_{mp} is also facet-defining for each of the polyhedra corresponding to block diagonal matrices with at least m rows and p columns. It is similar in spirit to Theorem 3.3 in Grötschel and

Wakabayashi (1990), Theorem 3.2 in Chopra and Rao (1991) and Theorem 2.2 in Deza and Laurent (1992).

Proposition 2. Assume that the inequality $\Pi X \leq \pi_0$ defines a facet of Q_{mp}, and let $a,b \in \mathbf{N}$. Then the inequality $\Gamma Y \leq \pi_0$ defines a facet of $Q_{m+a,p+b}$, where $\Gamma \in \mathbf{R}^{m+a,p+b}$, $\gamma_{ij} = \pi_{ij}$ for $i \leq m$ and $j \leq p$, and $\gamma_{ij} = 0$ otherwise.

Consider the inequality $\Pi X \leq \pi_0$ and assume that it defines a nontrivial facet of Q_B. Without any information about the numerical values of the coefficients (Π,π_0), some general structural properties of the inequality can be stated. To do this, associate with the inequality two edge-sets E and E^+, defined as follows:

$$E := \{ (u_i,v_j) \mid (u_i,v_j) \in B \text{ and } \pi_{ij} \neq 0\}, \quad E^+ := \{ (u_i,v_j) \mid (u_i,v_j) \in B \text{ and } \pi_{ij} > 0\}.$$

We call the graph $H := (V(E), E)$ (respectively $H^+ := (V(E^+), E^+)$) the *support* (respectively the *positive support*) of the inequality $\Pi X \leq \pi_0$.

Proposition 3. If B is a nonempty bipartite graph, and $\Pi X \leq \pi_0$ induces a facet of Q_B, then: (1) $\pi_0 > 0$;
(2) E^+ is nonempty;
(3) the support H of $\Pi X \leq \pi_0$ is connected;
(4) the positive support H^+ of $\Pi X \leq \pi_0$ is connected;
(5) $V(E) = V(E^+)$.
If moreover B is a complete bipartite graph, then:
(6) $E \backslash E^+$ is nonempty, i.e. Π has negative elements;
(7) the support H of $\Pi X \leq \pi_0$ is two-connected.

3. FACET-DEFINING INEQUALITIES

We present in this section various classes of facet-defining inequalities for Q_{mp}. These inequalities will be obtained by lifting facet-defining inequalities for a face Q_B of Q_{mp} (according to Proposition 1). Some of the subgraphs B which we will consider are 'squares' (i.e. C_4's), so-called 'spiked' C_4-free connected bipartite graphs and cycles.

For a given $X \in \{0,1\}^{mp}$ and a given subgraph B of K_{mp}, we use the shorthand x(B) to denote the sum $\Sigma(x_{ij} \mid (u_i,v_j) \in E(B))$.

3.1. Square inequalities

Crama and Oosten (1992) observed that the *square inequalities* :

$$x_{hj} + x_{hk} + x_{ij} - x_{ik} \leq 2 \quad (h,i \in \{1,...,m\}, j,k \in \{1,...,p\})$$

are valid for S_{mp} (hence, for its convex hull Q_{mp}), and that they yield, together with the integrality constraints on X, a valid description of S_{mp}; that is,

$$S_{mp} = \{ X \in \{0,1\}^{mp} \mid x_{hj} + x_{hk} + x_{ij} - x_{ik} \leq 2 \text{ for all } h,i \in \{1,...,m\} \text{ and } j,k \in \{1,...,p\} \}.$$

In fact, it is easy to see that the square inequalities are facet-defining for Q_{22}, and hence we deduce from Proposition 2 that they are facet-defining for Q_{mp}, for all $m,p \geq 2$.

3.2. Facet-defining inequalities based on spiked C_4-free connected bipartite graphs

The term 'spike' refers to special edges of the positive support H^+ of a valid inequality, say $\prod X \leq \pi_0$, or to the corresponding coefficients of Π. A *spike-leaf* of H^+, or of $\prod X \leq \pi_0$, is a vertex covered by exactly one edge of the subgraph H^+. That covering edge is called a *spike*. A *spike-root* is a vertex covered by a spike, but not a spike-leaf itself. For example the square inequality $x_{hj} + x_{hk} + x_{ij} - x_{ik} \leq 2$ has two spike leaves (vertices u_i and v_k), two spikes (the edges (u_h,v_k) and (u_i,v_j)) and two spike roots (vertices u_h and v_j).

Notice that there are facet-defining inequalities whose support consists of exactly one spike, namely the trivial inequalities $x_{ij} \leq 1$. It follows from Proposition 3(3) in Proposition 3 that nontrivial facet-defining inequalities never contain spikes covering two spike-leaves.

We say that a graph B is *spiked* if each vertex of B is covered by exactly one spike. We say that B is a *C_4-free* graph if it does not containany cycles of length four (i.e., C_4's). The following holds:

Proposition 4. For $k \in \mathbf{N}$, if B is a spiked C_4-free connected bipartite graph with exactly k spikes, then the inequality $x(B) \leq k$ defines a facet of Q_B.

In view of Propositions 1 and 2, the inequality $x(B) \leq k$ defined in Proposition 4 can be lifted to a family of facet-defining inequalities of Q_{mp}, for all $m,p \geq k$. A subset of this family can be described explicitly. To achieve this, a new definition is needed: a subset C of $(U_m \times V_p) \backslash E(B)$ is called a *chord set* for a spiked tree B if, for each path between two spike-leaves of B, there is an edge in C linking two (arbitrary) vertices of the path. The following proposition holds:

Proposition 5. For $k \in \mathbf{N}$ and $m,p \geq k$, if B is a spiked tree with exactly k spikes and C is a minimal chord set for B, then the inequality $x(B) - x(C) \leq k$ defines a facet of Q_{mp}.

To get better acquainted with these spiked tree inequalities, consider for instance the special case in which the spike-roots $u_2, u_3, \ldots, u_k$ of the spiked tree B are all adjacent to the spike-root v_1, as shown in Figure 1 below. A minimal chord set C for this tree must consist of the following edges : for all $i,j \geq 2$, (u_i,v_j), and either (u_i,v_i) or (u_j,v_i). Carrying out this construction with $k = 1$ or $k = 2$ demonstrates that the trivial inequalities $x_{ij} \leq 1$ and the square inequalities belong to the family of spiked tree inequalities.

Another subset of the family of facet-defining inequalities based on C_4-free connected bipartite graphs can also be described explicitly as follows. Let a *spiked cycle* be a spiked graph whose spike roots induce a cycle (notice that this is a slight abuse of our general definition of a spiked graph, since a spiked cycle is not a cycle; but this abuse is convenient, and will not cause any confusion). A subset C of $(U_m \times V_p) \backslash E(B)$ is a *chord set* for the spiked cycle B if, for each pair (s,t) of spike-leaves of B, and for each of the two paths P_i (i=1,2) between s and t, there is an edge $e(s,t,P_i) \in C$ such that :

(a) $e(s,t,P_i)$ links two (arbitrary) vertices of P_i (i=1,2);

(b) $e(s,t,P_1)$ and $e(s,t,P_2)$ are distinct;

(c) if one of to the leaves s and t is covered by both edges $e(s,t,P_1)$ and $e(s,t,P_2)$, then one of these edges covers s and t.

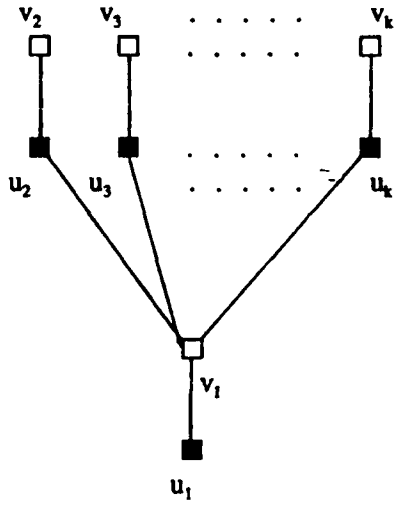


Figure 1. A spiked tree.

Proposition 6. For all $m,p \geq k \geq 6$, if B is a spiked cycle with exactly k spikes and C is a minimal chord set for B, then the inequality $x(B) - x(C) \leq k$ defines a facet of Q_{mp}.

The smallest example of a C_4-free spiked cycle is shown in Figure 2. Call this graph B_6.

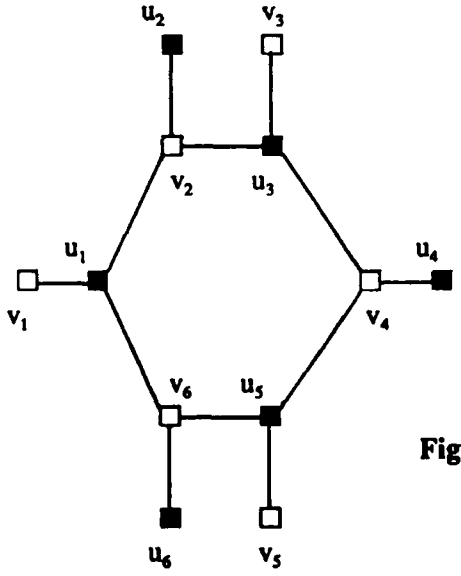


Figure 2. A spiked cycle with 6 spikes.

It is easy to see that all of the edges (u_2,v_1), (u_2,v_3), (u_4,v_3), (u_4,v_5), (u_6,v_5) and (u_6,v_1) must be in any chord set for B_6, but on the other hand, there exist various ways to complete this list to a minimal chord set. As a matter of fact, it can be checked that each of the matrices Π^1, Π^2 and Π^3 hereunder gives rise to a facet-defining inequality of the form $\Pi X \leq 6$, derived from B_6 as explained in Proposition 6:

$$\Pi^1 = \begin{bmatrix} 1 & 1 & 0 & 0 & -1 & 1 \\ -1 & 1 & -1 & -1 & 0 & 0 \\ -1 & 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & -1 & 1 & -1 & -1 \\ 0 & 0 & -1 & 1 & 1 & 1 \\ -1 & -1 & 0 & 0 & -1 & 1 \end{bmatrix} \quad \Pi^2 = \begin{bmatrix} 1 & 1 & -1 & -1 & 0 & 1 \\ -1 & 1 & -1 & -1 & 0 & 0 \\ 0 & 1 & 1 & 1 & -1 & -1 \\ 0 & 0 & -1 & 1 & -1 & -1 \\ -1 & -1 & 0 & 1 & 1 & 1 \\ -1 & -1 & 0 & 0 & -1 & 1 \end{bmatrix} \quad \Pi^3 = \begin{bmatrix} 1 & 1 & -1 & -1 & -1 & 1 \\ -1 & 1 & -1 & -1 & 0 & -1 \\ 0 & 1 & 1 & 1 & 0 & 0 \\ -1 & 0 & -1 & 1 & -1 & 0 \\ 0 & 0 & -1 & 1 & 1 & 1 \\ -1 & 0 & 0 & -1 & -1 & 1 \end{bmatrix}$$

The question arises wether it is possible to describe explicitly other, possibly more general, subfamilies of facet-defining inequalities based on C_4-free connected bipartite graphs. In Section 5, we present a patching procedure which partially answer this question.

Some interesting variants of the spiked tree and spiked cycle inequalities can be generated by adding a single special vertex to B. This yields facet-defining inequalities whose positive support is not spiked. Details are omitted from this extended abstract.

Let us finally observe that the incidence graphs of projective planes are very special (non-spiked) C_4-free connected bipartite graphs, which also give rise to interesting valid inequalities for Q_{mp}. Details are again omitted.

3.3. Facet-defining inequalities based on cycles

Let C_k be a cycle of length k, with k even. If $k \geq 6$, then no component of a complete bipartite partitioning of C_k can contain more than two edges. Therefore, the total number of edges of a complete bipartite partitioning of C_k cannot exceed $\frac{2}{3}$ k. If k is not a multiple of three, then the inequality $x(C_k) \leq \lfloor \frac{2}{3} k \rfloor$ is facet-defining for Q_{Ck}.

Define now $3C_k$ to be the graph induced by the three-chords of C_k (a three-chord of C_k is an edge joining two vertices at distance 3 in C_k). Then the following holds:

Proposition 7. For all $k \geq 4$ and all m, p with $k \leq 2\min\{m,p\}$, the inequality $x(C_k) - x(3C_k) \leq \lfloor \frac{2}{3} k \rfloor$ is valid for Q_{mp}. If $k = 1 \pmod 3$, then the inequality induces a facet of Q_{mp}.

4. LIFTING THEOREMS

Let $v_j \in V_p$. The *covering* $c_\Pi(v_j)$ (also denoted as $c(v_j)$, when no confusion can arise) of a vertex v_j with respect to the valid inequality $\Pi X \leq \pi_0$ for Q_{mp} is defined as follows:

$$c_\Pi(v_j) := \pi_0 - \max \{ \Pi X \mid X \in S_{mp}, \text{ and } x_{ij} = 0 \text{ for all } i = 1, 2, ..., m \}.$$

(A similar definition would of course apply to any vertex $u_i \in U_m$). The covering of an arbitrary vertex is always nonnegative. The covering of v_j is zero if $\pi_{ij} = 0$ for all i = 1, 2, ..., m, i.e. if v_j is not covered by any edge in the support of the inequality.

A *tight inequality* is a valid inequality for Q_{mp} with the property that there exists a complete bipartite partitioning X satisfying the inequality with equality, and such that all vertices having a strictly positive covering with respect to the inequality are in the same connected component of X (it can be checked that all square inequalities are tight).

Finally, call *U-extension of* K_{mp} the graph B obtained from K_{mp} by adding a vertex u_{m+1} to U_m, a vertex v_{p+1} to V_p, and all the edges between u_{m+1} and $V_p \cup \{v_{p+1}\}$; that is,

$$B := (U_m \cup \{u_{m+1}\},\ V_p \cup \{v_{p+1}\},\ (U_m \times V_p) \cup (\{u_{m+1}\} \times V_p) \cup (u_{m+1}, v_{p+1})).$$

The following statement describes how a tight facet-defining inequality for Q_{mp} can be lifted to a facet-defining inequality for Q_B:

Proposition 8. Let $\Pi X \leq \pi_0$ be a tight facet-defining inequality for Q_{mp}. Let the inequality $\Gamma Y \leq \gamma_0$ be constructed in the following way:

$$\gamma_{ij} = \begin{cases} \pi_{ij} & \text{if } (u_i, v_j) \in U_m \times V_p; \\ c_\Pi(v_j) & \text{if } i = m+1 \text{ and } v_j \in V_p; \\ \sum_{k=1}^{p} c_\Pi(v_k) & \text{if } i = m+1 \text{ and } j = p+1; \end{cases}$$

$$\gamma_0 = \pi_0 + \sum_{k=1}^{p} c_\Pi(v_k).$$

Then the inequality $\Gamma Y \leq \gamma_0$ defines a facet of Q_B, where B is the U-extension of K_{mp}.

Proposition 8, together with Propositions 1 and 2, implies that the inequality $\Gamma Y \leq \gamma_0$ can in turn be lifted to a family of facet-defining inequalities for Q_{nq} ($n \geq m$, $q \geq p$). A special subset of such inequalities, which we call 'totally spiked tight inequalities', can be described explicitly. A *totally spiked inequality* is an inequality $\Pi X \leq \pi_0$ whose support is spiked, and such that the *spiked solution* S, defined by $s_{ij} = 1$ if and only if $\{u_i, v_j\}$ is a spike, satisfies $\Pi S = \pi_0$. A simple example of totally spiked inequality is again provided by any square inequality, or by any of the facet-defining inequalities described in Section 3.2. Now, our next proposition allows to lift totally spiked, tight, facet-defining inequalities for Q_{mp} to totally spiked, tight, facet-defining inequalities for $Q_{m+1,p}$ (of course, a similar result holds for $Q_{m,p+1}$). When stating this result, we assume that the spikes, the spike-roots and the spike-leaves of the inequality are numbered in such a way that spike-root i and spike-leaf i are covered by spike i.

Proposition 9. Let $\Pi X \leq \pi_0$ be a totally spiked tight inequality defining a facet of Q_{mp}. Let γ_0 and γ_{ij} be defined as in Proposition 8 for all edges (u_i, v_j) of the U-extension of K_{mp}, and let $\gamma_{ij} = \sum_{k=1..p \mid k \neq i} [\ \pi_{ik}\ ]^+$ if $u_i \in U_m$ and $j = p+1$. Then, the inequality $\Gamma Y \leq \gamma_0$ is a totally spiked tight inequality defining a facet of $Q_{m+1,p}$.

5. PATCHING FACET-DEFINING INEQUALITIES

Sometimes, families of valid inequalities, or even facet-defining inequalities, can be constructed by combining together a number of other valid inequalities. This section presents such a *patching* procedure. For simplicity, we assume that only two valid inequalities for Q_{mp} are to be combined, say $AX \leq a_0$ and $BX \leq b_0$. We denote by A and B, respectively, the vertex-sets of the supports of these two inequalities; A and B are assumed to be disjoint. We define the *neighborhood* of a vertex i with respect to A to be the set $N_A(i) := \{\ j : a_{ij} > 0\ \}$ (similarly for B). Notice that a vertex is not in its own neighborhood.

Now the patching procedure can be roughly sketched as follows. It makes use of the concept of covering introduced in Section 4. First, vertices having a strictly positive

covering with respect to $AX \leq a_0$ are selected in A, in such a way that their neighborhoods with respect to A are disjoint; an equal number of vertices are selected in B in a similar way. Then, the selected vertices from A and B are matched. The graph K_{mp} for which a valid inequality $\Gamma Y \leq \gamma_0$ will be derived, is the complete bipartite graph induced by the union of the vertex-sets A and B. Construct now the coefficients of the inequality $\Gamma Y \leq \gamma_0$ as follows:

$$\gamma_{ij} = \begin{cases} a_{ij} & \text{if } u_i \in A \text{ and } v_j \in A, \\ b_{ij} & \text{if } u_i \in B \text{ and } v_j \in B, \\ \min\{ c_B(u_i), c_A(v_j)\} & \text{if } (u_i,v_j) \text{ is a matched pair, } u_i \in B \text{ and } v_j \in A, \\ \min\{ c_A(u_i), c_B(v_j)\} & \text{if } (u_i,v_j) \text{ is a matched pair, } u_i \in A \text{ and } v_j \in B, \\ -\gamma_{kh} & \text{if } (u_h,v_k) \text{ is a matched pair, } u_i \in N_A(u_h) \text{ and } v_j \in N_B(v_k), \\ & \quad \text{or } u_i \in N_B(u_h) \text{ and } v_j \in N_A(v_k), \\ 0 & \text{otherwise,} \end{cases}$$

$$c_0 = a_0 + b_0.$$

Proposition 10. The inequality $\Gamma Y \leq \gamma_0$ is a valid inequality for Q_{mp}.

Proposition 11. If $AX \leq a_0$ and $BX \leq b_0$ are valid inequalities for Q_{mp} which have been obtained by patching together a number of spiked tree and spiked cycle inequalities, then the inequality $\Gamma Y \leq \gamma_0$ obtained by patching $AX \leq a_0$ and $BX \leq b_0$ as explained above is facet-defining for Q_{mp}.

ACKNOWLEDGEMENTS: The first author was partially supported in the course of this research by NSF grant DMS89-06870, AFORS grant 90-0008, and ONR grant N0001492J1375 to Rutgers University.

REFERENCES

Bondy, J.A., and Murty, U.S.R. (1976), *Graph Theory with Applications*, North Holland, New York.

Chopra, S., and Rao, M.R. (1991), On the multiway cut polyhedron, *Networks* 21, 51-89.

Crama, Y., and Oosten, M. (1992), "Models for machine-part grouping in cellular manufacturing", Res. Rep. M 92-04, University of Limburg, Dept. Mathematics, Maastricht, Netherlands.

Deza, M., and Laurent, M. (1992), "Facets for the cut cone I", *Mathematical Programming* 56, 121-160.

Faigle, U., Schrader, R., and Suletzki, R. (1986),"A cutting-plane algorithm for optimal graph partitioning". In: W. Domschke, W. Krabs, J. Lehn, P. Spellucci (eds.), *Methods of Operations Research: XI Symposium on Operations Research*, Athenäum, Darmstadt.

Gondran M., and Minoux, M. (1984), *Graphs and Algorithms*, John Wiley & Sons, New York.

Grötschel, M., and Wakabayashi, Y. (1989), "A cutting plane algorithm for a clustering problem", *Mathematical Programming* 45, 59-96.

Grötschel, M., and Wakabayashi, Y. (1990), "Facets of the clique partitioning polytope", *Mathematical Programming* 47, 367-387.

Nemhauser G.L.,and Wolsey L.A. (1988), *Integer and Combinatorial Optimization*, John Wiley & Sons, New York.

A flexible framework for Group decision Support: WINGDSS Version 3.0

P. Csáki, L. Csiszár, F. Fölsz, K. Keller,
Cs. Mészáros, T. Rapcsák, P. Turchányi

Department of Operations Research and Decision Systems
Computer and Automation Institute
Hungarian Academy of Sciences
H-1518 P.O.Box 63, XI. Kende u. 11-13, Budapest, Hungary

1. Introduction

For more than a decade already, modeling, methodological research and software development for stand-alone decision support systems have belonged to the scope of our department. Both on mainframes and PC's, we have had several joint project with the Hungarian Electricity Board on electrical energy optimization problems, projects on inventory and production control have been carried out in the steel industry, smaller special applications, for example menu planning for hospitals, optimal design of trusses for a bus manufacturing company must also be mentioned among the succesful applications.

In our department, research and software development on group decision support began in 1989 only, by a small team [1]. The reason for the increasing interest for such systems is quit simple: in today's organizations decisions are made mostly collectively. As managers spend more of their time in meetings, the study of information technology to support meetings becomes increasingly important.

Several type of group support systems have been developed by the Group Support Systems research community, varying from collaborate writing to computer supported negotiation and decision making. A Group Support System can support meetings, which are distributed geographically and temporally. Tasks in a group decision situation include communication, planning, idea generation, problem analysis and design, problem solving, negotiation, conflict resolution, collaborative document preparation. Group support systems should provide the sharing of information among group members and between the group and computer. Decision makers may

[1] Research supported in part by the Hungarian Foundation for Scientific Research, grant No.2568

get individually all the necessary information they need, and, in some extent, they can carry out their task-dependent decision process also individually. The group support system should help in defining and formalizing the decision problem for the group. The system should provide the necessary data, tools and methods for solving the specific decision task individually. Last but not least, the system should help to achive the result satisfactory for all group members.

The basic concept for us has been to develope a rather flexible framework, which

- has an attractive user interface,
- is adjustable to different type of group decision situation,
- is able to integrate the knowledge and experiences, accumulated over the last decade in our department on stand-alone decision support system design and development.

Within three years, a PC based system working in the MS WINDOWS environment has been realized. At present, we are like conducting a mission with our WINGDSS system in the really difficult process of convincing people to use computers for supporting their group decision problems, but the real life applications of WINGDSS should convince its possible users about its higher efficiency.

WINGDSS has already been proved very helpful in evaluating bids for tenders. for example at the Tender Bureau of the Hungarian Telecommunication Company. We developed a model for appraisal of hotels for the State Property Agency. At the Ministry of Welfare, the purpose of the usage of WINGDSS is to support budget allocation processes for social institutions. We are working on extending the applicability of our system for more complex problems, for example in environmental impact analysis problems. Our experiences collected with real life applications define new directions for further developments in WINGDSS.

2. What kind of group decision processes are supported by WINGDSS ?

The decision problem can be typifed as follows:

A group of experts from different fields but with a common interest has the task of ranking certain alternatives characterized by a finite set of properties or attributes. Attributes can be factual data and subjective factors. Applying a proper utility function to the set of alternatives leads to a ranking of the alternatives according to their numerical values. The individual ranking will reflect the individual

preferences, group ranking, in addition, will incorporate the differences of priorities and the expertise of decision makers. The arrival to a group ranking satisfactory to all members is supported by a series of possibilities for the interactive usage of WINGDSS. The system provides userfriendly tools for a lot of operations that can be carried out in the decision process during program execution, on-screen, by the users themselves. Practically, feedbacks from the individuals can be integrated at any stage of the decision process. The system is always ready for updates.

In the past, stand-alone PC-s were more frequently used in Hungary than LANs and workstations, this is why the present version works on a single PC. In spite of that, the system provides the athmosphere of a decision room with networked computers: task formulation, idea generation, and team building is supported in many ways, but at the same time, the individuals' privacy is ensured as well.

The group decision process is concerned as a *three-phase event*:

- the preparation of the decision task,
- the process of individual evaluation,
- the phase of aggregation (group result processing).

This concept defined three main menu groups for a virtual separation of the activities, however, they do not describe the sequence of actions obligatory: moving back and forth among the different phases is possible at any stage.

2.1 Task preparation phase

The key problem is transforming the actual decision task into an appropriate form. *Idea organization* is one of the main issues in a decision process. The hierarchy of criteria is a tree in our wingdss: one starts with the most general criterion, which corresponds to the root of the tree, and gradually decomposes it to more specific criteria. The leaves represent the criteria, which can be evaluated independently from each other. Some decision problems can be represented with a tree of several levels, while others are less decomposable. In the earlier versions of WINGDSS, variables defined at one leaf criterion were not reachable at the other leaves. The third version eliminated this drawback by separating the definition and storing of the variables from the definition and storing of the tree–components.

Creating and modifying the criterion tree with on screen operations is technically possible due to a modul, which is applicable to graph handling tasks separately from the WINGDSS system as well: nodes and subtrees can be constructed, moved,

copied, deleted, renamed and arranged in several ways.

The data of *alternatives* can be typed in directly or they can also be selected from an *outer database.* Any database handling system, running under MS Windows, can be fitted to WINGDSS. A methodology for selecting records from the outer database can also be defined from the WINGDSS, providing a screening on the alternatives.

The evaluation of the alternatives starts at the leaf criteria, with *functions defined exlusively to the actual decision task*. Finding the appropriate functions and / or procedures is also a key problem in decision support. Thanks to an interpreter built into the WINGDSS, the functions can also be created or modified on screen by any authorized individual. The system offers a collection of ready made functions as well. Version 3.0 has already the capability of integrating *program solvers*.

Once the problem has been set up in the necessary form, the next steps are,

- for each decision maker (DM), to assign weights to each criteria reflecting their importance,

- to assign weights – voting powers to each DM at each criteria, expressing the DM's competency in evaluating the criteria.

We assume the presence of a system facilitator or supervisor, who, with on screen operations, composes the decision group, determines the individuals' authorities, and assigns the voting powers. The authorities include the right to construct and modify the decision task (criteria, alternatives, evaluation procedures), the right for participating in the individual and in the group decision process.

2.2 Individual decision phase

Criteria are factual or subjective data. The functions or procedures defined at the leaf criteria must be identical for all group members, but the values of these functions are equal on factual data only. The result of the decision makers' individual evaluation will still vary due to the diversity of preferences. The weights expressing the preferences of a DM should be given explicitly, but *we plan to integrate methods for this process*. The function values on the subjective criteria are very likely dependent on the experts' opinion, and the individual preferences will modulate these differences further. The final score of an alternative during the individual decision process will be calculated as the weighted average of the function values, starting at the leaf criteria, combining them with the weights, and then proceeding toward the root of the criteria tree. The mathematical formulation is relativly simple:

Consider a decision problem with l group members $D_1 \dots D_l$, n alternatives $A_1 \dots A_n$ and m criteria $C_1 \dots C_m$.

Denote the result of the individual evaluation of decision maker D_k for alternative A_j on each leaf criterion C_i by $a^k_{i,j}$. Assume that the problem arising from the differences in dimension of the attributes has already been settled.

Let $w^k_i \geq 0$ weight assigned by D_k to C_i, $\quad i = 1 \dots m$ at each branching of the tree.

The calculation starts at each simple subtree (denoted by N') consisting from leaf criteria and their father, by the formula

$$\mu^k_j = \frac{\sum_{i \in N'} w^k_i a^k_{i,j}}{\sum_{i \in N'} w^k_i} \qquad j = 1 \dots n, \qquad k = 1 \dots l. \tag{1}$$

The μ^k_j value is assigned to the root of this simple subtree. The calculation proceeds toward the root of the criterion tree with combining the weights on the higher level criteria with values resulted from one level below. The individual utility given by D_k for A_j will be assigned to the root.

Note that an additive multiattribute model is only applicable to the decision problems when the additive independence of the criteria can be proved [4], [5].

2.3 Group ranking phase

For objective attributes only the weights given by a decision maker will be revised (at each criterion) by the *voting power for weighing*. However, in case of subjective attributes, not only the weights but also the evaluation itself (the $a^k_{i,j}$ values) will be modified at the corresponding leaf criteria by the ***voting power for qualifying***, where

$V(w)^k_i$ is the voting power assigned for D_k for the DM's weighing on a criterion C_i, and

$V(q)^k_i$ is the voting power assigned for D_k for the DM's qualifying on a ***subjective leaf criterion*** C_i.

Now the method of calculating the group utility of the alternative A_j is carried out on the tree of criteria, basically in the same way as it has been done by calculating the individual utilities.

First we aggregate the weights at each node i and get the ***group weights***:

$$W_i = \frac{\sum_{k=1}^{l} V(w)_i^k w_i^k}{\sum_{k=1}^{l} V(w)_i^k}, \qquad i = 1 \ldots m. \tag{2}$$

Then we compute the *aggregated qualification* at each *leaf* criterion C_i and get the *group qualification* at the *leaves* for each alternative A_j:

$$Q_{i,j} = \frac{\sum_{k=1}^{l} V(q)_i^k a_{i,j}^k}{\sum_{k=1}^{l} V(q)_i^k}, \qquad i \in N', \qquad j = 1 \ldots n. \tag{3}$$

The group utility of A_j is the result of the linear combination of the aggregated qualification values with the aggregated weights (proceeding from the leaf level toward the root):

$$U_j = \frac{\sum_i W_i Q_{i,j}}{\sum_i W_i}, \qquad j = 1 \ldots n \tag{4}$$

A *correct* group utility function must satisfy the axioms given in [6]. The function (4) appearently used in WINGDSS is appropriate in this respect.

The third main menu group provides various possibilities to compare the decision makers' individual weighing and evaluation. The opinions of other group members will often cause one member to reconsider and modify his evaluation. Such feedbacks can be realized by WINGDSS: any decision maker is allowed to activate the appropriate menu again for performing modifications in the evaluation of the subjective criteria or for changing his/her preference structure (the individual weights on the criteria). Changes in the structure of the decison task, and /or in the function at the leaf criteria can be performed by the Supervisor or any authorized user.

2.4 Sensitivity analysis of the result

Analysis the impact of certain decision parameters (individual preference structure, voting power of decision makers) to the final result can be performed with a method developed separately [8], integrated recently into Version 3.0. The algorithm can be used for different purposes:

- What are the intervals in which the weights can vary without effecting the ranking of the alternatives?

- If the weights are allowed to vary in given intervals, how the value and position of the alternatives will be changed?

- What kind of transformations are needed to change the position of one particular alternative (to make one low ranked alternative acceptable, for example)?

- If group members will agree in the ranking of a subset of the alternatives (the top set of one, two, three, four, ... alternatives), what changes are required in the weights?

3. TECHNICAL DATA

Distributional format:
One floppy disk of 740 KB or higher.
Hardware requirements:
IBM-PC/AT 386 or 486, VGA card, mouse.
Operating system:
PC-DOS 3.3 or higher.
Software requirement:
MS Windows operating environment version 3.0 or higher.

HOW THE AUTHORS DID CONTRIBUTE TO WINGDSS?

Tamás Rapcsák and Piroska Turchányi are the present managers of the Group Decision Project, together with Kriszta Keller, they also carry out research and modelling. The tree handling modul and the handy interface for data input has been developed by Péter Csáki. Levente Csiszár has been working on the interface to program solvers, on the interpreter for the criteria evaluating functions, on the methods both in the individual and group evaluation phase. Ferenc Fölsz has been responsible for any kind of data base functionalities. The sensitivity algorithm is from Csaba Mészáros.

REFERENCES

[1] M. Bíró, P. Turchányi and M. Vermes, CONDOR: Consensus Development and Operations Research tools in Group Decision Support Sytems, Research Report CAI HAS 23/1989 (December 1989)

[2] P. Csáki, T. Rapcsák, P. Turchányi and M. Vermes, WINGDSS: a Microsoft Windows based Group Decision Support System, presented at the 2nd ISDSS Conference in Ulm, submitted to the International Journal of Decision Support Systems

[3] S.Dyer and K. Sarin, Measurable Multiattribute Value Functions, Operations Research 27, No. 4 (July–August 1979)

[4] P.C. Fishburn, Utility Theory for Decision Making, John Wiley & Sons, New York, 1970.

[5] R.L. Keeney, Multiplicative Utility Functions, Operations Research 22, No. 1 (January-February 1974)

[6] R.L. Keeney, Group Preference Axiomatization with Cardinal Utility, Management Science 23, No. 2 (October 1976)

[7] R.L.Keeney, Building Models of Values, European Journal of Operations Research 37, No. 2 (November 1988)

[8] Cs. Mészáros and T. Rapcsák, Sensitivity Analysis on Decision Problems, Research Report CAI HAS (Aug. 1992)

[9] S.Soofi and J.Retzer, Adjustment of Importance Weights in Multiattribute Value Models by Minimum Discrimination Information, European Journal of Operations Research 60, No. 1 (July 1992)

[10] R.Vetschera, Integrating Databases and Preference Evaluations in Group Decision Support, Decision Support Systems 7, No. 1 (January 1991)

[11] M.J. Wang, H.P. Singh and W.V. Huang, A Decision Support System for Robot Selection, Decision Support Systems 7, No. 3 (August 1991)

Global optimal solutions with tolerances and practical composite laminate design

Tibor Csendes*, Zelda B. Zabinsky†, Birna P. Kristinsdottir†

Consider the nonlinear optimization problem

$$\min f(x) \tag{1}$$

where $f(x) : \mathbb{R}^n \to \mathbb{R}$ is a continuous nonlinear function, and the variables are subjects of the constraints

$$g_j(x) \leq 0 \qquad j = 1, 2, \ldots, m, \tag{2}$$

where $g_j(x) : \mathbb{R}^n \to \mathbb{R}$ are also continuous functions. Let us denote the set of feasible points by A, that is $A := \{x \in \mathbb{R}^n : g_j(x) \leq 0 \text{ for each } j \in (1, 2, \ldots, m)\}$.

It happens many times that the solution x^* (or an approximation of it) of a constrained nonlinear optimization problem is known, yet this result is not suitable for practical use. It is the case when the solution should be realized with a certain tolerance $\delta > 0$. If, moreover, at least one of the constraints is active at the solution, then the n-dimensional interval $[x_i^* - \delta, x_i^* + \delta]$ for $i = 1, 2, \ldots, n$ is not entirely feasible (cf. [5] and [8]).

From practical point of view, it would be better to have a suboptimal solution in the form of an n-dimensional interval X^* (i.e. for which $f(x) \leq f(x^*) + \epsilon$ and $g_j(x) \leq 0$ $j \in (1, 2, \ldots, m)$, where $x \in X^*$, and $\epsilon > 0$). Such a result interval would also reflect the sensitivity of the objective function for changes in the arguments on the set of feasible points.

*Kalmár Laboratory, József Attila University, Szeged, Hungary
†Industrial Engineering Program, University of Washington, Seattle, USA

In contrast to interval optimization methods like [2], here an algorithm for finding a large feasible n-dimensional interval for constrained global optimization is presented. The resultant interval is iteratively enlarged about a seed point while maintaining feasibility. An interval subdivision method is used to check feasibility of the growing box. The algorithm utilises the inclusion functions [1,4,7] of the objective and constrain functions. These are calculated by natural interval extension. The resultant feasible interval is constrained to lie within a given level set, thus ensuring it is close to the optimum. It is proved that the algorithm converges in a finite number of iterations.

The ability to determine such a feasible interval is useful for exploring the neighbourhood of the optimum, and can be practically used in manufacturing considerations. The numerical properties of the algorithm are tested and demonstrated by an example problem, and the procedure is applied to a real life engineering design problem to construct manufacturing tolerances for an optimum design of composite materials.

Acknowledgements

This work was supported by the Grants OTKA 2879/1991 and OTKA 2675/1991, and by the Boeing Commercial Airplane Company and the NASA-Langley Research Center. The authors want to express their gratitude to Christian Lawo and Christian Jansson for making C-XSC [6] and BIBINS/2.0 [3] available.

References

[1] Alefeld, G., J. Herzberger: Introduction to Interval Calculations. Academic Press, New York, 1983.

[2] Csendes, T.: Test results of interval methods for global optimization. IMACS Annals on Computing and Applied Mathematics 12(1991) 417-424.

[3] Jüllig, H.-P.: BIBINS/2.0 — C++ Bibliotheken für Vektoren und Matrizen über beliebigem skalaren Datentyp unter Berücksichtigung

spezieller spärlicher Strukturen sowie Intervalldatentypen. Bericht 92.6, Technical University of Hamburg-Harburg, 1992.

[4] Klatte, R., U. Kulisch, M. Neaga, D. Ratz, Ch. Ullrich: PASCAL-XSC — Language Reference with Examples. Springer-Verlag, Berlin, 1992.

[5] Kovács, Z., F. Friedler, L.T. Fan: Recycling in a separation process structure. Accepted in AICHE Journal.

[6] Lawo, Ch.: C-XSC — A Programming Environment for Verified Scientific Computing and Numerical Data Processing. Manuscript, University of Karlsruhe, 1991.

[7] Ratschek, H., J. Rokne: Computer Methods for the Range of Functions. Ellis Horwood, Chichester, 1984.

[8] Zabinsky, Z.B., D.L. Graesser, M.E. Tuttle, G.I. Kim: Global Optimization of Composite Laminates Using Improving Hit-and-Run. In C. Floudas and P. Pardalos, editors, Recent Advances in Global Optimization, pages 343-368, Princeton University Press, Princeton, 1992.

Contact author: Tibor Csendes, Kalmár Laboratory, József Attila University, Hungary, Szeged, P.O. Box 652.

PREPROCESSING AND CUTTING PLANES FOR A CLASS OF PRODUCTION PLANNING PROBLEMS

Ken Darby-Dowman
Department of Mathematics and Statistics
Brunel, The University of West London
Uxbridge, Middlesex,
UB8 3PH,England.

Ma do Socorro N. Rangel
Department of Mathematics and Statistics
Brunel, The University of West London
and CNPq/UNESP - Brasil

I - INTRODUCTION

This paper investigates the properties of a class of integer programming model applied to a production planning problem. Each product may require processing on several machines and may involve precedence relationships. The machines are already (partially) committed and only the residual capacities of each machine in each time period of the planning horizon are available for use. The problem is to efficiently deploy unused capacities by determining, subject to market conditions, a production schedule. The model lies at the heart of a decision support system for advising sales executives in determining the products on which to focus their efforts. The models can be computationally demanding and techniques for speeding up solution times are highly desirable. Various preprocessing techniques have been investigated and their effectiveness evaluated. In addition, a number of cutting plane approaches have been applied. The performance of these approaches which are both general and application specific is examined.

II - THE MODEL

The production environment is made up of a set of manufacturing cells, each of which may have an amount of unallocated capacity (resource) in each of a set of time periods over the planning horizon. Each product can be produced according to a number of production structures, each of which specifies the cell resources required per unit of product in each of the (cell,time period) combinations used in the structure.

Let the following parameters define the size of the problem:

np = number of products
ns = number of production structures
nc = number of manufacturing cells
nt = number of time periods

And let (i,j,k,t) be the index set as defined below.

i : product, i = 1...np
j : production structure, j = 1...ns
k : manufacturing cell, k = 1...nc
t : time period, t = 1...nt

The problem data and the variables are:

cp_{ijkt} = *the amount of capacity used per unit of product i produced according to structure j, on cell k , in period t*

sp_{kt} = *the spare capacity for cell k in period t*

$limprod_i$ = *market size for product i*

v_i = *profit per unit of product i*

l_{ij} = *the minimum production quantity of product i, when produced according to structure j*

u_{ij} = *the maximum production quantity of product i, when produced according to structure j*

x_{ij} = *number of units of product i, to be produced according to production structure j*

$$y_{ij} = \begin{cases} 1 \text{ if product } i \text{ is produced according to structure } j \\ 0 \text{ otherwise} \end{cases}$$

The problem can be formulated as:

$$\max \quad \sum_{i=1}^{np} v_i \left(\sum_{j=1}^{ns} x_{ij} \right)$$

st:

$$\sum_{i=1}^{np} \sum_{j=1}^{ns} cp_{ijkt}\, x_{ij} \le sp_{kt} \qquad (1)$$

$$k = 1..nc, \quad t = 1..nt$$

$$\sum_{j=1}^{ns} x_{ij} \le limprod_i \qquad (2)$$

$$i = 1..np$$

$$x_{ij} - l_{ij}\, y_{ij} \ge 0 \qquad (3)$$

$$x_{ij} - u_{ij}\, y_{ij} \le 0 \qquad (4)$$

$$i = 1..np, \quad j = 1..ns$$

$$x_{ij} \ge 0 \text{ and integer}, \; y_{ij} \in \{0,1\}$$

Constraint set (1) states that the total amount of resource used in cell k, time period t, must not exceed the spare capacity; constraint set (2) represents the market size for each product i; and constraint sets (3) and (4) state the minimum and maximum quantity of product i, when produced according to structure j.

III - PREPROCESSING

In order to obtain a tighter formulation to the problem the following preprocessing techniques have been investigated.

1) Euclidean reduction [HP-91]

i) For each capacity row (constraint set (1)) let kexp be the smallest nonnegative integer such that:

$$c_{ijkt} * 10^{kexp} \text{ is integer for all } i, j$$

ii) Find K the greatest common divisor of the resulting integer coefficients;

iii) Multiply the row by 10^{kexp} and divide both sides by K;

iv) If the RHS after the division is not integer a tighter representation may be derived. Since the constraint type is 'less than or equal' the RHS can be truncated to the next lowest integer.

2) Redundant constraints [BMW-75]

For each capacity row:

i) Compute the constraint upper bound:

$$\text{Since } cp_{ijkt} \text{ is nonnegative for all } (i,j,k,t):$$

$$U_{kt} = \sum_{i=1}^{np} \sum_{j=1}^{ns} cp_{ijkt}\, u_{ij}$$

ii) The constraint is redundant if:

$$U_{kt} \leq sp_{kt}$$

3)Singleton rows [BMW-75]

Consider a capacity constraint such that:

$$\sum_{i=1}^{n_p} \sum_{j=1}^{n_s} cp_{ijkt}\, x_{ij} \leq sp_{kt}$$

where $cp_{ijkt}=0$ *for all* $i \neq g$ *and* $j \neq h$

Then let:

$$u'_{gh} = \frac{sp_{kt}}{cp_{ghkt}}$$

If:

i) $u'_{gh} \leq u_{gh}$ replace u_{gh} by $\lfloor u'_{gh} \rfloor$

ii) $u'_{gh} < l_{gh}$ fix $x_{gh} = 0$ and $y_{gh} = 0$

Remove the constraint.

4)Infeasibility and simple redundancy [HP-91]

In this form of preprocessing redundant constraints may be identified and removed. The procedure is as follows:

i)In each capacity row, determine the nonzero count.

ii)For the rows with equal nonzero count determine rows whose nonzeros match exactly both in terms of value and column number.

iii)When two rows agree, check their RHS:

a)if the RHS are equal, remove one of the rows;

b)if one inequality dominates the other, remove the dominated one.

This technique can be extended to detect infeasibilities in the form of conflicting constraints. However the capacity constraints of the model are all 'less than or equal to' and therefore this possibility cannot arise.

IV - CUTTING PLANES

The most efficient way to get a tighter formulation for an integer problem is to incorporate strong valid inequalities. We have investigated two classes of valid inequalities.

The first relates to a reduced form of the model. Consider the bounding constraints (3) and (4). Replace these constraints by the aggregate bounding constraints (5) and (6).

$$\sum_{j=1}^{ns} (x_{ij} - l_{ij}\, y_{ij}) \geq 0 \qquad (5)$$

$$\sum_{j=1}^{ns} (x_{ij} - u_{ij}\, y_{ij}) \leq 0 \qquad (6)$$

$$i = 1..np$$

The new model is a relaxation with 2np(ns-1) fewer bounding constraints. If any of the original bounding constraints are broken they can be introduced as cuts to the new model.

For the second approach, we consider the capacity rows together with constraints (3) and (4) and reformulate them as a single node fixed charge flow.

Given :

$$\sum_{i=1}^{np} \sum_{j=1}^{ns} cp_{ijkt}\, x_{ij} \leq sp_{kt} \qquad k = 1..nc,\ t = 1..nt$$

$$x_{ij} - l_{ij}\, y_{ij} \geq 0$$

$$x_{ij} - u_{ij}\, y_{ij} \leq 0 \qquad i = 1..np,\ j = 1..ns$$

Let:

$x'_{ij} = cp_{ijkt}\, x_{ij}$, $\quad u'_{ij} = cp_{ijkt}\, u_{ij}$, $\quad l'_{ij} = cp_{ijkt}\, l_{ij}$

and relax x_{ij} to $x_{ij} \in \mathbb{R}$

We get the single node fixed charge flow model

$$\sum_{i=1}^{np} \sum_{j=1}^{ns} x'_{ij} \leq sp_{kt}$$

$$k = 1..nc, \; t = 1..nt$$

$$l'_{ij} \, y_{ij} \leq x'_{ij} \leq u'_{ij} \, y_{ij}$$

$$i = 1..np, \; j = 1..ns$$

Suppose there exists C (a generalized cover) such that:

$$\lambda = \sum_{i,j \in C} u_{ij} - sp_{kt} > 0$$

Let:

$$\overline{u} \geq \max(\lambda, \max_{i,j \in C} u_{ij}), \; \overline{u_{ij}} = \max(\overline{u}, u_{ij}),$$

$$\text{and } \overline{C} = \{(i,j) \; / \; (i,j) \notin C\}$$

Then:

$$\sum_{i,j \in C} [x'_{ij} + (1 - y_{ij})(u'_{ij} - \lambda)^+] + \sum_{i,j \in \overline{C}} [x'_{ij} - y_{ij} (\overline{u_{ij}} - \lambda)] \leq sp_{kt}$$

$$\text{where} \quad (u'_{ij} - \lambda)^+ = \max(u'_{ij} - \lambda, 0)$$

is a valid inequality for the single node fixed charge flow model [VW-86].

These procedures are not restricted to this class of problem but have widespread applicability [VW-87], [PW-91].

V - SUMMARY

A production planing model to advise sales executives on the products on which they should concentrate in order to efficiently deploy unused factory capacities has been developed. To speed up the solution process, a set of preprocessing techniques and valid inequalities have been investigated. Experimental results for a

range of model sizes (to be reported) indicate that the procedures have a beneficial effect.

ACKNOWLEDGEMENT

We are grateful for the financial support provided by the Brazilian organizations: CNPq (Conselho Nacional de Desenvolvimento Científico e Tecnológico) e UNESP (Universidade Estadual Paulista 'Júlio de Mesquita Filho') which has enabled this research to be carried out.

We would also like to thank Richard Maher for bringing the application to our attention.

REFERENCES

[BMW-75] - Brearley, A.L., G. Mitra and H.P.Williams, 'Analysis of Mathematical Programming Problems Prior to Applying The Simplex Algorithm', **Mathematical Programming**, 8, 54-83, 1975.

[HP-91] - Hoffman, K.L. and M. Padberg , 'Improving LP-Representations of Zero-One Linear Programs for Branch-and-cut' - **ORSA Journal on Computing** , 3 (2), 121-134, 1991.

[PW-91] - Pochet, Y and L.A. Wolsey , ' Solving Multi-item Lot Sizing Problems using Strong Cutting Planes', **Management Science**, 17, 53-67, 1991.

[VW-86] - Van Roy, T.J. and L.A. Wolsey , Valid Inequalities for Mixed 0-1 Programs, **Discrete Applied Mathematics**, 14, 199 -213,1986.

[VW-87] -...'Solving mixed integer programming problems using automatic reformulation' , **Operations Research** , 35 (1), 45-57, 1987.

MODELLING STRATEGIES IN INTEGER PROGRAMMING APPLIED TO SCHEDULING A FLEET OF SHIPS

K. Darby-Dowman, *
C.Lucas*
G. Mitra*
J. Smith**

* Department of Mathematics and Statistics
Brunel, The University of West London
Uxbridge, Middlesex, UB8 3PH, England

** United States Coast Guard,
R&D Center, Groton, Connecticut CT06340-6096, USA

I. INTRODUCTION

The application considered in this paper is the determination of an annual, day-by-day, schedule for a fleet of United States Coast Guard cutters within a given geographical area. The tasks assigned to the cutters are varied and include patrolling in specified districts of the area, training exercises and maintenance. The feasibility of a particular cutter schedule is governed by a set of operational rules that depend, in part, on the timing and nature of the tasks already assigned to the cutter prior to the start of the scheduling year in question. Other factors include transit allowances before and after a task, the duration of in-port time after the completion of a task and cutter capabilities. The requirements placed on the fleet fall into two principal categories. The first relates to minimum levels of cover in terms of the number of cutters of given classes on patrol in each area at any given time. The second category of requirement concerns training and cutter maintenance. In these cases upper limits are placed on the number of cutters undergoing these tasks at any given time.

One approach to modelling scheduling problems of this type is to generate, for each cutter, a set of possible schedules, and to determine the 'best' fleet schedule by selecting one of the possible schedules for each cutter. This formulation leads to an integer programming model which has been widely advocated (e.g. [2]) in various guises.

However, in many scheduling applications, especially in the area of vehicle scheduling, the list of 'requirements' often results in the non-existence of a feasible solution. In such cases, a relaxation of the requirements is necessary in order to obtain a schedule. On further examination of the practical issues it is frequently the case that some of the 'requirements' merely reflect desirable characteristics rather than strict requirements.

In developing a scheduling model that yields useful solutions, Darby-Dowman and Mitra [1] proposed the extended set partitioning model which was essentially an integer goal programming formulation of a set partitioning model and admitted set covering, set packing and set partitioning as special cases.

In that model, the requirements were treated as targets and undercover (below target) and overcover (above target) were allowed but penalised. Our approach to the cutter scheduling problem follows a similar vein.

II. MODELLING THE SCHEDULING PROBLEM

Within the operational rules that govern the tasks and the cutter duties, a set of possible schedules is generated for each cutter. An optimal schedule is one which is as close to meeting requirements as is possible. The generic model is stated below:

Parameters:

nc = number of cutters to be scheduled

n_k = number of columns (possible schedules) for cutter k, k = 1,2,...nc

nt = number of time periods in the scheduling year

ng = number of constraint groups (schedule 'requirements')

Index sets

i : schedule requirement, i = 1,2,...ng

j : time period, j = 1,2,...nt

k : cutter identifier k = 1,2,...nc

l : identifier of possible schedule for cutter k
l = 1,2,...n_k

Model Variables

x_{kl} = 1 if the l'th possible schedule for cutter k is selected

= 0 otherwise

u_{ij} : Extent of the under-achievement in respect of schedule requirement i in time period j.

o_{ij} : Extent of the over-achievement in respect of schedule requirement i in time period j.

Model Coefficients

$a_{ijkl} = 1$ if the l'th possible cutter schedule for cutter k contributes to schedule requirement i in time period j.

$= 0$ otherwise.

r_{ij} : target/limit/threshold for schedule requirement i in time period j in terms of number of cutters contributing to the schedule requirement.

$w^-_{ij} (\geq 0)$: Penalty for each unit of under-achievement in respect of r_{ij}

$w^+_{ij} (\geq 0)$: Penalty for each unit of over-achievement in respect of r_{ij}

Model

$$Min \sum_{i=1}^{ng} \sum_{j=1}^{nt} (w^-_{ij} u_{ij} + w^+_{ij} o_{ij})$$

subject to

$$\sum_{k=1}^{nc} \sum_{l=1}^{n_k} a_{ijkl} x_{kl} + u_{ij} - o_{ij} = r_{ij} \quad \begin{matrix} i = 1,2,\ldots ng \\ j = 1,2,\ldots nt \end{matrix}$$

$$\sum_{l=1}^{n_k} x_{kl} = 1 \qquad k = 1,2,\ldots nc$$

$$x_{kl} \in [0,1] \quad \begin{matrix} k = 1,2,\ldots nc \\ k = 1,2,\ldots n_k \end{matrix}$$

Remarks on the Model

(1) The model is stated in a generic form. In any given application instance, simplifications may be possible. For example, if schedule requirement i is such that r_{ij} represents a desired lower limit then the penalty for over-achievement, w^+_{ij}, can be set at zero and o_{ij} considered as a logical variable. Similarly, if schedule requirement i is such that r_{ij} represents a desired upper limit then the

penalty for under-achievement, w_{ij} can be set at zero and u_{ij} considered as a logical variable.

(2) The number of time periods in the scheduling year is a matter for judgement in relation to the individual application. A day-to-day schedule covering one year is required. Thus in its simplest form, nt equals 365 or 366. However, model size and hence solution time can be reduced by considering a larger time unit. In certain cases this can be achieved without loss of model validity. For example, if the duration of tasks and activities is always an integral number of weeks, the problem can be modelled on a week-to-week basis with nt reduced to 53. Even if the duration of tasks and activities is not always an integral number of weeks it may still be worthwhile to adopt the time unit change in order to obtain solutions more quickly with a possible sacrifice on solution quality. This aspect is investigated in section 3.

(3) The cutter scheduling problem reported here has the following size parameters. There are 30 cutters to be scheduled. There are 9 sets of schedule requirements, 5 of which represent desired minimum levels with the remaining 4 representing desired upper limits. The minimum levels/upper limits are invariant through time and range from a value of 1 to a value of 5.

III. TIME UNIT COMPRESSION

A major factor influencing the difficulty with which the model may be solved is the number of constraints. Each scheduled requirement results in (nt) constraints. As stated in the previous section the most natural form of the model has (nt) equal to the number of days in the year (365 or 366). With 9 (sets of) schedule requirements, the model would have over 3000 rows. In addition the columns are very dense with, typically, 200-250 nonzeros per column. The idea of developing a 'coarser' model in which each time period is increased in size (e.g. from 1 day to 7 days = 1 week) is attractive in significantly reducing the size of the model both in terms of the number of rows and the number of nonzeros. Clearly there may be a reduction in solution quality since the model may be a less precise description of the scheduling problem.

The activities performed by the cutters involve patrolling, maintenance and training. Each of these activities takes place in various forms. For example, the area within which patrolling takes place is divided into various districts, each with its own requirements in terms of cutter coverage. Additionally there are activities such as transit between tasks and necessary time spent in port. With the exception of transit times, the required durations or range of durations of the tasks tend to be specified in terms of an integral number of weeks and, as a consequence, the time unit compression from days into weeks appears more likely to be viable. Some tasks (e.g. training and maintenance) are required to start on a specified day of

the week - but not necessarily the same day of the week. It is for this reason that the weekly model introduces an element of approximation compared to the daily model. Consider the example shown in Figure 1. Suppose that the time unit compression is such that each Monday through Sunday time period of 7 consecutive days is considered as one new time unit. Suppose further that a training task is required to start on a Monday and a maintenance task is required to start on a Wednesday. Then in the example, training takes place in week k since it takes place on every day of week k. However, the maintenance task takes place only for part of week k. In converting from a daily to a weekly model, the question of whether a given task takes place in a given week must be addressed.

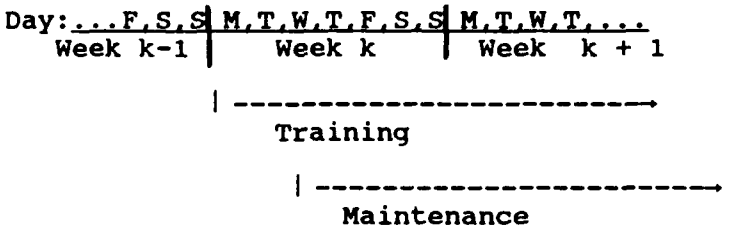


Figure 1: Task start days example

The proposed model treats this issue conservatively such that a feasible weekly schedule, when converted back, will necessarily yield a feasible daily schedule. To achieve this, the treatment depends on the type of constraint considered. The schedule requirements are specified in terms of desired upper limits (implied less than or equal to constraints) or desired lower limits (implied greater than or equal to constraints). In the former, over-achievement is penalised whilst in the latter, under-achievement is penalised.

The constraint coefficients of the weekly model are therefore determined as follows:-

If the constraint group (schedule requirement) i is a 'desired upper limit' type constraint then

$a_{ijkl} = 1$ if the l'th possible (daily) schedule for cutter k covers all 7 days of week j

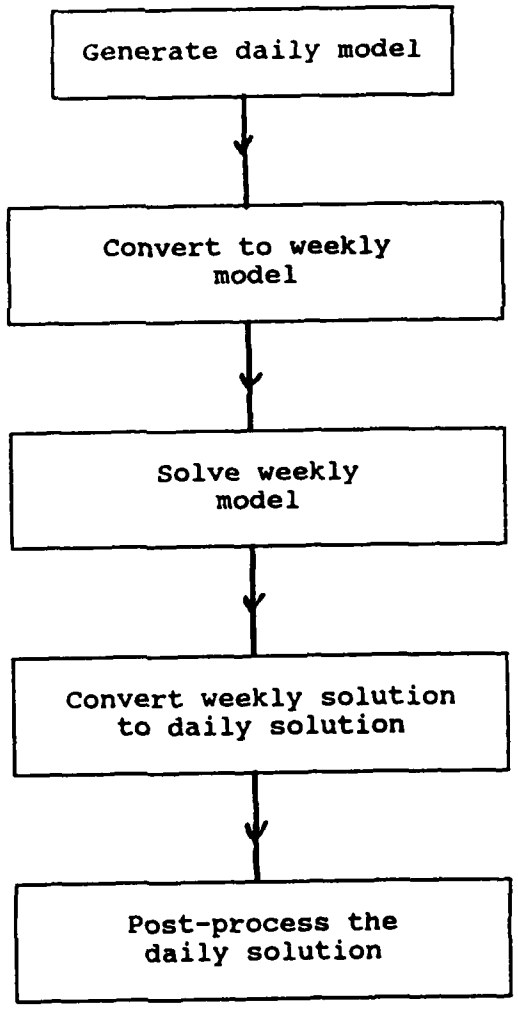
$= 0$ otherwise

If the constraint group (schedule requirement) i is a 'desired lower limit' type constraint then

$a_{jkl} = 1$ if the l'th possible (daily) schedule for cutter k covers <u>any day</u> of week j.

$= 0$ otherwise

The overall modelling strategy is illustrated in Figure 2. The daily model is generated and converted to a weekly model as described above. The weekly model is then solved and the solution in terms of a weekly schedule for each cutter is obtained. These weekly schedules are matched with the schedules of the daily model to obtain daily schedules which are then post-processed. The post-processing performs a series of quick local optimisations which apply daily time shifts to patrolling tasks whenever such shifts lead to local improvements



<u>Figure 2: Overall Modelling Strategy</u>

IV. SUMMARY

A practical model for determining an annual day-by-day schedule for a fleet of United States Coast Guard cutters has been developed. The model was made computationally more tractable by considering a smaller number of larger time periods. Results (to be reported) indicate that little is sacrificed in terms of solution quality by adopting this form of time unit compression.

In common with many scheduling problems, the simultaneous satisfaction of all requirements may not be possible. The use of what is essentially an integer goal programming model ensures that model feasibility is assured. The solution is either a completely satisfactory schedule or as near to one as is possible.

The model lies at the heart of a decision support system that is in the process of being implemented by the United States Coast Guard.

V. REFERENCES

1. Darby-Dowman, K. and Mitra, G., 'An Extension of Set Partitioning with Application to Scheduling Problems', European Journal of Operational Research, Vol.21, pp 200-205, 1985.

2. Marsten,R.E. and Shepardson,F., 'Exact Solution of Crew Scheduling Problems using the Set Partitioning Model: Recent Successful Applications', Networks Vol.11, pp 165-177, 1981.

REPRESENTING PROCEDURAL KNOWLEDGE WITHIN MATHEMATICAL PROGRAMMING MODELLING SYSTEM (MPL)

Ken Darby-Dowman*
Bjarni Kristjansson**
Cormac Lucas*
Gautam Mitra*
Shirley-Anne Moody*

* Department of Mathematics and Statistics,
Brunel, The University of West London,
Uxbridge, Middlesex, UB8 3PH, England

** Maximal Software Ltd,
Klappara 11, 15-110 Reykjavik, Iceland

1. INTRODUCTION

Languages for representing Linear Programming Models for optimization are well established. These languages follow a simple algebraic structure to represent the linear form restrictions and are adequate for a vast range of LP models.

There are many experimental and commercial systems of this genre which are used by industry, for an uptodate review of such systems the reader is referred to (Steiger and Sharda,91), (Greenberg,91). Most modern modelling systems enable the modeller to specify models in a declarative algebraic language. A set of algebraic statements in a modelling language both specifies and documents a model, whereas the generation of a machine readable constraint matrix takes place in the background.

It is now increasingly realized that alternative modelling paradigms such as database modelling, mathematical programming modelling, simulation, logic programming and programs written in a high level computer language are essentially different forms of knowledge representations as perceived by the AI Community (Geoffrion, 1990), (Mitra, 1989). Knowledge expressed in a declarative form and knowledge specified in a procedural form are two main approaches to knowledge representation.

In this paper we first identify an important deficiency of many known Mathematical programming modelling languages. These languages are well designed to represent large classes of LP and IP models in the declarative form. A wide class of other optimization models applying to many real problems such as crew scheduling, cutting stock, VLSI routing and ship scheduling provide instances of models which are highly dependent on domain knowledge. For these models the domain knowledge concerning the rules of crew duties, alternative ways of defining cutting patterns, possible minimum cost routes, a set of tasks around calendar dates can be only specified in a procedural form. Although modelling systems are well set out to structure the model components, by their very nature these modelling systems lack

the procedural constructs. As a result activity based LP models which are constructed by a column generation strategy across a fixed structure of rows, cannot be developed within these systems.

We have introduced extensions to an established LP modelling system namely MPL whereby the procedural knowledge is introduced through a dynamic binding of the high level modelling language. We also introduce object orientation thus taking advantage of encapsulation, inheritance, and information hiding. In this way we can capture procedural knowledge in the form of methods within self contained objects. The extension is illustrated by an example of an optimum cutter scheduling problem studied by the authors.

ACTIVITY BASED LP/IP MODELS

As explained in the introduction many practical models are such that the underlying LP can only be constructed if the activities specifying the technology matrix, that is, the columns, are computed using the domain knowledge of the application. We consider a few examples.

Crew Scheduling: Both air crew (Johnson (1990)) and bus crew scheduling (Darby-Dowman and Mitra, 1985) problems are by nature, set partitioning or set covering problems or their extensions. The rows represent legs of flight or pieces of work which must be covered by the crew. The columns represent ways of carrying out a work shift that is legal within union regulations and accepted practises.

VLSI Routing: VLSI routing (Pulleyblank,1992) is a well known combinatorial problem which has the structure of a set problem in which columns are generated after solving a travelling salesman problem.

Cutting Stock Problem: One and two dimensional cutting stock problems (Gomory,1965) have many industrial applications in the area of minimum wastage of sheet material. Here again the columns are constructed by a combinatorial procedure for fitting patterns within a two dimensional master area. Alternatively by solving a knapsack problem using dynamic programming recursion efficient patterns and corresponding columns can be generated.

Cutter Scheduling Problem: We have developed a model (Darby-Dowman,1992) which uses integer goal programming in respect of a set problem with some additional choice constraints. The problem is set out in the next section.

2. STATEMENT OF THE CUTTER SCHEDULING PROBLEM

The problem involves creating an annual schedule for nc number of cutters to carry out tasks such as Patrol, Maintenance, training whereby the schedule specifies the activity of each cutter for each day of the year.

Define the parameters

nc = number of cutters to be scheduled,

n_k = number of columns (possible schedules) for cutter k,

nt = number of time periods in the scheduling year,

ng = number of constraint groups (schedule 'requirements'),

The corresponding index sets

i : i = 1,2,...,ng; schedule requirements,

t : t = 1,2,...,nt; time period,

k : k = 1,2,...,nc; cutter identifier

ℓ : ℓ = $1,2,\ldots,n_k$; identifier of possible schedule for cutter k,

Model Variables

$x_{k\ell}$ = 1 if the ℓ^{th} possible schedule for cutter k is selected 0 otherwise,

u_{it}: extent of the under-achievement in respect of schedule requirement i in time period t,

o_{it}: extent of the over-achievement in respect of schedule requirement i in time period t,

Model Coefficients

$a_{itk\ell}$ = 1 if the ℓ^{th} possible cutter schedule for cutter k contributes to schedule requirement i in time period t, 0 otherwise,

r_{it}: target/limit/threshold for schedule requirement i in time period t in terms of number of cutters contributing to the schedule requirement,

$w^-_{it}(\geq 0)$: penalty for each unit of under-achievement in respect of r_{it},

$w^+_{it}(\geq 0)$: penalty for each unit of over-achievement in respect of r_{it}

The model is stated as,

$$Min \sum_{i=1}^{ng} \sum_{t=1}^{nt} (w^-_{it}u_{it} + w^+_{it}o_{it})$$

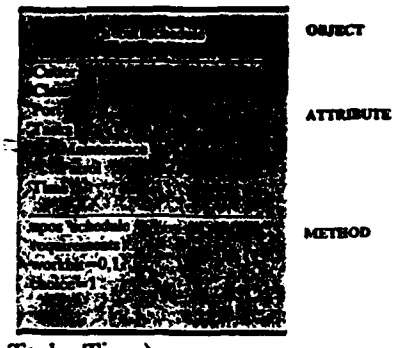
$$subject\ to$$

$$\sum_{k=1}^{nc} \sum_{l=1}^{n_k} a_{itkl}x_{kl} + u_{it} - o_{it} = r_{it} \quad \begin{matrix} i = 1,2,\ldots,ng \\ t = 1,2,\ldots,nt \end{matrix}$$

$$\sum_{l=1}^{n_k} x_{kl} = 1 \qquad k = 1,2,\ldots,nc$$

$$x_{kl} = 0 \text{ or } 1 \qquad \begin{matrix} k = 1,2,\ldots,nc \\ l = 1,2,\ldots,n_k \end{matrix}$$

3. COLUMN GENERATION PSEUDO CODE



a_cut_schedule (Cutter,Class,Port,Tasks,Time)

: do for all cutters.

Analyse DATES

: do for task group one

/maintenance and training tasks/

: go to task order (j),{j = 1,2,3}

task order (1) : allocate maintenance to this cutter group
go to loop end

task order (2) : allocate reftraining to this cutter group
go to loop end

task order (3) : allocate training availability to this cutter group
go to loop end

loopend : endo

: do for allowable patrol tasks

/patrol tasks/

patrol order (1) : allocate patrol
go to patrol end

....

patrol order (4) : allocate patrol

patrol end : endo

cutters end : endo

4. EXTENDED MPL SYNTAX

MPL is a modelling language for specifying linear programming problems. As in any other algebraic LP modelling language the model can be specified by progressively introducing a series of keywords which divide the model components across sections. The syntax and structure of MPL is set out below in a summary form, (Maximal, 1991)

The Definition Part

TITLE	- Names the problem
INDEX	- The dimension of the problem
DATA	- Scalars, datavectors and datafiles
DECISION	- Define vector variables
MACRO	- Reusable macros for expressions

The Model Part

MODEL	- Description of the problem
MAX or MIN	- The objective function
SUBJECT TO	- The constraints
BOUNDS	- Simple upper and lower bounds
FREE	- Free variables
INTEGER	- Integer variables
BINARY	- Binary (0/1) variables
END	

TYPICAL LINEAR FORM SYNTAX

SUM (<index> : <table ref>*<decision variable>)
<relation><table ref>

SCHEDULING MODEL SPECIFIED IN EXTENDED MPL

```
TITLE
    csap_schedule
INDEX
    nmbcutters = 1..30;
#DYNAMIC npos (nmbcutters)
    notimeperiod = 1..53;
    notasks = 1..9;
    patrol = (D1,D3,D5,D7);
    maintenance = (drydock, dockside);
DATA
    tasklimits[patrol] : =          DATAFILE (tlimpat.dbs);
      •
      •
      •
    oversatcost[notasks,notimeperiod]: =DATAFILE(ostcost.dbs);
    undersatcost[notasks,notimeperiod]: = DATAFILE(ustcost.dbs);
    coverreqmt[notasks,notimeperiod]: = DATAFILE(cover.dbs)

DECISION VARIABLES

    oversat[notasks,notimeperiod];
    undersat[notasks,notimeperiod]
#DYNAMIC xschedule[npos(nmbcutters)];
```

```
MODEL
    MIN deviation = SUM(notasks,notimeperiod: oversat*oversatcost +
                undersat*undersatcost);

SUBJECT TO
    COVER[notasks,notimeperiod]:
    SUM(nmbcutter,npos:a_cut_schedule*xschedule)
      + undersat - oversat = coverreqmt;

    CHOOSE 1[nmbcutters]: SUM(npos:xschedule) = 1;

BINARY
    xschedule;

END
```

6. REFERENCES

Darby-Dowman, K. and Mitra, G.,(1985) "Cru-Sched - A computer based bus crew scheuling system using integer programming" in Transit Vehicle and Crew Scheduling, Editor J.M. Rousseau, North Holland.

Geoffrion, A.M. (1992), "The SML language for structured modeling", Operations Research 40:1 pp 38-75.

Greenberg, Harvey,J. (1991), "A comparison of mathematical programming modelling systems", Mathematics Department, University of Colorado at Denver.

Maximal Software (1991), MPL Modelling System Release 2 User Manual, Maximal Software.

Mitra,G. (1988), "Models of decision making: An overview of problems, tools and major issues, in Mathematical Models for Decision Support, NATO ASI, series F, Vol.48, Springer Verlag, 17-53.

Johnson,E.,(1991) "Airline applications and mixed integer programming", presented to IBM Europe Institute, Oberlech, Austria, July.

Steiger,D. and Sharda,R. (1991), "LP modelling languages for personal computers: A comparison", Oklahoma State University; also to appear in Annals of OR, Applied Mathematical Programming and Modelling, I. Maros and G. Mitra (eds), Baltzer Press.

Pulleyblank,W.R.,(1992), "Solving global routing problems using linear programming", presented to IBM Europe Institute on Optimization Solutions, Oberlech, Austria.

Gomory,R. and Gilmore, P.C., (1965), "Multistage cutting stock problems of two and more dimensions", in Operations Research, p94-120.

Lower and Upper Bounds for the Single Machine Scheduling Problem with Earliness and Flow Time Penalties

Mauro Dell'Amico
Dipartimento di Elettronica, Politecnico di Milano, Italy

Silvano Martello
Dipartimento di Informatica, Università di Torino, Italy

Daniele Vigo
Dipartimento di Elettronica, Informatica e Sistemistica, Università di Bologna, Italy

1 Introduction

Given a machine which can process at most one task at time, and a set $T = \{T_1, \ldots, T_n\}$ of n tasks with associated processing times $p_1, \ldots, p_n$, deadlines $d_1, \ldots, d_n$, flow time penalties $\alpha_1, \ldots, \alpha_n$, and earliness penalties $\beta_1, \ldots, \beta_n$, the *Single Machine Scheduling Problem with Earliness and Flow Time Penalties* ($SMEF$) is to determine a processing sequence for the tasks that minimizes the total cost incurred by the penalties, while preserving deadline requirements of each task. The processing cost associated with each task T_j is equal to its completion time C_j multiplied by the flow time penalty, plus its earliness $E_j = d_j - C_j$ multiplied by the earliness penalty. Using the three-field classification introduced in Graham, Lawler, Lenstra and Rinnooy Kan [7], the problem is denoted as $1|d_j|\sum(\alpha_j C_j + \beta_j E_j)$.

We assume that processing times, deadlines and penalties are positive integers, that tasks are available at time zero, that setup times, if any, are identical and included in the processing time and that preemption of tasks is not allowed. A schedule (i.e. a solution for problem $SMEF$) is defined through the vector $(C_1, C_2, \ldots, C_n)$ of the completion time of the tasks: task T_j is processed in time interval $(C_j - p_j, C_j]$

The flow time penalty has classically been used to model overhead and capital carrying costs sustained during production, while the earliness penalty takes into account the cost incurred for storing a finished product until it is shipped.

The problem is strongly NP-hard, since it is a generalization of the single machine scheduling problem calling for the minimum weighted flow time sequence with no tardy

task $(1|d_j|\sum \alpha_j C_j)$ which is known to be NP-hard in the strong sense (see Lenstra, Rinnooy Kan and Brucker [8]). An exact algorithm for $SMEF$, based on a dynamic programming approach, has been developed by Bard, Venkatraman and Feo [2]. Feo, Venkatraman and Bard [4] have recently presented a heuristic algorithm based on a Greedy Randomized Adaptive Search Procedure (GRASP). Special cases and related problems have also been studied by Fry and Leong [6], Bagchi and Ahmadi [1], Faaland and Schmitt [5], and Sen, Raizadeh and Dileepan [10].

In the following sections we develop lower bounds and an approximation algorithm for $SMEF$ and show, through computational experiments, the effectiveness of the proposed approaches. In Section 2 we present some simple lower bounds and a better one based on a preemptive relaxation of the problem. In Section 3 we use the preemptive lower bound to obtain an approximation algorithm. The approximation algorithm is experimentally analyzed in Section 4.

Unless otherwise specified, we will always assume that the tasks are numbered so that:

$$d_1 \leq d_2 \leq \ldots \leq d_n. \tag{1}$$

2 Lower bounds

The objective function of $SMEF$ can be written as:

$$\begin{aligned} z(SMEF) &= \min \sum_{j=1}^{n} (\alpha_j C_j + \beta_j (d_j - C_j) = \sum_{j=1}^{n} \beta_j d_j + \min \sum_{j=1}^{n} w_j C_j = \\ &= \sum_{j=1}^{n} \beta_j d_j + z(SMEF'); \end{aligned} \tag{2}$$

where $w_j = \alpha_j - \beta_j$ is the overall penalty of task T_j.

2.1 A simple bound

We can partition T into $TR = \{T_j \in T : w_j < 0\}$ and $TL = \{T_j \in T : w_j \geq 0\}$. These two subsets contain tasks that have different behaviour in an optimal schedule: the tasks of set TR require to be processed as late as possible, while those of set TL must be scheduled as soon as possible.

Let P_R (resp. P_L) denote the sub-instance of $SMEF'$ in which only the tasks in TR (resp. TL) are considered, and let $z(P_R)$ (resp. $z(P_L)$) denote the corresponding solution value. We will consider the relaxation of $SMEF'$ obtained by assuming that a task in TR can be processed in parallel with a task in TL: the optimal solution to this problem is

clearly provided by the separate solutions to P_R and P_L, so $L = \sum_{j=1}^{n} \beta_j d_j + z(P_R) + z(P_L)$ is a lower bound on $z(SMEF)$.

Since any instance of $1|r_j|\sum w_j C_j$, which is known to be strongly NP-hard (see Lenstra, Rinnooy Kan and Brucker [8]), can be easily transformed into an equivalent instance of P_R, we know that this problem too is strongly NP-hard. Problem P_L is the already mentioned $1|d_j|\sum w_j C_j$, which is also known to be NP-hard in the strong sense. Hence above lower bound L cannot be computed in polynomial time, but we can determine lower bounds $L(P_R)$ and $L(P_L)$ for the two subproblems, obtaining lower bound $L = \sum_{j=1}^{n} \beta_j d_j + L(P_R) + L(P_L)$.

A lower bound for problem P_R can be obtained by allowing that more than one task of TR can be processed at a time. The optimal solution is clearly obtained in $O(n)$ time by scheduling each task as late as possible, i.e. setting $C_j = d_j$ for each $Tj \in TR$, and its value is:

$$L_0(P_R) = \sum_{T_j \in TR} w_j d_j. \tag{3}$$

A lower bound for problem P_L can be computed by relaxing the deadline constraints, obtaining the problem $1||\sum w_j C_j$ which can be exactly solved (see Smith [11]) in $O(n \log n)$ time by scheduling the tasks in order of decreasing value of the ratio w_j/p_j: let $L_0(P_L)$ denote the solution value. Then

$$L_0 = \sum_{j=1}^{n} \beta_j d_j + L_0(P_R) + L_0(P_L), \tag{4}$$

is a valid lower bound for $SMEF$.

The time complexity for the computation of L_0 is clearly $O(n \log n)$.

2.2 A new lower bound

Let us consider the following new problem, called $S(SMEF')$ in the sequel, derived from $SMEF'$ by allowing that each task T_j can be split into $k(j)$ *pieces* $T_{j_1}, \ldots, T_{j_{k(j)}}$ with deadlines $d_{j_i} = d_j$ for each i and j, positive processing times $p_{j_1}, \ldots, p_{j_{k(j)}}$ such that $\sum_{i=1}^{k(j)} p_{j_i} = p_j$ for each j, and weights $w_{j_1}, \ldots, w_{j_{k(j)}}$ having the same sign as w_j and such that $\sum_{i=1}^{k(j)} w_{j_i} = w_j$ for each j.
Let C_{j_i} be the completion time of piece T_{j_i}: the objective function of $S(SMEF')$ is

$$\bar{z} = \sum_{j=1}^{n} \sum_{i=1}^{k(j)} w_{j_i} C_{j_i}. \tag{5}$$

Posner [9] proves (for the case $w_j \geq 0$, but the proof holds also for unrestricted w_j) that, given a feasible solution to $SMEF'$, of value z, the corresponding solution for

$S(SMEF')$ (obtained by consecutively scheduling $T_{j_1}, \ldots, T_{j_{k(j)}}$ in time interval $(C_j - p_j, C_j]$) has value $\overline{z}$ such that $z = \overline{z} + CBRK$ where

$$CBRK = \sum_{j=1}^{n} \sum_{i=1}^{k(j)} w_{j_i} \sum_{h=i+1}^{k(j)} p_{j_h}. \tag{6}$$

Let $\overline{z^*}$ be the optimal solution value of $S(SMEF')$, then $\overline{z^*} \leq \overline{z}$, hence

$$L_1 = \sum_{j=1}^{n} \beta_j d_j + \overline{z^*} + CBRK \tag{7}$$

is a valid lower bound on the value of $z(SMEF)$.

Given a feasible task splitting, we can partition the set of pieces of $S(SMEF')$ into $T^+ = \{T_{j_i} : w_{j_i} \geq 0\}$ and $T^- = \{T_{j_i} : w_{j_i} < 0\}$. Let n^+ and n^- be the cardinalities of these two subsets, were obviously we have $n^+ + n^- = \sum_{j=1}^{n} k(j)$. Let us also rename the pieces in such a way that

$$T^+ = \{T_1^+, T_2^+, \ldots, T_{n^+}^+\}, \quad T^- = \{T_1^-, T_2^-, \ldots, T_{n^-}^-\}$$

(with d_j^+, d_j^-, p_j^+, p_j^-, w_j^+, w_j^- renamed accordingly) and that

$$d_j^+ \leq d_{j+1}^+; \quad d_j^- \leq d_{j+1}^-. \tag{8}$$

Definition 1 *Given set T^-, a* block *is a set $B_i = \{T_{a_i}^-, T_{a_i+1}^-, \ldots, T_{b_i}^-\}$ of consecutive pieces (ordered according to (8)), whose total processing time is not grater than the time interval between the deadline of $T_{a_i-1}^-$, and the deadline of $T_{b_i}^-$. Let $s_i = d_{b_i} - \sum_{j=a_i}^{b_i} p_j$: the associated* block interval *is $BI_i = (s_i, d_{b_i}]$.*

Let us define

$$\tau = \min\{t : \sum_{j=1}^{n^+} p_j^+ + \sum_{T_j^- : d_j^- \leq t} p_j^- \leq s_i \text{ for each } i : d_{b_i} > t\}, \tag{9}$$

and note that τ must be the completion time of a piece belonging to T^+. Then we can divide problem $S(SMEF')$ in two subproblems:

P_A: $S(SMEF')$ for the pieces in $T_A = T^+ \cup \{T_j^- : d_j^- < \tau\}$,

P_B: $S(SMEF')$ for the pieces in $T_B = \{T_j^- : d_j^- > \tau\}$,

and observe that $\sum_{T_j \in P_A} p_j = \tau$.

Without loss of generality, let us assume from now that $p_{j_i} = 1$ for each j_i.

Theorem 1 *The separate optimal solutions to P_A and P_B do not overlap and produce the optimal solution to $S(SMEF')$.*

Proof: the pieces in T_A must obviously be scheduled before τ. Observe that in any optimal solution to $S(SMEF')$ the pieces in T_B must be scheduled after τ. Assume indeed that a unary piece $T_j^- \in T_B$ is scheduled before τ: by definition of τ, a piece $T_k^+ \in T_A$ must be scheduled after τ, so interchanging the two units the solution would improve. Hence the thesis, since scheduling any piece of T^+ after τ would leave a useless idle time before τ. □

Problem P_B it is equivalent to $1|r_j|\sum v_jC_j$, with tasks allowed to be splitted as described above, and $v_j = -w_j$, $r_j = \max_k\{d_k\} - d_j$. This problem is exactly solved by the algorithm proposed by Belouadah, Posner and Potts [3].

Theorem 2 *In the optimal solution to P_A any unit $T_j^- \in T_A$ is scheduled in the block interval associated with the block containing T_j^-.*

Proof: we prove the theorem by absurd. Let B_i be the rightmost block not containing a unit $\hat{T}^-$ belonging to the block, and observe that such $\hat{T}^-$ must be scheduled at a time instant preceding BI_i. Since in any optimal solution, no idle time can exist between 0 and τ, B_i must contain at least one unit of T^+ (indeed no unit of T^- could come from a block on its right, by definition of B_i, nor from a block on its left, since this would violate the deadline): let $\tilde{T}^+$ denote the rightmost such unit, scheduled at $\tilde{t} \in BI_i$. Find the rightmost unit $\tilde{T}^-$, scheduled at a time instant preceding $\tilde{t}$, which can be scheduled in $\tilde{t}$ (note that such unit must exist, since, by definition, a block interval can be filled by units of T^- with no idle time). Interchanging $\tilde{T}^+$ and $\tilde{T}^-$, we would improve the solution, a contradiction. Further observe that, if $\tilde{T}^- \neq \hat{T}^-$, the process can be iterated until $\hat{T}^-$ is moved to BI_i. □

Corollary 1 *Problem P_A decomposes into: (i) the problem P_A^- of optimally scheduling the pieces belonging to T^- of each block of P_A in the associated block interval; and (ii) the problem P_A^+ of optimally scheduling the pieces belonging to T^+ in intervals $(0,\tau]\backslash\cup_i\{BI_i : d_{b_i} \leq \tau\}$.*

Problem P_A^- can be optimally solved applying the algorithm of Belouadah, Posner and Potts [3] to the tasks $T_j \in TR$ such that the corresponding pieces are in T_A. Problem P_A^+ can be optimally solved applying the Posner algorithm to an instance defined by the tasks $T_j \in T_L$, plus a number of dummy tasks $\hat{T}_i^-$, one for each block B_i with $d_{b_i} \leq \tau$, having deadlines d_{b_i}, processing times $(d_{b_i} - s_i)$ and weights $-\epsilon$ (with $\epsilon > 0$). Observe that such algorithm would schedule each dummy task exactly at s_i without splitting it.

Hence, in order to compute lower bound L_1, we should derive from the original problem the three problems P_A^+, P_A^- and P_B, and separately solve them. However, we can obtain a unique $O(n \log n)$ algorithm that determines and solves these problems at a time, by modifying the Posner algorithm. The Posner algorithm for $1|d_j|\sum w_jC_j$ with

task splitting starts with a time instant $t = \sum_{j=1}^{n} p_j$, and schedules, by decreasing time instants, tasks or pieces in $[0, t]$. This algorithm, which works also for negative weights, can directly solve both P_A^+ and P_A^- if applied to P_A with starting time $t = \tau$. To solve P_B, it is sufficient to observe that the algorithm proposed by Belouadah, Posner and Potts for the problem obtained from P_B by replacing v_j with $-w_j$ and r_j with $\max_k\{d_k\} - d_j$, is equivalent to iteratively apply the Posner algorithm to the tasks of each block BI_i, starting from the rightmost one. When a block is completely scheduled, if the sum of the processing times of the unscheduled tasks is greater than the ending time of the next block, then this is the first block of P_A, and problem P_B is optimally solved.

3 Approximation algorithm

In this section we introduce an approximation algorithm for $SMEF$ based on lower bound L_1. The algorithm determines a feasible schedule for problems $SMEF$, starting from the optimal solutions of problem $S(SMEF')$. Given the optimal solution to $S(SMEF')$, and observing that this can be infeasible for $SMEF$ only because of the splitted tasks, we can easily obtain a feasible sequence as follows. We start with $t = \max_i\{d_{b_i}\}$ and proceed by decreasing completion times until we encounter a piece T_{j_a} obtained by splitting a task (or a piece) T_j into T_{j_a} and T_{j_b}, with processing times p_{j_a} and p_{j_b}, respectively, scheduled at time instants $t \equiv t_a$ and t_b, with $t_a > t_b + p_{j_b}$. We can eliminate this infeasibility in three possible ways;

a) scheduling T_{j_b} at time instant $t_a - p_{j_b}$ and shifting left, of p_{j_b} time units, all the tasks currently scheduled between T_{j_a}, and T_{j_b};

b) scheduling T_{j_a} at time instant $t_b - p_{j_a}$ and shifting left the necessary tasks preceding T_{j_b}, until an idle time interval of length at least p_{j_a} is created, if the corresponding schedule is feasible;

c) scheduling T_{j_a} at time instant $t_b + p_{j_b}$ and shifting right the tasks between T_{j_b} and T_{j_a}, of p_{j_a} time units, if the corresponding schedule is feasible.

Whenever a piece is encountered, the algorithm evaluates all these alternatives and selects the one producing the minimum objective function increase.

The final approximate solution to $SMEF$ is then obtained by optimally inserting idle times. This can be done through the $O(n)$ procedure described in Bard, Venkatraman and Feo [2].

4 Computational Results

We have coded in C language the lower bound L_1 described in Section 2, the *GRASP* heuristic described in Feo, Venkatraman and Bard [4] and the approximation algorithm of Section 3.

We executed computational experiments on a PC 486 with a 33 MHz clock, by considering problems as those described in Feo, Venkatraman and Bard [4].

For each task T_j the associated values of p_j are uniformly random in range $[1, 10]$.

For each value of n, $(n = 10, 15, 20, 25, 30)$, three classes of random test problems are defined:

I) $\alpha_j \leq \beta_j$ for approximately 50% of the tasks;

II) $\alpha_j \leq \beta_j$ for approximately 66% of the tasks;

III) $\alpha_j \leq \beta_j$ for approximately 33% of the tasks;

where both $\alpha_j, \beta_j \in [1, 10]$.

The deadline of each task T_j is uniformly random in range $[\beta^- \sum_{j=1}^{n} p_j, \beta^+ \sum_{j=1}^{n} p_j]$, with the following (β^-, β^+) pairs: (0.75, 1,75), (0.25, 1.75), (0.75, 1.75), (0.50, 2.50), (0.25, 1.25), (0, 1.25), and (0, 1).

For each class, for each value of n and for each pair (β^-, β^+) ten feasible problem were generated, giving a total of 350 instances.

The computational experiments have proved that the algorithm of Section 3 produces solutions better than those produced by *GRASP*, with running times which are up to two orders of magnitude smaller.

References

[1] U. Bagchi and R. Ahmadi (1987). "An Improved Lower Bound for Minimizing Weighted Completion Times with Deadlines", *Operations Research* 35, 311–314.

[2] J. Bard, K. Venkatraman and T. Feo (1989). "A Dynamic Programming Approach to a Single Machine Scheduling Problem", *Technical Report, Operations Research Group, Mechanical Engineering Department, University of Texas at Austin.*

[3] H. Belouadah, M. E. Posner and C. N. Potts (1992). "Scheduling with Release Dates on a Single Machine to Minimize Total Weighted Completion Time", *Discrete Applied Mathematics* 36, 213–231.

[4] T. Feo, K. Venkatraman and J. Bard (1991). "A GRASP for the solution of a Hard Single Machine Scheduling Problem", *Technical Report, Operations Research Group, Mechanical Engineering Department, University of Texas at Austin.*

[5] B. Faaland and T. Schmitt (1987). "Scheduling Tasks with Due Dates in a Fabrication/Assembly Process", *Operations Research* 35, 378–388.

[6] T. Fry and G. Leong (1987). "A Bi-Criterion Approach to Minimizing Inventory Costs When Early Shipments are Forbidden" *Computers and Operations Research* 14, 363-368.

[7] R.L Graham, E.L. Lawler, J.K. Lenstra and A.H.G. Rinnooy Kan (1979). "Optimization and approximation in deterministic sequencing and scheduling a survey", *Annals of Discrete Mathematics* 5, 287–326.

[8] J.K Lenstra, A. Rinnooy Kan and Brucker (1977). Complexity of Machine Scheduling Problems", *Annals of Discrete Mathematics* 1, 343–362.

[9] M. E. Posner (1985). "Minimizing Weighted Completion Times with Deadlines", *Operations Research* 33, 562–574.

[10] T. Sen, F. Raizadeh and P. Dileepan (1988). "A Branch and Bound Approach to the Bicriterion Problem Involving Total Flow Time and Range of Lateness", *Management Science* 34, 254–260.

[11] W.E. Smith (1956). "Various Optimizers for Single Stage Production", *Naval Research Logistics Quarterly* 3, 59–66.

Combining Genetic and Local Search for Solving the Job Shop Scheduling Problem

Ulrich Dorndorf* Erwin Pesch†

Abstract

This paper describes a genetic algorithm that uses a local search based improvement operator for solving the job shop scheduling problem. The genetic algorithm serves as a meta-heuristic that guides a procedure for building starting solutions which are then improved by local search. Initial computational results are encouraging: The algorithm has solved the famous 10 x 10 problem instance formulated by Fisher and Thompson in 1963 which has defied solution for over 20 years.

1 Introduction

The minimum makespan problem of job shop scheduling (JSP) is a classical combinatorial optimization problem that has received considerable attention in the operations research literature; in the recent years, exact algorithms [6, 3, 5] and tailored approximation methods [2] for the JSP have been significantly progressed. It is well known that the JSP is NP-hard [23] and belongs to the most intractable problems considered. The problem is thus a good test for evaluating the power of generally applicable approximation techniques [1]. The algorithm described here combines two such techniques, genetic and local search. The idea of using problem specific information in form of local search within the framework of a genetic algorithm has been suggested before in a number of publications, see for instance [18, 26, 15, 27, 1, 21, 33, 9]. This paper focusses on the combination of relatively simple building blocks rather than on fine-tuning the inidividual parts; for instance, more intricate local search neighbourhood structures than the one employed here are known [22, 8].

The remainder of this paper is organised as follows. After a short introduction to the JSP in the next section, section 3 presents a simple variable depth search improvement heuristic; we assume that the reader is familiar with the concepts of local search. Section 4 describes the genetic framework in which this procedure operates. We conclude with a description of initial computational results.

2 The Job Shop Scheduling Problem

A job shop consists of a set M of m different machines that perform operations on a set J of jobs. Each job has a specified processing order through the machines, i.e. it is an ordered list of operations from set $V = \{1, \ldots, n\}$. An operation is characterized by the machine it requires and by its processing time. Operations cannot be interrupted (non-preemption), each machine can handle only one job at a time, and each job can only be performed on one machine at a time. The problem is to find operation sequences on the machines which minimize the makespan, the maximum of the completion times of all operations.

*INFORM — Institut für Operations Research und Management GmbH, Pascalstr. 23, D-5100 Aachen, F.R.G., e-mail: uli%inform.uucp@germany.eu.net

†Faculty of Economics and Business Administration, University of Limburg, P.O. Box 616, NL-6200 MD Maastricht, The Netherlands, e-mail: efkwepes@hmarl5.bitnet

An illuminating description of the problem is the disjunctive graph model introduced by Roy and Sussman [31]. In the edge-weighted graph, there is a vertex for every operation $i \in V$ and two dummy vertices 0 and $n+1$ representing the start and end of a schedule. For every two consecutive operations of the same job there is a directed arc from the arc set A between the corresponding vertices; the start and end vertices 0 and $n+1$ are considered to be the first and last operation, respectively, of every job.

For each machine $k \in M$ the edge set E_k contains all pairs $\{i, j\}$ of operations to be performed on k. Because these operations cannot overlap, an orientation of the disjunctive edges in E_k must be chosen: an operation i either has to be performed before j (choose the orientation (i, j)) or after j (choose (j, i)). A solution to the JSP (a schedule) can be seen as an orientation of all disjunctive edges in $E = \{E_1, \ldots, E_m\}$ such that the resulting Graph $G(V, A, E)$ is acyclic, i.e. there are no precedence conflicts between operations.

Each arc or oriented disjunctive edge (i, j) in G is labeled with a weight corresponding to the processing time of the operation (vertex) i from which the arc/edge starts. The earliest starting time t_i of an operation i in a schedule is equal to the length of the maximum weight or longest path in G from the start node 0 to vertex i; the makespan of the schedule is equal to the length of the maximimum weight or critical path from start node 0 to end node $n+1$.

3 A Local Search Procedure for the JSP

Most neighbourhood structures that have been employed in local search algorithms for the JSP can be considered to be based on an idea used in one of the first exact solution methods for the problem due to Balas [4]. His implicit enumeration algorithm makes use of the fact that in every schedule with a makespan shorter than the one of the current schedule, at least one of the disjunctive edges on the critical path of the current solution graph must be reversed. Reversing an edge on the critical path of a directed graph always yields an acyclic graph [4], in other words a new feasible solution without precedence conflicts between operations. These observations suggest the following neighbourhood structure:

> The neighbourhood $N(x)$ of a solution x characterized by the solution graph G_x is the set of solutions y with a solution graph G_y that can be obtained from G_x by reversing the orientation of a disjunctive edge (i, j) on the critical path of G_x, i.e. by replacing (i, j) with (j, i).

Reversing the edge (i, j) means changing the order in which i and j are processed on a machine. This neighbourhood is *connective* [4, 22]: It is possible to transform an arbitrary solution into every other solution, including the optimal one, by going through a sequence of *moves* in the neighbourhood, in other words by iteratively replacing a current solution x with one of its neighbours $y \in N(x)$.

In order to use the neighbourhood in a local search algorithm, a gain must be associated with every move. The gain $g(i, j)$ from reversing an edge (i, j) can be estimated based on considerations about the minimal length of the critical path of the resulting graph (finding the exact gain of a move would generally involve a longest path calculation). The gain of a move can be negative, thus leading to a deterioration of the objective function. For details, we refer to [4].

The simple neighbourhood structure described above has been extended by Matsuo et al. [25] and Dell'Amico and Trubian [8], see also [32, 1, 22].

In the remainder of this section, we present a local search procedure that uses the neighbourhood defined above. The algorithm is based on a technique described by Kernighan and Lin [20, 24], which has later been named 'variable depth search' by Papadimitriou and Steiglitz [29]. The method can be seen as a special case of a more general approach introduced by Glover [14]. The basic idea is similar to the one used in tabu search [12, 13], the main difference being that the list of forbidden (tabu) moves grows dynamically during a variable depth search iteration and is reset at the beginning of the next iteration.

Figure 1: *A variable depth search algorithm*

start with an initial solution x^*;
repeat
  $T := \emptyset$; {T is the tabu list}
  $d := 0$; {d is the current search depth}
  do
    $d := d + 1$,
    find the best move, i.e. the disjunctive edge (i^*, j^*) for which $g(i^*, j^*) = \max \{g(i,j) | (i,j) \in E - T\}$; {note that $g(i^*, j^*)$ can be negative}
    make this move, i.e. reverse the edge (i^*, j^*), thus obtaining the solution x_d at search depth d;
    $T := T + \{(j^*, i^*)\}$;
  while $|T| \neq |E|$;
  let d^* denote the search depth at which the best solution x_{d^*} with $f(x_{d^*}) = \min \{f(x_d) | 0 < d < |E|\}$ has been found;
  if $d^* > 0$ **then**
    $x^* = x_{d^*}$;
until $d^* = 0$;

The algorithm is outlined in figure 1; $f(x)$ is the objective function. Beginning with a starting solution $x^{*(0)}$, the procedure looks ahead for a certain number of moves and then sets the starting solution $x^{*(1)}$ for the next iteration to the best solution found in the look-ahead phase at depth d^*. These steps are repeated as long as an improvement is possible. When the maximal look-ahead depth where the length $|T|$ of the tabu list equals the cardinality $|E|$ of the edge set is reached, every disjunctive edge has been reversed once. The step leading from a starting solution $x^{*(k)}$ in iteration k to the starting solution $x^{*(k+1)}$ in the next iteration consists of a varying number d^* of moves in the neighbourhood, hence the name variable depth search. At the inner level of the 'do while' loop, the algorithm can escape local optima by allowing moves with negative gain. Cycling is avoided via the dynamically growing tabu list T. At the outer level of the 'repeat until' loop, the procedure stops as soon as an iteration without improvement occurs.

As an extension of the algorithm, the outer level ('repeat until') could be embedded in yet another control loop (not shown here) and use a search strategy similar to the inner level, thus leading to a multi-level search algorithm [14].

4 A Genetic Algorithm for the JSP

Genetic algorithms (GAs) are general search strategies and optimization methods motivated by the theory of evolution; they date back to the early work of Holland [19] and Rechenberg [30], see also [16]. Simply speaking, a GA aims at producing near-optimal solutions by letting a set (population) of random solutions (individuals) undergo a sequence of unary and binary transformations governed by a selection scheme biased towards high-quality solutions. The solutions manipulated by a GA are usually represented as binary strings, e.g. a binary number or a vector of such numbers. The transformations are applied to the population by three simple operators: selection, mutation, and crossover. The effect of the operators is that implicitely good properties are identified and combined into new individuals of a new population which hopefully has the property that the best solution and the average value of the solutions are better than in previous populations. This

Figure 2: *A genetic algorithm*

```
t := 0;            {t is the generation counter}
initialize P(t);   {P(t) is the population in generation t}
evaluate P(t);
while the stopping criteria are not satisfied do
begin
   t := t + 1;
   select P(t) from P(t − 1);
   recombine P(t);
   evaluate P(t);
end;
```

process is repeated until some stopping criteria are met. Figure 2 shows the outline of a GA.

In the selection step, a copy of an old individual is produced with a probability proportional to its fitness value, i.e. better strings probably get more copies. Selection can be realized in a number of ways; one could adopt the scenario of Goldberg [16] or use deterministic ranking. Besides, it matters whether the newly recombined offspring compete with the parent solutions or simply replace them. Recombination consists of crossover and mutation. In order to apply the crossover operator, the population is randomly partitioned into pairs. Next, for each pair, crossover is applied with a certain probability by randomly choosing a position in the string and exchanging the tails (defined as the substring starting at the chosen position) of the two strings. The mutation operator which makes random changes to single elements of a string only plays a secondary role; its main purpose is to maintain diversity in the population.

Compared to standard heuristics, for instance for the traveling salesman problem "genetic algorithms are not well suited for fine-tuning structures which are very close to optimal solutions" [18]. However, it is often easy to extend a GA to incorportate (local search) improvement operators in the evaluation step. The resulting algorithm has been called genetic local search heuristic; in case of the traveling salesman problem we refer to the papers of Ulder et al. [33] and Kolen and Pesch [21].

In order to apply a GA to an optimization problem, solutions must be encoded in a format that can be manipulated by the GA. The traditional GA based on a binary string representation of a solution is often unsuitable for combinatorial optimization problems because it is very difficult to represent a solution in such a way that the substrings have a meaningful interpretation. Choosing a more natural representation, however, involves more intricate recombination operators to ensure that the offspring is feasible; for an example see the crossover operators developed for the JSP by Aarts et al. [1] and Nakano and Yamada [28].

The underlying idea of the GA described in this paper is to use the genetic information to guide a heuristic which finds a starting solution for the JSP. The GA thus serves as a meta-heuristic which produces a sequence of decision rules that direct another algorithm. The output of this algorithm can then be improved by a local search procedure, and the improved solution is finally inserted into the GA population again. Using the strings of a GA to guide a scheduling heuristic has first been suggested by Davis [7]. Applications of a GA to the JSP have been described in [28, 9, 34].

Before we take a closer look at the GA itself, let us briefly introduce the algorithm of Giffler and Thompson [11], which can be considered as a common basis of all priority rule based heuristics for the JSP. The procedure can generate all active and hence also all optimal schedules. The algorithm, which is outlined in figure 3, iteratively assigns operations from the set Q of unscheduled operations to machines. In the description in figure 3, r_i and c_i denote the earliest possible start and completion time, respectively, of operation i.

In the first step of each iteration, the machine on which the next operation has to be scheduled

Figure 3: *The algorithm of Giffler and Thompson*

$Q := \{1, \ldots, N\}$;
repeat
among all unscheduled operations in Q let j^* be the one with smallest completion time, i.e. $c_{j^*} = \min \{j \in Q\}$; let m^* denote the machine j^* has to be processed on;
randomly choose an operation i from the conflict set $C = \{j \in Q | j$ has to be processed on machine m^* and $r_j < c_{j^*}\}$;
$Q := Q - \{i\}$; modify r_j and c_j for all operations $j \in Q$;
until $Q = \emptyset$;

is chosen in such a way, that only active schedules will be generated (see [11]). In the second step, conflicts, i.e. operations competing for the same machine, are resolved randomly. In priority rule based heuristics, an operation from the conflict set C is selected according to a priority rule rather than randomly, for instance "choose the operation with the smallest processing time".

Using the Giffler/Thompson algorithm within the framework of a GA is straightforward. The random choice of an operation from the conflict set can be replaced with a choice based on a decision rule, where either the rule itself or the information used within the rule is supplied by a GA. For example, as described in [9], a GA can manipulate strings of priority rules that are then evaluated by using them in the iterations of the Giffler/Thompson algorithm.

Here, we let the GA manipulate the information to be used in a decision rule. An individual member of the population corresponds to a job shop schedule; it is a string of n entries, where n is the number of operations in the problem instance. An entry i represents the starting time t_i of operation i in the schedule. Because the vector of starting times can easily be stored in the traditional form of a binary string, the standard crossover and mutation operators can be applied. A newly recombined string is evaluated by using it as input for guiding the Giffler/Thompson algorithm. Instead of randomly picking an operation from the conflict set C, the choice is based on the string information that is used in the following 'earliest starting time' rule:

Choose the operation i^* in the conflict set C for which $t_{i^*} = \min \{t_i | i \in C\}$.

Yamada and Nakano [34] have independently described a crossover operator that is based on the same genetic string representation. During crossover, the schedules of the individuals to be crossed are used to guide the Giffler/Thompson algorithm; the random choice of an operation from the conflict set is replaced by the following decision sequence:

1. Apply mutation with a small probability by randomly choosing an operation from the conflict set.
2. If there was no mutation then randomly (with equal probability) select one of the two parents to be crossed and choose the operation i^* in the conflict set C for which $t_{i^*} = \min \{t_i | i \in C\}$, where t_i denotes the starting time of operation i in the selected parent's schedule.

5 Computational Results

The GA with local variable depth search has been implemented in C and tested on a Sun SPARCstation IPX. We have used Grefenstette's general purpose genetic search system GENESIS [17] for the GA part of our algorithm. Limited initial tests have been performed using the three famous

problem instances introduced by Fisher and Thompson in 1963 [10] which have since then served as a test for almost every algorithm for the JSP. We have tested both the standard crossover where the resulting individual string serves for guiding the Giffler/Thompson algorithm and the Giffler/Thompson crossover of Yamada and Nakano; both versions of the algorithm have given similar results. The algorithm has been run five times on each problem instance, and all instances have been solved to optimality within a GPU time of ten minutes for a single run. While the 6×6 problem and the 5×20 problem are relatively easy, it is quite remarkable that the algorithm has always solved the notoriously difficult 10×10 instance.

In our tests, we have used the following GA parameters: a crossover rate of 0.8, no mutation, a generation gap of 1 and a window size of 5 (see [17]), an elitist strategy, where the best individual of a generation always survives reproduction, and an improvement probability of 0.2, meaning that on average 20% of the newly recombined individuals are improved by the local search procedure; this parameter has been selected after a few experiments and it seems possible that it can be improved. It is likely that the look-ahead depth $|E|$, the cardinality of the edge set, used in the variable depth search could be optimized; in our experiments, the optimal depth d^* has usually been reached after reversing only a small fraction of the total number of disjuncive edges. Since the control parameters have not been fine-tuned, we suspect that the efficiency of our algorithm could still be increased by the 'tender loving care factor'.

Because our intial tests have been limited to a small number of experiments with only three problem instances, the results are not yet very conclusive, so great care needs to be taken when comparing them to the results obtained by applying other generally applicable approximation techniques to the JSP as described in [25, 32, 22, 1, 8, 28, 34]. We would just like to remark that the results and running times indicate that our algorithm is at least competitive. When compared to modern exact methods and tailored approximation methods for the JSP [2, 6, 3, 5], the running times of the algorithm seem relatively high. However, these methods are substantially more involved than the algorithm described here, and extending them to modified versions of the problem is not easy.

6 Conclusions

We have presented a genetic algorithm that guides the Giffler/Thompson heuristic for building active schedules which are then improved by a variable depth search procedure. The algorithm which is comprised of quite simple building blocks has solved the notoriously difficult 10×10 problem instance of Fisher and Thompson to optimality.

The work described in this paper will be extended in several directions. Firstly, more conclusive computational results will be produced by applying the algorithm to a larger suite of standard test problems and by comparing its results to those obtained by applying the individual components separately. Secondly, more sophisticated search neighbourhood structures as described in [8] might be used, and thirdly, the variable depth search technique could be replaced with tabu search as described in [12, 13, 8] for comparing the two methods.

References

[1] E.H.L. Aarts, P.J.M. van Laarhoven, and N.J.L. Ulder (1991): Local-search-based algorithms for job shop scheduling. Working paper, University of Eindhoven.

[2] J. Adams, E. Balas, and D. Zawack (1983): The shifting bottleneck procedure for job shop scheduling. Management Science 34, 391–401.

[3] D. Applegate and W. Cook (1991): A computational study of the job-shop scheduling problem. ORSA Journal on Computing 3, 149–156.

[4] E. Balas (1969): Machine sequencing via disjunctive graphs: an implicit enumeration algorithm. Operations Research 17, 941–957.

[5] P. Brucker, B. Jurisch, and B. Sievers (1991): A fast branch & bound algorithm for the job-shop scheduling problem. Discrete Mathematics (to appear).

[6] J. Carlier and E. Pinson (1989): An algorithm for solving the job-shop problem. Management Science 35, 164–176.

[7] L. Davis (1985): Job shop scheduling with genetic algorithms. Proc. an Int. Conference on Genetic Algorithms and their Applications (J.J. Grefenstette, ed.), Lawrence Erlbaum Ass., 136–140.

[8] M. Dell'Amico and M. Trubian (1992): Applying tabu-search to the job-shop scheduling problem. Annals of Operations Research (to appear).

[9] U. Dorndorf and E. Pesch (1992): Evolution based learning in a job shop scheduling environment. Research Memorandum 92-019, University of Limburg.

[10] H. Fisher and G.L. Thompson (1963): Probabilistic learning combinations of local job-shop scheduling rules. Industrial Scheduling (J.F. Muth and G.L. Thompson, eds.), Englewood Cliffs, Prentice Hall.

[11] B. Giffler and G.L. Thompson (1960): Algorithms for solving production scheduling problems. Operations Research 8, 487–503.

[12] F. Glover (1989): Tabu Search — Part I. ORSA Journal on Computing 1, 190–206.

[13] F. Glover (1990): Tabu Search — Part II. ORSA Journal on Computing 2, 4–32.

[14] F. Glover (1991): Multilevel tabu search and embedded search neighborhoods for the traveling salesman problem. Working paper, University of Colorado, Boulder.

[15] F. Glover and H.J. Greenberg (1989): New approaches for heuristic search: A bilateral linkage with artificial intelligence. European J. Operational Research 13, 563–573.

[16] D.E. Goldberg (1989): Genetic Algorithms in Search, Optimization and Machine Learning. Reading, Addison Wesley.

[17] J.J. Grefenstette (1987): A user's guide to GENESIS. Navy Center for Applied Research in Artificial Intelligence, Naval Research Laboratory, Washington, D.C.

[18] J.J. Grefenstette (1987): Incorporating problem specific knowledge into genetic algorithms. Genetic Algorithms and Simulated Annealing (L. Davis, ed.), Pitman, 42–60.

[19] J.H. Holland (1975): Adaptation in Natural and Artificial Systems. Ann Arbor, The University of Michigan Press.

[20] B.W. Kernighan and S. Lin (1970): An efficient heuristic procedure for partitioning graphs. The Bell System Technical Journal 49, 291-307.

[21] A. Kolen and E. Pesch (1991): Genetic local search in combinatorial optimization. Discrete Applied Mathematics (to appear).

[22] P.J.M. van Laarhoven, E.H.L. Aarts, and J.K. Lenstra (1992): Job shop scheduling by simulated annealing. Operations Research 40, 113–125.

[23] J.K. Lenstra and A.H.G. Rinnoy Kan (1979): Computational Complexity of discrete optimization problems. Annals of Discrete Mathematics 4, 121–140.

[24] S. Lin and B.W. Kernighan (1973): An effective heuristic algorithm for the traveling salesman problem. Operations Research 21, 498–516.

[25] H. Matsuo, C.J. Suh, and R.S. Sullivan (1988): A controlled search simulated annealing method for the general jobshop scheduling problem. Working paper 03-44-88, Graduate School of Business, University of Texas, Austin.

[26] H. Mühlenbein, M. Gorges-Schleuter, and O. Krämer (1988): Evolution algorithms in combinatorial optimization. Parallel Computing 7, 65–85.

[27] H. Mühlenbein (1991): Evolution in time and space – the parallel genetic algorithm. Foundations of Genetic Algorithms (G. Rawlins, ed.), Morgan Kaufmann, 316–337.

[28] R. Nakano and T. Yamada (1991): Conventional genetic algorithm for job shop scheduling problems. Proc. 4th Int. Conference on Genetic Algorithms and Their Applications. San Diego, CA, 474–479.

[29] C.H. Papadimitriou and K. Steiglitz (1982): Combinatorial Optimization. Algorithms and Complexity. Englewood Cliffs, Prentice-Hall.

[30] I. Rechenberg (1973): Optimierung technischer Systeme nach Prinzipien der biologischen Evolution. Problemata, Frommann-Holzboog.

[31] B. Roy and B. Sussman (1964): Les problèmes d'ordonnancement avec contraintes disjonctives. Paris, SEMA, Note D.S. No. 9.

[32] E. Taillard (1989): Parallel taboo search technique for the jobshop scheduling problem. Internal Report ORWP 89/11, Département de Mathématiques, Ecole Polytechnique Fédérale de Lausanne.

[33] N.L.J. Ulder, E.H.L. Aarts, H.-J. Bandelt, P.J.M. van Laarhoven, and E. Pesch: Genetic local search algorithms for the traveling salesman problem. Proc. 1st. Int. Workshop on Parallel Problem Solving from Nature (H.-P. Schwefel and R. Männer, eds.), Lecture Notes in Computer Science 496, 109–116.

[34] T. Yamada and R. Nakano (1992): A genetic algorithm applicable to large-scale job-shop problems. Proc. 2nd. Int. Workshop on Parallel Problem Solving from Nature (to appear).

Finding an Initial Feasible Point in the Large Scale GRG Code CONOPT.

by

Arne Stolbjerg Drud
ARKI Consulting and Development A/S
Bagsvaerdvej 246 A
DK-2880 Bagsvaerd, Denmark
Phone: (+45) 44 49 03 23
Fax: (+45) 44 49 03 33
E-mail: ad@vip.imsor.dth.dk

1. Introduction.

Nonlinear Programming (NLP) algorithms can be classified into algorithms that generate a sequence of feasible points and algorithms where the intermediate points in general are infeasible. The first class, called feasible path methods, can often be made very reliable because they work with feasible points. However, they require a method for generating an initial feasible point.

This paper describes a new algorithm for finding an initial feasible point in connection with the Generalized Reduced Gradient (GRG) algorithm (Abadie and Carpentier, 1969), and in particular in the large sparse GRG algorithm CONOPT, (Drud, 1985 and 1992). The algorithm is based on ideas from Crash procedures in Linear Programming (LP) with adjustments that take into account the special features of nonlinear models.

The paper is organized as follows: Section 2 defines our problem and assumptions. Section 3 summarizes traditional methods used for finding an initial feasible solution in GRG algorithms. Section 4 describes the proposed crash procedure. Section 5 contains some limited computational experience, and section 6 concludes the paper.

2. Assumptions.

We consider nonlinear programs of the following general form:

$$\min f(x) \tag{1}$$

subject to

$$g(x) = b \tag{2}$$

and

$$l \le x \le u \tag{3}$$

where x is the n-vector of decision variables, g is the m-vector of constraint functions, f is the objective function, b is the m-vector of right hand sides, and l and u are n-vectors of lower and upper bounds. Some of the bound values may be minus or plus

infinity. We assume that f and g are defined and have continuous derivatives for all values of x satisfying the bounds (3).

We assume that there are hundreds or thousands of equations and variables, and that the Jacobian is sparse. In addition, we assume that the individual nonlinear functions and their derivatives can be computed independently. This is a reasonable assumption when the model is communicated to the solution algorithm by a modeling system such as the General Algebraic Modeling System, GAMS (Brooke et.al., 1988), currently the most widely used input generator for CONOPT.

3. The GRG Algorithm and its Phase-1 Procedure.

When the GRG algorithm is described the problem of finding an initial feasible solution is usually ignored.

Traditionally, the problem of finding an initial feasible solution has been attacked similar to the way it is done in Phase 1 in LP: Artificial variables with suitable bounds are added to the infeasible equations to yield a relaxed but feasible model. The sum of the artificials is then minimized, subject the equations of the relaxed model. The solution to this phase-1 model is either a point in which all artificial variables are zero, i.e. a feasible solution to the original problem, or a strictly positive local minimum of the sum of infeasibilities in which case the model is considered (locally) infeasible.

The computational cost of this phase-1 procedure will depend on the number of artificial variables in the initial point, i.e. on the number of infeasible equations, and on the size of the infeasibilities. The procedure may be relatively slow on models with many small infeasibilities because the removal of each artificial, independent on its initial size, requires at least one iteration. We have therefore implemented an initial Phase-1 heuristic in CONOPT to get around this problem. The heuristic is summarized in Fig. 1.

1. Select a set of basic variables favoring variables away from bounds.
2. Perform a Newton step using these basic variables
   a Use steplength < 1 if a basic variable otherwise would exceed a bound
   b Change the basis of the critical basic variable is at bound
   c If the iterations do not converge due to non-linearities then
      - Change the basis or
      - Remove "bad" equations from the Newton process
3. When the "good" equations are feasible, add artificials to the "bad" equations and minimize the sum of artificials using the standard GRG procedure.

Figure 1: The Phase-1 heuristic in CONOPT.

When many variables have good initial values the heuristic will be able to select a good basis with many variables away from their bounds. And a feasible solution can be found very quickly, essentially with the speed of a Newton procedure.

However, there are several disadvantages. If there are few variables away from bounds in the initial point then it is difficult to find a good basis and the procedure may use many degenerate iterations before it gives up and switches to the standard minimization of artificials. Even worse, the procedure does not work well with the way many model builders define initial values: Important or critical variables are assigned reasonable values while unimportant variables are left uninitialized. Variables with "good" values, i.e. those away from bounds, are likely to be selected as basic variables and changed during the initial iterations, while variables with "bad" values, usually at their bounds, are kept unchanged.

4. The crash procedure.

Most commercial LP systems have a so-called crash procedure that can be applied to find an initial basis. The purpose is, with few resources, to find a point with few artificial variables and use it as an initial point for the Phase-1 LP.

A good description of crash procedures can be found in (Gould and Reid, 1989). One of the procedures advocated in this paper is based on very simple principles:

1. Order the equations and variables into almost triangular form.
2. Solve the equations one by one in this order, keeping the variables from previous equations fixed.
3. If an equation is infeasible, solve a larger subproblem involving some of the previous equations. If still infeasible, introduce an artificial variable.

The procedure can easily be generalized to nonlinear equations. The ordering into almost triangular form is independent of linearity. Step 2 involves the solution of one equation at a time, and it can easily be generalized to nonlinear equations by substituting an explicit solution with an iterative procedure based on Newton's method. Step 3 involves the solution of sets of several equations. We have not yet implemented this step, but a method based on Newton's method can also be used here.

The ordering in Gould and Reid's paper is based on the P3 procedure in (Hellerman and Rarick, 1971). It can also be implemented in a slightly different way as follows:

1. Compute row counts as the number of nonzero Jacobian elements in each row. Columns with fixed variables are excluded.
2. If there are no rows left, Stop. Otherwise, find RCmin, the minimal row count.
3. RCmin = 0: Select the row(s) with row count 0, remove the row(s) from further consideration, and go to 2.

4. RCmin = 1: Select a row with row count 1 and its corresponding column, remove the row and column from the Jacobian, update row counts, and go to 2.
5. RCmin > 1: Select a row with row count RCmin and its corresponding columns, remove the row and columns from the Jacobian, update row counts, and go to 2.

The order in which the rows and columns are selected defines their order in the almost triangular form.

The ordering procedure is not well defined in the selection in step 4 and 5. Builders of LP models will usually not provide initial values and the selection is therefore only based on the sparsety structure of the LP matrix.

The situation is quite different for nonlinear models. As a result of the sequential solution procedure some variables will be kept fixed, while the remaining variables will be computed as functions of these fixed variables and of variables computed earlier in the sequential process. We should therefore try to order the variables and equations such that fixed variables have "good" initial values, while the equations preferably are solved with respect to variables without a "good" initial value.

The freedom in selecting variables to keep fixed appear only when RCmin > 1 in step 4 above. In the following we will discuss how we determine whether an initial value is "good", and how this influences the ordering. We will start with a small example.

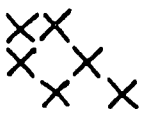


Figure 2: The Jacobian Structure of a Small Model.

Fig. 2 shows the structure of the Jacobian of a model with four variables, three equations, and RCmin = 2. If any one of the four variables is fixed, the three equations can be solved recursively for the remaining three variables.

Let $\underline{x}(x_i)$ denote the solution for x_i fixed and let $\underline{X}$ be a vector of initial values. Depending on which of the four variables we fix we can compute four initial points: $\underline{x}(X_1)$, $\underline{x}(X_2)$, $\underline{x}(X_3)$, and $\underline{x}(X_4)$. Note that some of these points may be infeasible because of bounds on the variables. Also note that if x_i is fixed then the initial values of the other variables are ignored, except as initial points in the solution process.

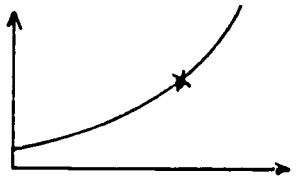
Our problem is to select one of these four points without actually evaluating them all, i.e. from $\underline{X}$ only. A number of characteristics of the different points may help us in the selection:

A If X_i has a "good" initial value then $\underline{x}(X_i)$ will be a "good" solution.
B User supplied initial values are likely to be "better" than default initial values.
C If an equation is feasible, the variables appearing in it are more likely to have "good" values.
D If x_i and x_j are the only variables in an equation and this equation is feasible then $\underline{x}(X_i) = \underline{x}(X_j)$.

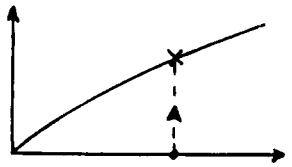
We will in the following separate the variables into two groups: variables with default initial values and variables with defined initial values. The definition will depend on how the model is communicated to CONOPT. When the model comes from GAMS zero projected on the bounds is considered a default initial value and other values are defined. Variables with defined initial values will in general be considered "better" that variables with default initial values.

We will also separate the equations into feasible and infeasible constraints. An initial value that appears in a feasible constraint will in general be considered "better" than an initial value that appears in an infeasible constraint. If an infeasibility can be repaired by adjusting a variable with default initial value the default variable could be an uninitialized intermediate variable, and the infeasibility is not considered to be bad.

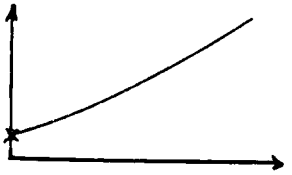
Based on these considerations we select a "best" row with row count = RCmin whenever RCmin > 1. The selection is done by giving priorities from 1 to 6 to the candidate rows and selecting a row with the smallest priority. The priorities are defined as follows:



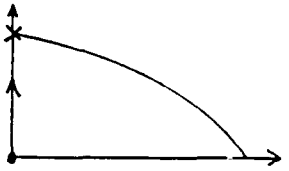
1. Feasible with at least 2 defined variables. Since the equation is feasible the defined values seem to be "good". Select the variable with the largest Jacobian as basic.



2. Infeasible with 1 default variables that can be changed to satisfy the equation. The defined values seem to be reasonable. Select the default variable as basic and solve the equation w.r.t this variable.



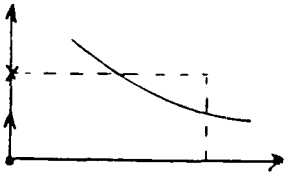
3. Feasible with 1 defined variable. Since the equation is feasible the defined value seem to be "good". Select the defined variable as basic.



4. Infeasible with at least 2 default variables that both can be changed to satisfy the equation. The equation can be made feasible, but the solution depends on which variable is changed. We select the variable with the largest pivot to be changed to minimize the absolute change.



5. Feasible with only default values. Since all values are default the feasibility seems to be accidental. Select largest pivot as basic.



6. Infeasible. No single variable can be changed to make the equation feasible. Select the variable that will reduce infeasibility the most.

Whenever a row is selected we try to make it feasible immediately. The updated values of the variables are then used to evaluate feasibility during the selection of the next row. This is in contrast to the LP environment where the logical ordering is done before the solution process is started.

When RCmin = 1 we must select a particular row with row count one. If each row has its own column then the solution is independent of the order in which the rows are selected. However, if a potential pivot column intersects more than one candidate row the solution will depend on which row is selected. In this case we try to minimize the sum of infeasibilities in the remaining rows, and the row selection is similar to the CHUZR procedure in Rarick's Phase-1 procedure for LP.

Many equations will be feasible and we will have a basic variable in most equations after the procedure outlined above has been used. However, there may still be some equations without a basic variable: Equations selected when RCmin = 0, and equations that cannot be made feasible.

The basis can be completed with artificial variables and the traditional phase-1 procedure can be applied to minimize the sum of the artificials. Alternatively, we may select the missing basic variables from the variables away from bounds and use the heuristic in Fig. 1.

5. Computational Experience.

A set of tables with computational comparisons is available from the author. We will here summarize some numbers from a medium sized 8-period refinary model. The model has 1793 variables of which 1464 have defined initial values. The models 1593 equations are divided onto 95 pre-triangular equations that can be solved recursively before the optimization and 2 post-triangular equations that can be collapsed into the objective function. Of the remaining 1496 equations 1199 (80%) are made feasible with a basic variable by the crash procedure while 297 equations are not assigned a basic variable initially.

The sum of infeasibilities is initially 4844.6. After solving the 95 pre-triangular equations and removing the 2 post-triangular equations there are 165 infeasible equations and the sum of infeasibilities is 4772.6. The crash procedure produces a point with only 30 infeasible equations (a reduction of 82%) and a sum of infeasibilities of 33.14 (a reduction of 99.3%).

The original feasibility heuristic mentioned in Fig. 1 including the following ordinary phase-1 procedure required 1165 iterations and 582 seconds to find a feasible solution and the overall optimization required 2596 iterations and 1698 seconds. The crash procedure followed by the heuristic required 246 iterations and 125 seconds to find a feasible solution (78% saving) and 1669 iterations and 1139 seconds to find the optimal solution (32% saving). The crash procedure followed by a straight minimization of artificials required 613 iterations and 219 seconds to find a feasible solution (62% saving) and 2269 iterations and 1591 seconds to find the optimal solution (6% saving).

The saving on other models vary considerable, but is is positive on almost all models. There is also considerable variation between the options for finishing the basis -- the old feasibility heuristic or minimization of artificials.

One interesting result is that some difficult models that CONOPT declared infeasible before now prove to be feasible. The reason seems to be that the crash procedure moves many variables with default initial values away from their bounds or from zero, resulting in a better behaved point that is further away from any singularities.

6. Conclusions.

Although the computational testing is still ongoing we can already conclude that the new crash procedure is very promising. Given a few good initial values we will on most models be able to reduce the time to find an initial feasible solution. The initial feasible solution will often be better which reduces the following optimization time. And we seem to be able to solve more difficult models that could not be solved before.

The more intelligent use of initial values may mean that it is no longer necessary to supply initial values for many intermediate variables. This simplifies model construction and may

incourage model builders to concentrate on essential initial values. User specifications of the quality of initial values could enhance the procedure.

More work is still needed, in particular on:
- improving the selection of variables to fix, e.g. based on information from several equations and on the influence on the objective,
- completing the basis after the crash procedure, and
- limited backtracking when equations are infeasible.

7. References.

J. Abadie and J. Carpentier, 1969. Generalization of the Wolfe Reduced Gradient Method to the case of Nonlinear Constraints, in *Optimization*, R. Fletcher (ed.), Academic Press, New York, pp 37-47.

A. Brooke, D. Kendrick, and A. Meeraus, 1988. *GAMS - A User's Guide*, The Scientific Press, Redwood City.

A. Drud, 1985. A GRG Code for Large Sparse Dynamic Nonlinear Optimization Problems, *Mathematical Programming 31*, 153-191.

A. Drud, 1992. CONOPT - A Large Scale GRG Code, *ORSA Journal on Computing*, (forthcoming).

N.I.M. Gould and J.K. Reid, 1989. New Crash Procedures for Large Systems of Linear Constraints, *Mathematical Programming 45*, 475-502.

E. Hellerman, and D. Rarick, 1971. Reinversion with the Preassigned Pivot Procedure, *Mathematical Programming 1*, 195-216.

Shape Optimization of Complex Shell Structures in a Parallel Computing Environment

Eschenauer, H.A. *Weinert,M.*

Research Center for Multidisciplinary Analysis
and Applied Structural Optimization
Institute for Mechanics and Control Engineering
University of Siegen
D-W 5900 Siegen, Germany
E-mail:weinert@fb5.uni-siegen.de

1 Introduction

Structural optimization has become an increasingly important tool in the design process, due to the continuously increasing demands on technical systems and their components. Because of an extended application of structural optimization techniques to real, industrial problems, the portion of so-called large scale problems increases accordingly. The latter problems are characterized by a high demand on computer resources (storage capacity, calculation time) within the solution process. Various decomposition techniques have been developed in order to efficiently solve such large scale problems [Wis71, Him73]. Parallel processing means a computational decomposition of a task onto different processors or computer nodes, and therefore it is also a very general decomposition approach. Here, the solution of shape optimization problems of complex shell structures on a parallel computer system will be presented. As an application we have chosen the shape optimization of an automotive wheel with respect to several load cases.

2 Treatment of shape optimization problems

The mathematical formulation of shape optimization problems can be written as follows:

$$F^*[R^k(\xi^\alpha)] = \min_{R^k}\left\{F[R^k(\xi^\alpha)] \,\middle|\, R^k(\xi^\alpha) \in G\right\} \tag{1}$$

$$G = \left\{R^k(\xi^\alpha) \in R^3 \,\middle|\, \begin{array}{l} H[R^k(\xi^\alpha)] = 0, \\ G[R^k(\xi^\alpha)] \geq 0 \\ R^{kl} \leq R^k \leq R^{ku} \end{array}\right\}.$$

with

F	:	vector of objective functionals,
H, G	:	equality and inequality constraint functionals,
R^k	:	shape functions,
ξ^α	:	GAUSSian parameters,
R^{kl}, R^{ku}	:	lower and upper bounds for the shape functions,
G	:	set of feasible shape functions.

The shape optimization problems formulated in (1) can be solved by means of indirect and direct methods. Indirect procedures derive necessary conditions for the optimal shape using variational principles and, subsequently, solve the resulting differential equations - generally by approximation methods. When a direct solution method is applied, the shape optimization problem (1) is transformed into a *multicriteria parameter optimization problem* using approach functions with free parameters $\hat{R}(\xi^a, x^k)$. Especially parametrical spline functions known from the field of CAD [Boe84] proved to be highly efficient for various applications [Bra84]. The obtained multicriteria formulation is subsequently transformed by means of preference strategies into a *scalar parameter optimization problem* which can finally be solved by any Mathematical Programming algorithm (MP-algorithm).

Structural optimization problems can be solved by an optimization procedure following the Three-Columns-Concept [Esc91]. The three columns are the optimization algorithms, the structural analysis modules and the optimization model. All moduls can be chosen according to the problem formulation. The direct optimization strategy is realized in the design model (Fig. 1). The approach functions are chosen problem-dependent from an extensive library.

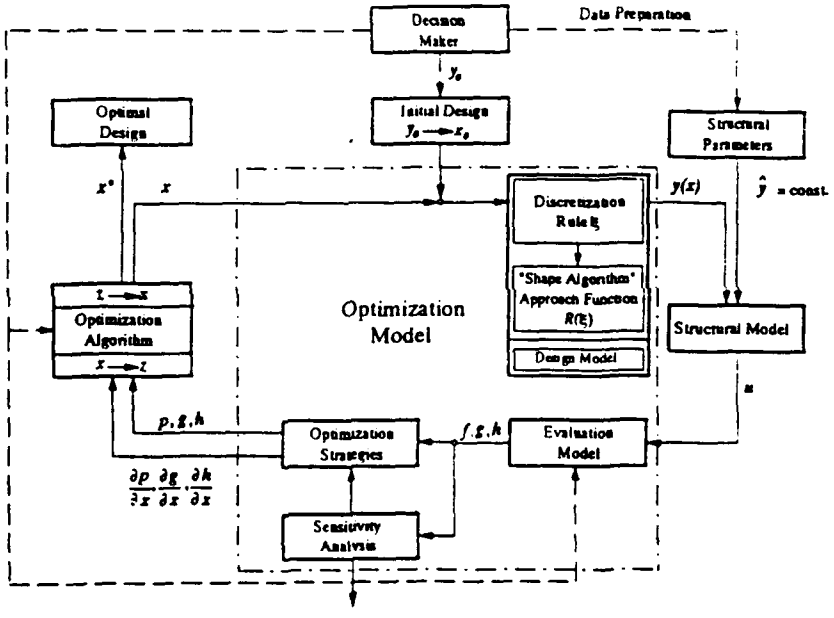


Figure 1: Optimization loop for shape optimization problems

3 Decomposition

The large-scale problems of type (1) occuring with the optimization of real-life technical systems require a high storage capacity and extensive calculation times. The efficient solution of such problems by means of the available resources (storage capacity/computation time) calls for the application of decomposition techniques.

Fig. 2 shows the potential decomposition *approaches* for problems in structural optimization. The decomposition *methods* usually employ several of the depicted decomposition approaches. While model decomposition mainly aims at reducing the required computer storage, the computational decomposition intends to reduce the computation time required for the solution.

The term *structural decomposition* means partitioning the state vector u and the corresponding

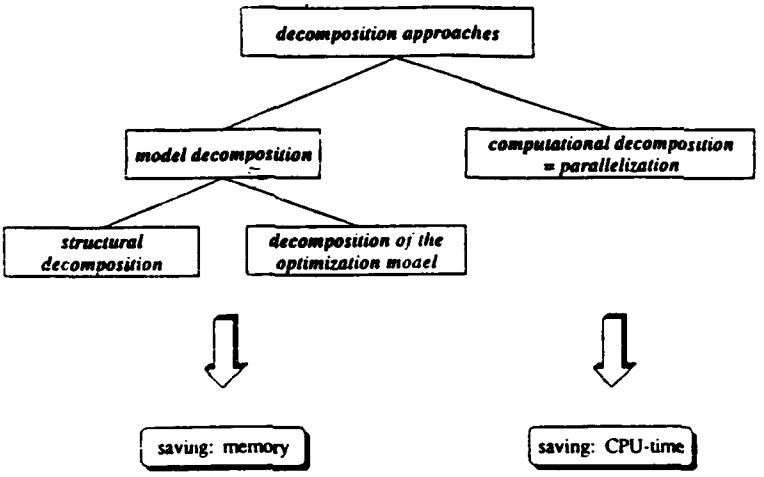


Figure 2: Classification of decomposition approaches in the field of structural optimization

state equations Φ into n_S subvectors,

$$u,\ \Phi \Longrightarrow [u_1, u_2, \ldots, u_{n_S}],\ [\Phi_1, \Phi_2, \ldots, \Phi_{n_S}].$$

This partitioning can be done according to several substructures [Bat76, Guy65] or according to different disciplines. If the structural subproblems are coupled with each other - this is in most cases true - a coordination procedure has to be used.

Decomposition of the optimization model requires the partitioning of the design variables x and/or the constraints h, g into subvectors,

$$x,\ h,\ g \Longrightarrow [x_1, x_2, \ldots, x_{n_S}],\ [h_1, h_2, \ldots, h_{n_S}],\ \left[g_1, g_2, \ldots, g_{n_S}\right].$$

After solving the subproblems independently, the solutions of the subsystems must be coordinated. The coordination and the concurrent subsystem optimizations form a iterative process. Methods using this approach are, for example, the DANTZIG-WOLFE-decomposition [Dan60] for large linear optimization problems, Multilevel-Methods for hierarchically structured systems (e.g. [Wis71, Las70] or for non-hierarchical systems (e.g. [Blo90, Wu92]), and methods based upon a substructuring (e.g. [Kir72, Bre89].

If a suitable parallel computer equipment is available, the *computational decomposition*, which means nothing else than a parallel or distributed execution of independent calculation tasks can be applied. Within the process of optimization, the sensitivity analysis is a subtask suitable for parallelization since the computation of the partial derivatives with regard to the design variables are uncoupled processes and can therefore be parallelized. In many cases, the parallelization is carried out in combination with a model decomposition (e.g. in combination with substructuring [Top91, ElS91]). The computations in the subsystem level are then independent of each other and can be parallelized. In order to quantitatively evaluate the gain achieved by parallelization, the so-called speed-up S is introduced:

$$S = \frac{T_{seq}}{T_n}, \tag{2}$$

where T_{seq} denotes the required calculation time on one processor and T_n the calculation time on n processors.

For the parallelization a transputer system consisting of 20 transputers (T800) is available (Fig. 3). The transputers are arranged in an array and their local storage capacities ranges between 1MB and 8MB. The definition of the processes, the communication between them, and the allocation of the processors is carried out by means of the declaration language CDL (Component Declaration Language) under the operating system HELIOS [NN90]. The topology of the processes can be

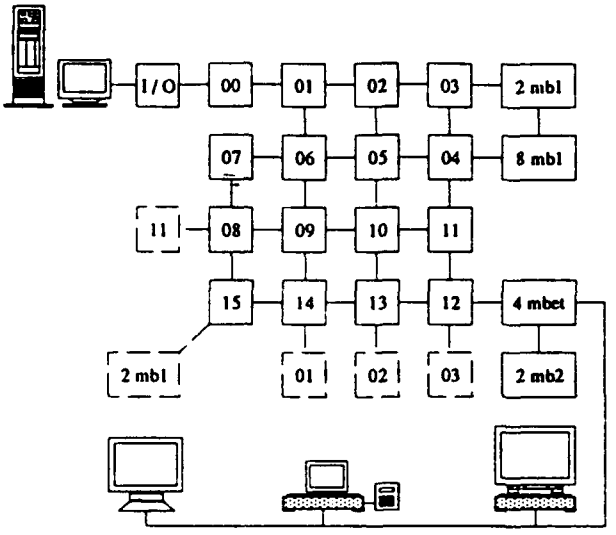


Figure 3: Parallel computing environment

defined independently from the hardware topology.

In the present work a *parallel substructure technique* and the *parallel sensitivity analysis* is employed for the solution of the shape optimization problems. A decomposition of the optimization model is not carried out.

a) Optimization using parallel substructuring:

Substructuring is commonly seen as the classical (static) decomposition technique for the structural analysis of complex components. Reduced subsystem matrices are calculated separately for each substructure, and these matrices are coupled at the main system level. After determining the state variables at the boundaries (coupling nodes) of the substructures, the local (internal) ones can be computed - again separately for each subsystem. Fig. 4a shows the flowchart of the parallel structural analysis realized here, based upon the substructuring, where the computations of the main and of the subsystems are carried out on an own processor each. Since the main system processor is not employed during the subsystem computations, one subsystem is treated on the main processor. The described procedure is not limited to the application of a special analysis program at the subsystem level, but this analysis program must be able to *create reduced stiffness matrices* and *consider prescribed displacements.*

The implementation of the above procedure on the transputer system follows the *master-slave-concept.* In this concept a process called *master* controls and coordinates a set of subsequent *slave* processes. The program representing the slave contains all modules required for the various calculation tasks at the subsystem level (Fig. 4a). Additionally, it possesses a local database which stores - even for several structural models - the necessary control data. This guarantees a minimal data transfer during optimization because the updated geometry and the resulting state variables have to be transfered only. The master process contains the complete optimization loop including the routines for subsystem calculations, because one subsystem is also analyzed by the master process. The necessary system-calls for the purpose of communication are carried out by a small set of hardware-independent modules only, which reduces the effort when this concept is implemented on another computer system.

b) Parallel sensitivity analysis:

The sensitivity analysis is a very time-consuming subtask within the optimization process. It requires the calculation of partial derivatives of the objectives and constraints with respect to the design variables ($\partial f/\partial x_i, \partial g/\partial x_i, \partial h/\partial x_i$). Here, we approximate them by finite differences (first order differences). The approximation of the first derivative of an arbitrary function $F(x)$ with

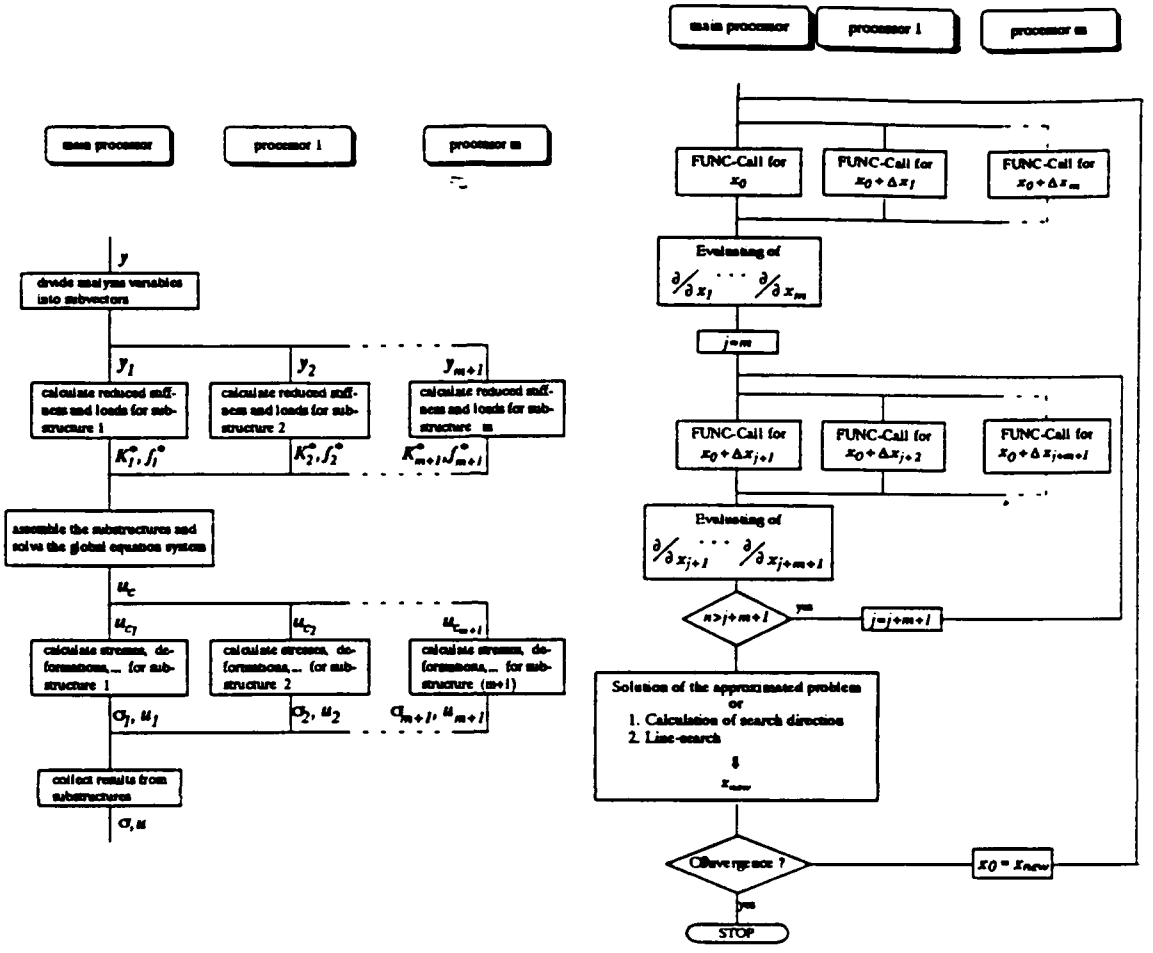


Figure 4: Parallel substructuring (a) and parallel sensitivity analysis (b)

respect to x_i using the forward difference is:

$$\left.\frac{\partial F(x)}{\partial x_i}\right|_{x=x_0} \approx \frac{F(x_0 + \Delta_i) - F(x_0)}{|\Delta_i|}, \qquad \Delta_i = (x_{0_1}, \ldots, x_{0_i} + \Delta x_i, \ldots, x_{0_n}). \tag{3}$$

As shown in (3), one needs as many functional evaluations as defined design variables n. One functional evaluation means one pass of the optimization loop (Fig. 1). Since the functional evaluations are independent from each other, the sensitivity analysis can be parallelized. In contrast to the decomposition method described in a), this method only reduces the required computer time but not the required computer memory. Fig. 4b depicts one iteration with parallel sensitivity analysis. The optimal load balance of the processors can be obtained, if the condition

$$(n+1)/(m+1) \in Z \tag{4}$$

is fulfilled.

The implementation on the transputer system is also done according to the master-slave-concept. In the master process *all* routines or modules which are necessary for the entire optimization are included. Thus, the master process is executable even without associate processors for sensitivity analysis. In contrast to the master, the slave process consists of those routines which

are necessary for the pass of the optimization loop (for instance, no optimization algorithm is included), and it has a local data base.

4 Application: Optimal layout of an automotive wheel

The automotive wheel to be optimized will be considered as a branched shell of revolution (unsymmetrical details as vent holes are not regarded). Fig. 5 shows the initial design of this wheel and one of the considered load cases (rolling bench test). In addition to the load case *rolling bench test*, the load case *rotating bending test* will be considered in the structural model. The latter load case is relevant for the design of the disk and for the connection of rim and disk. The degrees of freedom

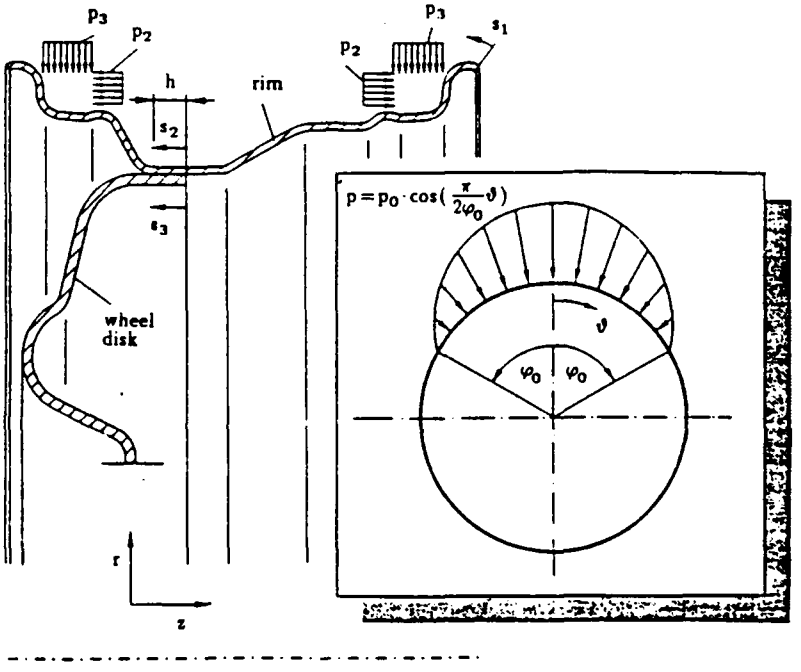


Figure 5: Initial design of the automotive wheel including a description of the load case rolling bench test

for finding the optimal design are the *thicknesses of rim and disk* (each constant), the *meridional shape of the disk* and the *meridional shape of the centrepart of the rim* (Fig. 5). For the description of the meridional shapes of the disk and the rim the approach functions "B-Spline ($k = 3$)" and "Coupled circular arc/straight lines" are used. The thicknesses and 10 control points of the shape functions are chosen as design variables. The weight is defined as the objective function and stress and deformation constraints as well as shape constraints are considered. Fig. 6 shows the optimal design of the automotive wheel for the given optimization model. The weight of the optimal design is 7.02kg which means a weight reduction of more than 30% in comparison to the feasible initial design (10.17kg, obtained by pure sizing). Then, the shape optimization problem of the wheel is solved by means of the decomposition methods described in the section 3.

Method 1: The parallel sensitivity analysis is applied. For that purpose, 0 to 12 associate processors will be used successively. Thus, we cover the whole range from sequential up to full parallel sensitivity analysis.

Method 2: Besides the parallelization of the sensitivity analysis the structural model will decomposed and analyzed by means of parallel substructuring. For this purpose, the wheel is cut off at the branch and partitioned into three substructures. Using this method the needed computer

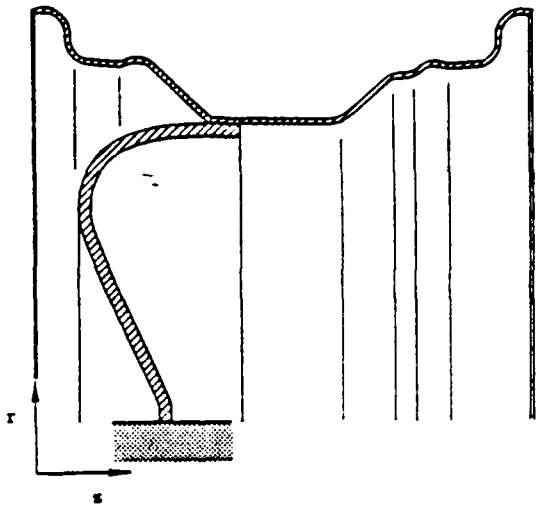


Figure 6: Optimal design W=7.02kg

memory is reduced in addition to the reduction of the computer time. The assignment of processors to the substructure processes (slaves) is fixed while we use variable numbers of processors for the sensitivity analysis (0 to 4 associate processors). Thus, for this method we use from 3 to 15 processors.

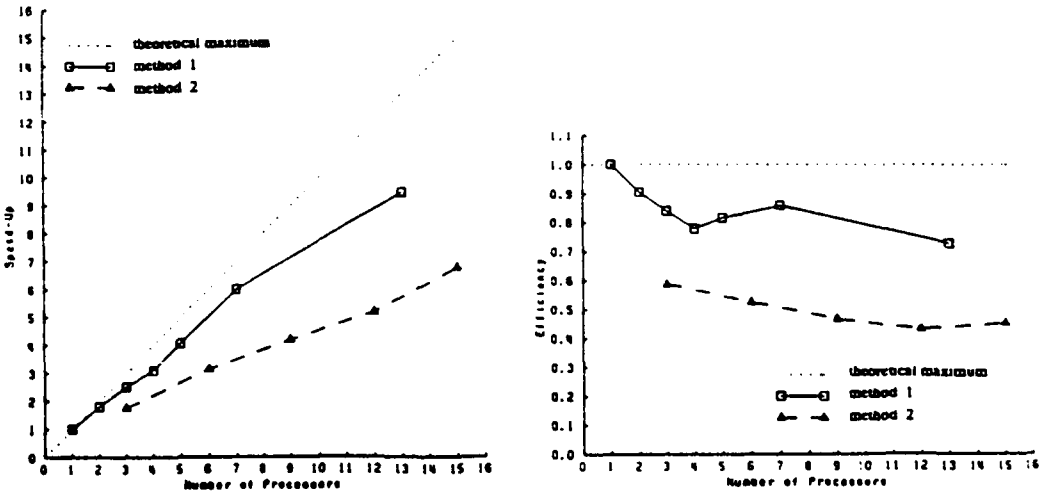


Figure 7: Speed-Up and efficiencies for method 1 and method 2

As shown in Fig. 7a, we save considerable computational time with both methods. For the method 2, the saving in computational time is less than for method 1 caused by the greater portion of sequential computations during the substructuring. For the valuation of these two methods however, one has to take into account that method 2 saves computation time *and* computer memory. The efficiency (efficiency=speed-up/number of used processors) is a mean value for the utilization of the processors. Concerning the efficiency, method 1 is also better than method 2. The non-monotonous course of the efficiencies in Fig. 7b is caused by the violation of condition (4). (except for 0 or 12 associated processors).

References

[Bat76] BATHE, K.; WILSON, E.L.: Numerical Methods in Finite Element Analysis. Prentice Hall, Inc., Englewood Cliffs, New Jersey, USA,1976.

[Blo90] BLOEBAUM, C.L.; HAJELA, P.; SOBIESZCZANSKI-SOBIESKI, J.: Non-Hierarchic System Decomposition in Structural Design. Third Air Force/NASA proc of the Symposium on Recent Advances in Multidisciplinary Analysis and Optimization, San Francisco, CA 1990.

[Boe84] BOEHM, W., FARIN, G ; KAHMANN, J.: A survey of curve and surface methods in CAGD. Computer Aided Geometric Design 1 (1984), 60.

[Bra84] BRAIBANT, V ; FLEURY, C.: Shape optimal design using B-splines Comput. Meths. Appl. Mech. Engrg. 44 (1984), 247-267.

[Bre89] BREMICKER, M.; ESCHENAUER, H.A.: Application of a Decomposition Technique for Treating a Shape Optimization Problem. Advances in Design Automation Vol. II, ASME Pub. DE-Vol 19-2, 1-6, 1989.

[Dan60] DANTZIG, G.B.; WOLFE, P.: Decomposition Principle for Linear Programs. Operation Research 8, 101 111, 1960.

[ElS91] EL-SAYED, M.E.M.; HSIUNG, C.-K.: Comparison between Structural and Computational Decomposition in Parallel Processing of Design Optimization. Advances in Design Automation, Vol. I. ASME Pub. DE-Vol. 32-1, 277-280, 1991.

[Esc91] ESCHENAUER, H.A.; WEINERT, M.: Structural Optimization Techniques as a Mathematical Tool for Finding Optimal Shapes of Complex Shell Structures To appea 1992 in the Proceedings of the Conference of Nonsmooth Optimization, Erice, Italy, 1991.

[Guy65] GUYAN, R.J.: Reduction of Stiffness and Mass Matrices. AIAA Journal, Vol 3, No. 4, 1965.

[Him73] HIMMELBLAU, D.M. (ed.): Decomposition of Large-Scale Problems. North-Holland, Amsterdam, 1973.

[Kir72] KIRSCH, U.; REISS, M.; SHAMIR, U.: Optimum Design by Partitioning into Substuctures. J. Struct. Div. ASCE 98, 249-261, 1972.

[Las70] LASDON, L.S.: Optimization Theory for Large Systems. Macmillan Co., London, 1970.

[NN90] N.N.: The CDL-Guide. Distributed Software Ltd., Bristol, UK. 1990.

[Top91] TOPPING, B.H.V.; KHAN, A.I.: Parallel Computations for Structural Analysis, Re-Analysis and Optimization. Proceedings of the NATO-Workshop "Optimization of Large Structural Systems", Berchtesgaden, Germany, October 23-24. 1991

[Wis71] WISMER, D.A.(ed.): Optimization Methods for Large-Scale Systems. M Graw-Hill, New York, 1971.

[Wu92] WU, B.C.; AZARM, S.: A reduction method for non-hierarchical optimization-based design. Proc. of ASME Design Automation Conference, 1992 (to appear).

On Multidimensional Partitioning Problems: Facial Structure and Applications

C.E. Ferreira M. Grötschel A. Martin
R. Weismantel

November 27, 1992

In this talk we deal with a subproblem arising in the design of a main-frame-computer. This problem can be stated as follows:
Let be given a set N of items and a set M of modules. Each item $i \in N$ has a weight f_i. Similarly, with every module $k \in M$ a capacity F_k is associated which represents the area of the module. Moreover, there is given a list of nets $\mathcal{Z} = \{T_1, \ldots, T_z\}$. Each net T_t is a subset of the set of items which has to be connected via a wire. Finally, every module $k \in M$ has a so-called cut capacity S_k, which restricts the number of nets that may leave this module. Roughly speaking, the problem we consider consists in finding an assignment of the items to the modules such that certain technical restrictions are satisfied and a very complicated objective function is optimized.

The most important constraints are the following:

> The sum of the weights of the items that are assigned to one module must not exceed its capacity.
>
> For every module $k \in M$ the following requirement must be satisfied: The total number of nets that contains an item assigned to k and some other item assigned to some module $l \in M \setminus \{k\}$ must be resticted by the cut capacity S_k of the module.

Let $a : N \to M$ denote some assignment of the items to modules such that the constraints are satisfied. The objective value of this assignment a is of

the form

$$\sum_{k \in M} K_k \cdot ic_k(a) + \lambda \cdot ec(a).$$

Let us first focus on the first term of the objective function, i.e., the internal cost. K_k is a constant which corresponds to the fabrication cost for module k. Roughly speaking, the internal cost of some module depend on the number of wires that must be routed within this particular module. In fact, the internal costs are a staircase function. This is due to the fact that every module consits of several layers on which the routing of the wires talkes place. Depending on the number of wires that must be routed within some module, a certain number of layers is required. A jump of the stair case fuction occurs whenever additional layers for some module are necessary, since the number of wires exceeds a certain threshold. The external cost $ec(a)$ represent the number of nets running between different modules. The parameter λ is an estimation of the cost for one wire.

From a mathematical point of view this application has the flavour of both a packing aspect and a multi-partitioning aspect. The packing aspect arises from the fact that certain items must be assigned to modules such that given capacities are not exceeded. Similarly, one has to decide which nets are connected via a wire within which module such that the given cut capacities are still satisfied. On the other hand, the multi-partitioning aspect is present as well, since the number of nets connecting items which are assigned to different modules has a strong impact on the objective function.

We modell this problem as an integer program with linear objective function. Due to the very complex objective function we obtain a model which involves several clumsy and technical conditions. Moreover, for practical applications, the model requires several hundreds of thousands integer variables. Thus, we decided to study relaxations of this problem. Working in this scheme a first relaxation consists in the ***multiple knapsack problem***, which can be viewed as the task of assigning a given set of items to a given set of modules. Here, we introduce boolean variables x_{ik}, $i \in N, k \in M$ with the interpretation $x_{ik} = 1$, if item i is assigned to module k and $x_{ik} = 0$ otherwise. The relaxation considers just the assignment of items to modules such that the correponding area capacity is taken into account. The number of nets running between different modules as well as the cut capacity of the modules is completely neglected. The second relaxation extends the multiple knapsack

model in the sense that nets running between different modules are approximately taken into account. For every pair of items i, j such that there exists a net connecting both i and j, we introduce an additional 0/1-variables y_{ij} with the following interpretation: $y_{ij} = 1$ if i and j are assigned to different modules and $y_{ij} = 0$, otherwise. Using these variables we estimate the number of nets running between different modules by the number $\sum_{i,j \in N} y_{ij}$. This yields to a multidimensional graph partitioning problem. The last relaxation we are going to consider improves the approximation of the nets and leads to a multidimensional partitioning problem in a hypergraph. Rather than introducing variables y_{ij} between pairs of items, we associate a variable z_{tk} with every module $k \in M$ and every net T_t. The variable z_{tk} is set to 1, if a proper subset of the set of items T_t is assigned to module k and is set to zero, otherwise. This class of variables enables us to model the cut capacities of the modules as well as to count the number of nets running between different modules.

With each of these relaxations we associate a polyhedron whose vertices are in one to one correspondence to the feasible solutions of the proper model. Then, solving one of the models reduces to optimizing a linear objective function over the corresponding polyhedron. In order to apply linear programming techniques, we need a description of the polytope by means of equations and inequalities. Thus, a first step in solving these problems via a polyhedral approach consists in a concise study of the underlying polyhedra. In this talk we will report on the facial structure of the three relaxations as well. In particular, there is a nice relationship between facet-defining inequalities for the three polyhedra.

First, one can prove that every facet for the multiple knapsack polytope defines a facet for the multidimensional graph partitioning polyhedron. Thus, the facial structure of the multiple knapsack polytope is completely inherited by the multipartitioning graph polyhedron. Similarly, valid inequalities for the polytope associated with the second relaxation can be modified such that they are valid for an appropriate multidimensional hypergraph polytope. Unfortunately, not every facet for the multidimensional graph partitioning polytope is inherited by the corresponding hypergraph polytope. Yet, there are several examples where we may resort to facet-defining inequalities for the multidimensional graph partitioning polytope, and by modifying them, we obtain facets for the hypergraph polytope. An example of this kind is the so-called cycle inequality which we will discuss in the talk.

We conclude by giving some remarks on the relationship between the third relaxation and the original problem. Here, it turns out that the third relaxation takes the original side constraints into account. The only difference between the two of them is that the objective function in the hypergraph model is simplified and provides just a heuristic estimate of what is really to be minimized. However, if one is able to handle the multipartitioning hypergraph polytope from a theoretical as well as from a practical point of view, one could start with some objective function and optimize over this polytope. If the solution is feasible for the original model we stop. Otherwise, we modify the estimate of the objective function for the hypergraph model in a lagrangean fashion and repeat this process until we terminate with a globally feasible solution. Surely, the solution provided that way is not necessarily optimal for the original problem. However the objective function is somehow related to the original one and thus, an optimal solution to the relaxed model that is still feasible for the original one should be not too bad. In particular, one should expect that it meets the requirements, practioners are interested in.

At least from our point of view, this type of approach (providing a series of reasonable relaxations to a very complex problem and handling the relaxations theoretically as well as practically) is best what one can expect, since theoretical and practical progress up to date is still far away from solving large scale real world problems to optimality, which are as complex as problems occuring in the design of main frame computers.

Contoured Beam Reflector Array Antenna Optimization with Dual Parameter Types

P.E.Frandsen (*, +) and S.B.Sørensen (*)
(*) TICRA, DK-1114 Copenhagen, Denmark
(+) The Technical University of Denmark,
DK-2800 Lyngby, Denmark

1. Introduction

Spaceborne communication antennas are often required to illuminate an irregular coverage region on the earth. To achieve this effectively the radiated beam is shaped in order to concentrate the power on the region. A shaped beam also known as a contoured beam can be obtained with an offset parabolic reflector with multi-feed array shown in figure 1.1. This antenna generates a set of small element beams. Each element beam is generated by a feed that radiates towards the reflector and the element beam is the reflected secondary field. Each feed is a small metal horn which transmits to free space. The feeds are arranged in an array such that the corresponding element beams together cover the region (see figure 1.1). The antenna input is transformed into feed excitations i.e. input amplitudes and phases by a beam forming network. Thus the element beams are excited correspondingly and the contoured beam is obtained.

The classical contoured beam optimization problem is then given the antenna to adjust the excitations to maximize the power gain within the coverage. Additionally, isolation regions may be included in which the power level shall be suppressed. Several procedures have been proposed for this problem including minmax formulations to maximize the minimum power gain ([1], [2] & [3]).

Here we shall describe a procedure in which the feed array parameters are included into the optimization. The shape of an element beam is highly dependent on the feed aperture. Feeds with circular or squared apertures create a power gain pattern with circular contours. Rectangular feeds leads to elongated element beams (figure 1.2). Because of size, weight and losses it is desirable to keep the number of feeds as small as possible. A well-fitting contoured beam can be obtained with a limited number of non-circular element beams. The size and position of the corresponding feeds are usually found by hand. As the number of parameters characterizing the feeds is large it is likely that a manually adjusted array is far from optimum.

2. Field calculations

The secondary far field from a separate rectangular feed must be calculated in different far-field directions. The parameters of a rectangular feed are the aperture dimensions a and b and the position of the aperture centre x_f and y_f in the focal plane (Fig. 3.1).

The focal plane coordinates are denoted x_f,y_f,z_f and the corresponding basis vectors $\mathbf{x}_f$, $\mathbf{y}_f$, $\mathbf{z}_f$. A unit direction vector is defined by $\mathbf{r}_f=(u_f,v_f,w_f) = (x_f,y_f,z_f)/r_f$, where $r_f = (x_f^2+y_f^2+z_f^2)^{1/2}$. The radiated electrical field from the feed can be written as

$$\mathbf{E}_f = A\frac{e^{-jkr_f}}{r_f}\left(\hat{\gamma}_f w_f - \left(\mathbf{x}_f\cdot\hat{\gamma}_f u_f + \mathbf{y}_f\cdot\hat{\gamma}_f v_f\right)\mathbf{z}_f\right)\cdot f(u_f,v_f)$$

where A is a normalization constant, j is the imaginary unit ($j^2=-1$) and k is the wave number related to the speed of light c and the frequency ν by $k=2\pi\nu/c$. The unit vector $\hat{\gamma}_f$ is either equal to $\mathbf{x}_f$ or $\mathbf{y}_f$ depending on the feed polarization. The function $f(u_f,v_f)$ is the Fourier transform of the aperture field of the feed denoted $h_f(x_f,y_f)$, where a simple model is used

$$h_f(x_f, y_f) = \begin{cases} \cos(\pi y_f / b) & \text{for } -a/2 < x_f < a/2 \\ & \text{and } -b/2 < y_f < b/2 \\ 0 & \text{otherwise} \end{cases} \qquad (2.2)$$

Next, the magnetic field $\mathbf{H}$ radiated from the feed is found from $\mathbf{H} = \mathbf{r}_f \times \mathbf{E}_f$. On the reflector surface a current distribution $\mathbf{J}(x,y,z)$ is induced, calculated by the so-called physical optics approximation, $\mathbf{J}(x,y,z) = 2\mathbf{n} \times \mathbf{H}$, where $\mathbf{n}$ is the unit surface normal. The field and currents are here considered to be functions of the coordinates x,y,z of the antenna coordinate system (see Figure 1.1). Hereafter the secondary field radiated by the reflector can be found from ([6])

$$\mathbf{E}_{far} = \iint_A \left(\mathbf{J}(\mathbf{r}) - \left(\mathbf{J}(\mathbf{r}) \cdot \hat{\mathbf{r}} \right) \cdot \hat{\mathbf{r}} \right) e^{jk\mathbf{r} \cdot \hat{\mathbf{r}}} \, ds \qquad (2.3)$$

which gives the electrical far field in the direction $\hat{\mathbf{r}}$. The vector $\mathbf{r}=(x,y,z)$ is the integration variable in the surface integral over the reflector area A. The quantities of interest are the polarization components e_{co} and e_{cx} obtained from $\mathbf{E}_{far}$ by the projections $e_{co} = \mathbf{E}_{far} \cdot \mathbf{e}^*_{co}$ and $e_{cx} = \mathbf{E}_{far} \cdot \mathbf{e}^*_{cx}$ where e_{co} and e_{cx} forms the desired polarization basis. (* denotes the complex conjugate.)

3. Array topology and minmax formulation

The feeds are mounted with their apertures in the focal plane (figure 1.1). During the optimization the apertures will vary in size and position. The feed array parameters cannot, however, vary independently since no aperture overlaps must occur. To avoid a considerable number of linear constraints we have chosen an approach where the array must be composed of a collocation of rows of rectangular apertures (Figure 3.1). Let the feed coordinate system in the focal plane have axes parallel to the aperture edges and let the rows be organized in the x_f-direction. Then all feed of an internal row must have the same height and y_f-coordinate for the aperture center

whereas the heights of the feeds in the top and bottom row can vary independently. All vertical edges of neighbouring feed apertures in a row must coincide and all horizontal edges of neighbouring rows must coincide. With this topology the independent variables are:

– One common height for all feeds in each internal row and individual heights for the feeds in the bottom and top rows.

– For each row the x_f-coordinate of the aperture center of one of the feeds and individual widths of all feed apertures.

– y_f-coordinate of the aperture center of a main reference feed and rotation of the complete array.

The maximum number of variables that can be used equals $N+2R+n_1+n_R$, where N is the number of feeds, R the number of rows and n_1 and n_R the number of feeds in the bottom and top row resp. This number may be reduced if identical feeds are required. Due to field model limitations bounds are needed on the aperture dimensions. Let $\mathbf{x}_a \in R^{n_a}$ be a vector with the chosen array variables, $n_a \leq N+2R+n_1+n_R$.

The desired power gain is specified over a set of synthesis stations adequately sampled to define the coverage and isolation regions. Let the complex number $e_{ij}(\mathbf{x}_a)$ denote the far field at the i'th station of the j'th element beam found as discussed in section 2 excited by unit amplitude and zero phase. The complex vector $\mathbf{e}_i(\mathbf{x}_a)$ C holds the values from the N elements. Further, let α_j denote the j'th complex excitation where $Re(\alpha_j) = A_j\cos(Ph_j)$ and $Im\alpha_j = A_j\sin(Ph_j)$, A_j and Ph_j being the j'th excitation amplitude and phase. The complex excitations are elements of the vector $\underline{\alpha} \in C^N$. Since the optimization is performed in real variables we use $\mathbf{x}_e \in R^{2N-1}$, such that $\alpha_j(\mathbf{x}_e) = (x_{2j-1} + i\, x_{2j})$, j=1,..,N-1 and $\alpha_N(\mathbf{x}_e) = (x_{2N-1} + i\, 0)$. Therefore 2N-1 independent variables are available, since phase is a relative quantity. The total vector of independent variables is then the concatenation $\mathbf{x} = \mathbf{x}_a // \mathbf{x}_e$, $\mathbf{x} \in R^n$, $n=2N-1+n_a$.

With the above notation the power gains at the station can be expressed as

$$p_i(x) = \frac{(\underline{a}(x_e)\cdot \underline{e}_i(x_e))(\underline{a}(x_e)\cdot \underline{e}_i(x_e))^*}{\underline{a}(x_e)\cdot\underline{\tilde{a}}(x_e)^*}, \quad i=1,\ldots,M \qquad (3.1)$$

where M is the total number of stations. (See [3]). For each synthesis station a residual function is now defined which is the difference between the realized gain normalized by the factor G_o and a specified relative gain goal p_{io}, i.e.

$$f_i(x) = w_i(p_i(x)/G_o - p_{io}) \qquad (3.2)$$

where G_o should be slightly above the expected peak power gain. The specified relative power gain is used to express the station levels of the desired pattern, such that $p_{io} \approx 1$ for a coverage residual and $p_{io} \approx 0$ for an isolation residual. The weights are used to equalize the size of coverage and isolation residuals. The minmax problem to be solved consist of determining $\hat{x} \in R^n$ which minimizes the maximum residual -

$$F(\hat{x}) = \min_{x \in R^n} F(x) = \min_{x \in R^n} \max_{1<i<M} |f_i(x)| \qquad (3.3)$$

4. Solution of the minmax problem

The problem (3.3) is solved by the approximate gradient version of the general minmax method of Madsen [4]. This is an iterative trust region method. In each iteration the residuals are linearized and the linear model function is minimized subject to a bound on the solution. The proposed step is accepted as the next iterant if F decreases. Otherwise the step is repeated with a reduced bound. To solve the problem on small computers some additions were needed. In each step worst and near worst case residuals are identified and then only these are linearized. Thus the storage needed is reduced by approx. 80%. Gradient approximations are obtained by a combi-

nation of Broyden's rank one formula and differences. The linear model function is minimized by the exchange method of Powell [5]. In a re-designed implementation an option for starting with a good guess of the final set of active linearized residuals defining the solution has been added. Then, in the non-linear minimax the set of active linearized residuals found in one step is used as the guess for final set of the next.

5. An example

One of the test cases for the procedure was the Brazilsat Antenna System discussed by Patel & Chan [2]. The requirements were 27 dBi for a high gain region shown as a polygon on figure 5.2 and 25 dBi for the rest of Brazil. (dBi is the power level above isotropic level in dB ($=10\log|p_i|$)). The antenna consists of an 1803 mm offset reflector illuminated by 6 rectangular feeds as shown in figure 5.1. A total of 97 synthesis stations were used and the total number of variables was 25. The original array and excitations were used as initial point for the iteration.

A minmax optimization using only excitations as variables yields 28.61 dBi and 26.61 dBi for the high and low gain zones resp. (Fig. 5.2). The result from the optimization with array parameters included is shown in figure 5.3. The power gains are 29.74 dBi and 27.74 dBi for the two zones.

6. Concluding remarks

General methods for non-linear minmax problems have been used successfully for contoured beam antenna optimization with the excitations of the feeds as variables. If the array consists of different sized rectangular feeds the performance of the antenna can be improved further, if the feed array parameters are used as variables together with the excitations.

7. Acknowledgments

This work was supported by the European Space Agency under ESTEC Contr. 7884/88/NL/PB(SC). We acknowledge the discussions with R. Jørgensen, Prof. K. Madsen and Ian Roberts.

8.References

[1] J. R. Mautz & R. F. Harrington, "Computational methods for antenna pattern synthesis", IEEE Trans. Ant. Propagat., Vol. AP-23, 1975

[2] P. D. Patel & K. K Chan, "Optimization of contoured beams for satellite antennas", IEE Proc., Vol. 132, Pt. H, 1985

[3] P. Balling, P. E. Frandsen & R. Jørgensen, "Development of Several Constraint Optimization procedure for Spacecraft Systems", TICRA Report S-234-02, Copenhagen 1981.

[4] K. Madsen, 'Minmax solution of non-linear equations without calculating derivatives", Math. prog. Study, Vol. 3, 1975.

[5] M. Powerll, "The minmax solution of linear equations subject to bounds on the variables", Rep. CSS11, AERE harwell, Oxfordshire, UK, 1974.

[6] R. E. Collin & F. J. Zucker, "Antenna Theory, Part 1", McGraw-Hill Book Company, New York, 1969.

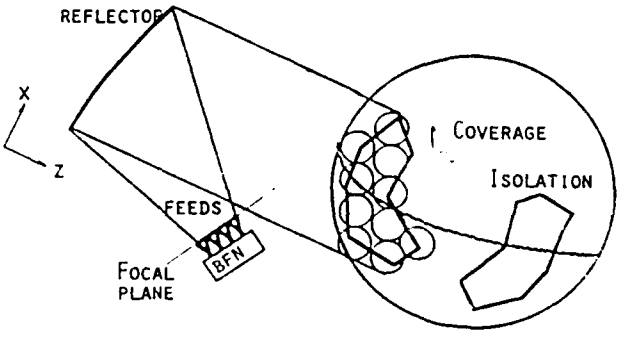


Figure 1.1

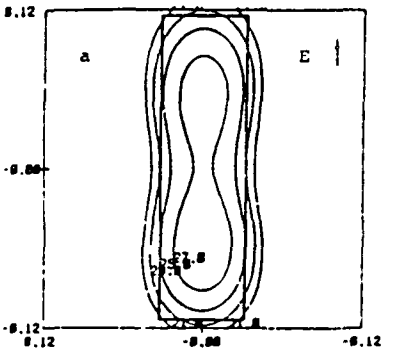


Figure 1.2 Secondary far field from rectangular feed

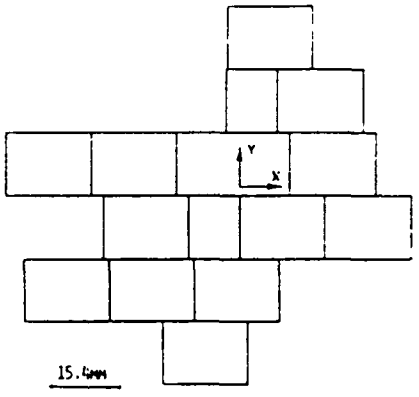


Figure 3.1 Rows of feed apertures

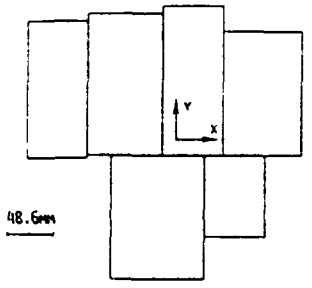


Figure 5.1 Original Brazilsat feed array

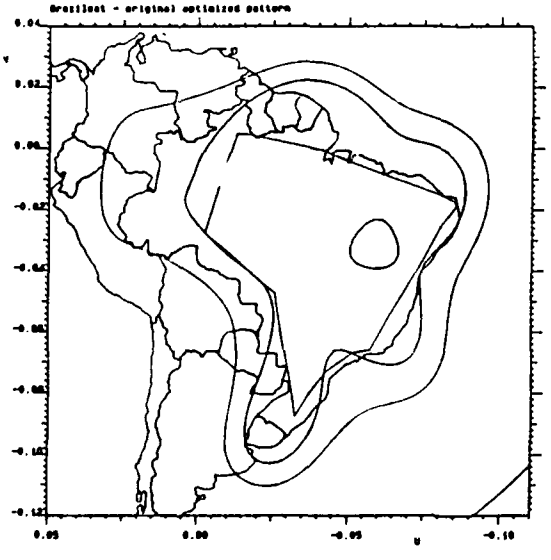


Figure 5.2 Optimized with excitation variables

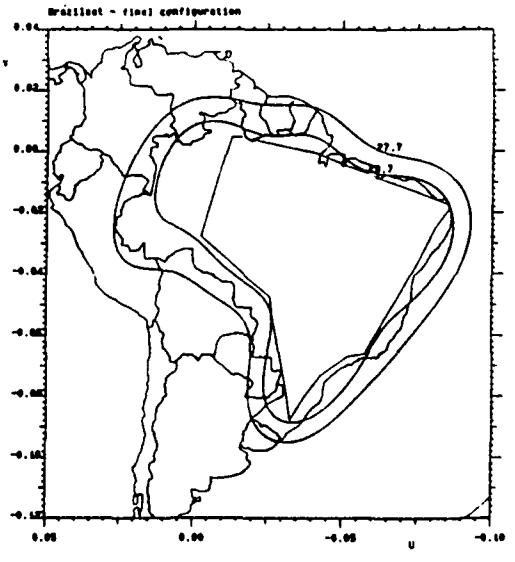


Figure 5.3 Optimized with excitation s and feed array parameters

APMOD93 Budapest

A Lagrangean relaxation for optimizing a discrete production system in manufacturing

Arnaud FREVILLE

Saïd HANAFI

University of Valenciennes and of Hainaut-Cambresis
Institut of Sciences and Technology
BP 311. 59304 France.

Abstract :

We study a production management in clothing industry. The model we use is a large scale linear integer programming problem with a lot of structure. Lagrangean relaxation method coupled with heuristics yields a good bracketing of optimal solution by dualizing the state equations. Subgradient technique is used to solve Lagrangean dual, each iteration reduces to exact solving of knapsack problems.

Key words : Lagrangean relaxation, integer linear program, production problem, subgradient optimization.

1. Introduction

We consider a problem of production management in process industry application. This problem can be formulated as a large-scale linear integer programming problem strongly structured.

The paper develops a Lagrangean relaxation technique that successively solves a sequence of generalized knapsack problems. A near optimal solution is obtained via a subgradient method coupled with an heuristic. Numerical experiments with real and randomly generated instances are in progress to validate the approach.

Section 2 deals with a short description of the production system and a deterministic model formulation. Section 3 describes a suitable strategy for large-scale instances: each iteration of an ad hoc Lagrangean relaxation reduces to exact solving of small-size knapsack problems.

2. Model formulation

We study an inventory production management problem in the clothing industry. An effective production planning system determines the appropriate levels of production and inventory according to fluctuating demand requirements and minimum costs.

Generally speaking, a manufacturing production system in the clothing industry can be viewed as a sequence of transformations applied to raw materials to obtain finally a finished product.

In the formulation, external subsystems such as extraction and transportation of the raw materials are neglected. Then the process can be subdivised into three subsystems (see figure) :

- transformation of the raw materials into raw pieces;
- shaping of the raw pieces into shaped pieces;
- assembling of the intermediate subproducts to provide the finished items.

Each subsystem is characterized by the following decision variables: vector of products, stocks and demands. We assume that a discrete deterministic model is available and that external demands of the global system are given ([4],[8]).

Let N denotes the number of items to fabricate, M cardinality of E_M set of raw materials index, J cardinality of E_J references set of elementary pieces forms, $L \subseteq E_J \times E_M$ the number of intermediate subproducts of transformation and shaping subsystems, T the number of periods in the planning horizon and k=3 the number of subsystems.

The following underscripts are also used in the description of the model.

k : index of system (k = 1,2,3);
t : time period (t = 1,..., T);
i : index of finished product if k=1 (i=1,...,N) or of intermediate subproduct if k = 2,3 (i = 1,...,L).

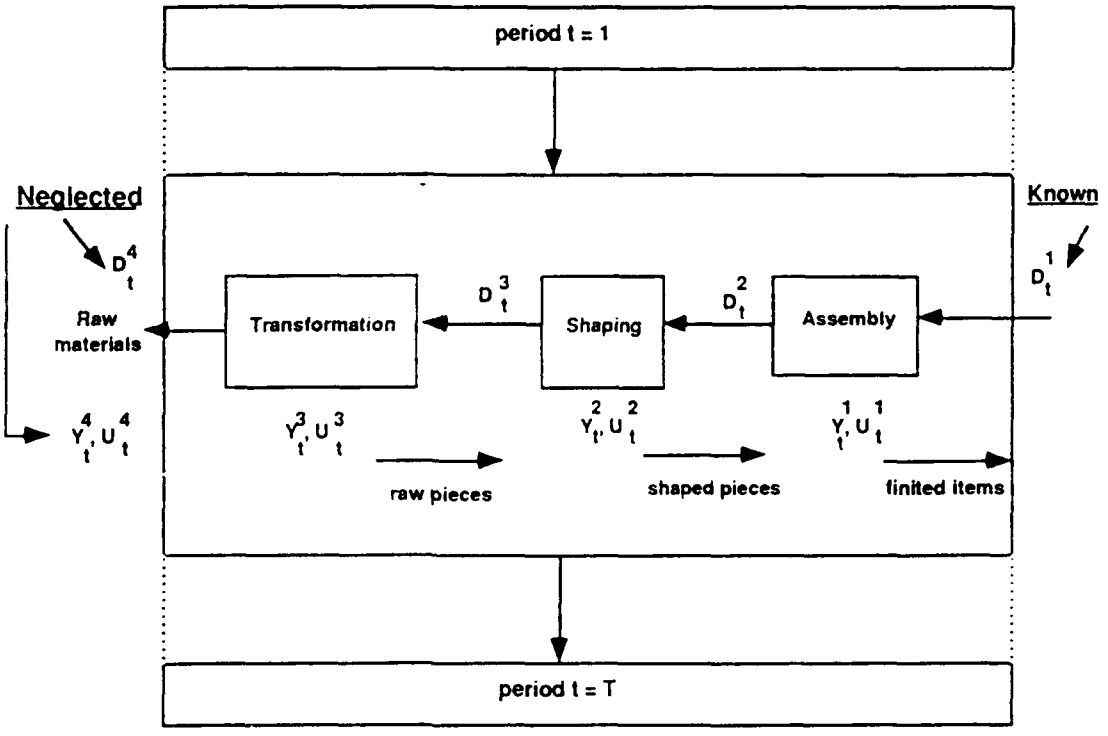
We now introduce vectors of decision variables of the problem :

$Y_t^k = [\, y_{i,t}^k \,]$: number of product i to be produced during period t in subsystem k.

$U_t^k = [\, u_{i,t}^k \,]$: number of storage i during period t in subsystem k.

$D_t^k = [\, d_{i,t}^k \,]$: demand of product i during period t in subsystem k (k=2,3)

$D_t^1 = [\, d_{i,t}^1 \,]$: given external demand of product i during period t .



Planning Horizon of the manufacturing production system

figure : Discrete production system in manufacturing

The following vectors of data are required for formulation

$R_t^k = [\, r_{i,t}^k \,]$: availability factor of product i of subsystem k during period t.

$S_t^k = [\, s_{i,t}^k \,]$: availability factor of storage i of subsystem k during period t.

φ_t^k : resource production availability of subsystem k during period t.

ψ_t^k : storage of resource availability of subsystem k during period t.

$Y_t^k = [\, \bar{y}_{i,t}^k \,]$: maximum of production allowed per period for item i of subsystem k.

$\underline{U}_t^k = [\, \underline{u}_{i,t}^k \,]$; $\overline{U}_t^k = [\, \overline{u}_{i,t}^k \,]$: lower and upper bounds for storage of product i from subsystem k during period t.

$C_t^{1k} = [\, c_{i,t}^{1k} \,]$: unit production cost over period t for item i of subsystem k.

$C_t^{2k} = [\, c_{i,t}^{2k} \,]$: unit storage cost over period t for item i of subsystem k.

$C_t^{3k} = [\, c_{i,t}^{3k} \,]$: cost of modifications when fabrication changes from a period to another.

and the matrix $\Omega = (\alpha_l^i)$:

α_l^i : number of necessary pieces l to fabricate item i.

The constraints of the problem are either technological or economical. These constraints and production cost are assumed to be linear. We suppose that the objective function is also linear according to product ($y_{i,t}^k$), stock ($u_{i,t}^k$) and change of activity level from a period to another ($y_{i,t}^k - y_{i,t-1}^k$). Hence the model can be formulated as the following large-scale integer linear program with 2(N+3L)T variables and (3N+8L+6)T constraints:

$$\min \quad z = \sum_{k=1}^{3} \sum_{t=1}^{T} C_t^{1k} Y_t^k + C_t^{2k} U_t^k + C_t^{3k} (Y_t^k - Y_{t-1}^k)$$

s.t.

$$R_t^k Y_t^k \le \varphi_t^k \qquad \text{(k-1)}$$

$$0 \le Y_t^k \le \Upsilon_t^k \qquad \text{(k-2)}$$

$$S_t^k U_t^k \le \psi_t^k \qquad \text{(k-3)}$$

$$\underline{U}_t^k \le U_t^k \le \overline{U}_t^k \qquad \text{(k-4)}$$

$$U_t^k = U_0^k + \sum_{\theta=1}^{t} \{ Y_\theta^k - D_\theta^k \} \qquad \text{(k-5)}$$

$$D_t^2 = \Omega\, Y_t^1 \qquad \text{(2-6)}$$

$$D_t^3 = Y_t^2 \qquad \text{(3-6)}$$

Y_t^k , U_t^k , $D_t^{2,3}$ integer k=1,2,3 t=1,...,T

The initial conditions U_0^k and Y_0^k are given and set to zero and we assume $C_t^{33}=0$.

3 Lagrangean approach

The coupling constraints (2-6) and (3-6) that link the subsystems are used to eliminate only the decision variables $D^{2,3}$. After rearranging terms, the problem can be stated as an equivalent linear program with 2(N+2L)T variables and (3N+6L+6)T constraints :

$$
(P)\quad
\begin{cases}
\min \displaystyle\sum_{k=1}^{3}\sum_{t=1}^{T}\{F_t^{1k}Y_t^k + C_t^{2k}U_t^k\} & \\
\text{s.t.} & \\
R_t^k Y_t^k \le \varphi_t^k & \\
0 \le Y_t^k \le \bar{Y}_t^k & \\
S_t^k U_t^k \le \psi_t^k & \\
\underline{U}_t^k \le U_t^k \le \bar{U}_t^k & \\
U_t^1 = \displaystyle\sum_{\theta=1}^{t}\{Y_\theta^1 - D_\theta^1\} & (1\text{-}5)\\
U_t^2 = \displaystyle\sum_{\theta=1}^{t}\{Y_\theta^2 - \Omega\, Y_\theta^1\} & (2\text{-}5)\\
U_t^3 = \displaystyle\sum_{\theta=1}^{t}\{Y_\theta^3 - Y_\theta^2\} & (3\text{-}5)\\
Y_t^k,\ U_t^k \ \text{integer} & k=1,2,3 \quad t=1,..,T
\end{cases}
$$

with $F_t^{1k} = C_t^{1k} + C_t^{3k} - C_{t+1}^{3k}$ and $C_{T+1}^{3k} = 0$

By dualizing the state equations ((k-5), k =1,2,3) with a multiplier V one obtains the Lagrangean relaxation subproblem :

$$
(LR(V))\quad
\begin{cases}
\min\ cste + \displaystyle\sum_{k=1}^{3}\sum_{t=1}^{T}\{G_t^{1k}Y_t^k + G_t^{2k}U_t^k\} & \\
\text{s.t.} & \\
R_t^k Y_t^k \le \varphi_t^k & \\
0 \le Y_t^k \le \bar{Y}_t^k & \\
S_t^k U_t^k \le \psi_t^k & \\
\underline{U}_t^k \le U_t^k \le \bar{U}_t^k & \\
Y_t^k,\ U_t^k \ \text{integer} & k=1,2,3 \quad t=1,..,T
\end{cases}
$$

where $V = (V_t^1, V_t^2, V_t^3)$ $\quad V_t^k = [v_{i,t}^k]$ $\quad cste = -\displaystyle\sum_{t=1}^{T} V_t^1 \sum_{\theta=1}^{t} D_\theta^1$

$$G_t^{11} = F_t^{11} + \sum_{\theta=t}^{T}\{V_\theta^1 - V_\theta^2\Omega\} \qquad G_t^{12} = F_t^{12} + \sum_{\theta=t}^{T}\{V_\theta^1 - V_\theta^2\Omega\}$$

$$G_t^{13} = F_t^{13} + \sum_{\theta=t}^{T} V_\theta^3 \qquad G_t^{2k} = C_t^{2k} - V_t^k$$

Each subproblem (LR(V)) can be decomposed into 6T knapsack problems over period t, subsystem k and variables Y_t^k and U_t^k. The first 3T subproblems are knapsack problems with upper bounded variables

$$([LR1(V)]_t^k) \qquad \begin{cases} \min \; G_t^{1k} Y_t^k & \\ \text{s.t.} & k=1,2,3 \\ \quad R_t^k Y_t^k \le \varphi_t^k & t=1,..,T \\ \quad 0 \le Y_t^k \le \bar{Y}_t^k & \end{cases}$$

and the 3T other subproblems are knapsack problems with lower and upper bounded variables :

$$([LR2(V)]_t^k) \qquad \begin{cases} \min \; G_t^{2k} U_t^k & \\ \text{s.t.} & k=1,2,3 \\ \quad S_t^k U_t^k \le \psi_t^k & t=1,..,T \\ \quad \underline{U}_t^k \le U_t^k \le \bar{U}_t^k & \end{cases}$$

The corresponding Lagrangean dual given by

$$(D) \qquad \begin{cases} \max \; v(LR(V)) \\ \text{s.t.} \quad (V) \end{cases}$$

where v(.) denotes the optimal value of problem (.), is solved by a subgradient algorithm ([3]) and v(D) is a lower bound of v(P).

4 Preliminary conclusions

This approach is a suitable alternative to the one provided by Soenen ([8]). In his paper, the size of the model is reduced to (N + 2L)T variables and (2N + 4L+6)T constraints by using the state equations (k-5) and coupling constraints (2-6)-(3-6) to eliminate the variables $U^{1,2,3}$ and $D^{1,2}$. Then the Dantzig-Wolfe decomposition ([2]) is applied to solve the LP-relaxation and an integer feasible solution is obtained by rounding the LP-optimal solution. The author has also suggested to split the problem by creating copies of the original variables Y_t^2 of the shaping subsystem. He was among the first ones to suggest the idea of variables splitting and later Guignard and Kim ([6],[7]) formalized this idea to a general mixed-integer programming problem. But in this case, the subproblems induced by the dualization of the copy constraint do not reduce to knapsack problems. Though our model has twice more variables, the approach is attractive when the size increases. First it is well-known that the bound provided by the Lagrangean dual is generally tighter than the LP one. Secondly, at each iteration of the subgradient algorithm, the subproblems are knapsack of small-size and easy to solve exactly. Moreover feasible solutions can be constructed, starting from the optimal solution of each subproblem, to furnish a bracket of the optimal solution of the initial problem (P).

The algorithm was implemented in GAMS 2.25 (General Algebriac Modeling System [1]) with solvers ZOOM or LAMPS. First numerical experiments show that our approach is suitable for large-scale instances.

References :

[1] A. Brooke, D. Kendrick, A. Meeraus, "GAMS: a user's guide", The Scientific Press, (1992).
[2] G.B. Dantzig, P. Wolfe, "The decomposition algorithm for linear programming", Econometrica 29, p. 767-778 (1961).
[3] M.L. Fisher, "The Lagrangean relaxation for solving integer programming problems", Management Science, 27, p. 1-18, (1981).
[4] A. Fréville "Optimisation combinatoire en gestion de production", Colloque Franco-Québecois sur l'enseignement et le transfert des technologies de production assistée par ordinateur, ENSAM Lille, (1988).
[5] A.M. Geoffrion, "Lagrangean relaxation for integer programming", Mathematical Programming Study 2, p. 82-114, (1974).
[6] M. Guignard, S. Kim, "Lagrangean decomposition for integer programming : theory and applications", RAIRO 21, p. 307-324, (1987a).
[7] M. Guignard, S Kim, "Lagrangean decomposition a model yielding stronger Lagrangean Bounds" Mathematical programming, 32, p. 215-228, (1987b).
[8] R. Soenen : "Contribution à l'étude des systèmes de conduite en temps réel en vue de la commande d'unités de fabrication", Thèse d'état, Lille, (1977).

APMOD93 Budapest

RELAXATIONS FOR MINIMAX PROBLEMS

Arnaud FREVILLE †

Monique GUIGNARD ††

† University of Valenciennes and of Hainaut-Cambrésis
Institut of Sciences and Technology
BP 311 59304 France

†† The Wharton School of the University of Pennsylvania
1320 Steinberg Hall-Dietrich Hall.
Philadelphia, PA 19104-6366 U.S.A.

Abstract

We consider various relaxations and/or decompositions for solving linear integer minimax problems. The main results concern the comparison of the bounds they provide and necessary and sufficient conditions to obtain sharper bound or null duality gap with Lagrangean decomposition.

1 Introduction

Confronted with the problem of minimizing, in integers, the maximum of several functions, one usually introduces an extra variable, say y, to be minimized, and writes constraints which force y to be no less than these functions. These new contraints destroy whatever structure the problem had initially, and render its resolution much harder. One can obtain lower bounds on the optimal value of y by relaxing these constraints and then optimize the bound thus obtained. We will consider several relaxations and compare the bounds they provide. We will also study some specific minmax models and provide preliminary computational evidence on the quality of the bounds.

In section 2 we show how various relaxations and decompositions compare in terms of the bound they provide when only the new constraints are dualized. In section 3 we consider a two-level relaxation scheme, where complicating constraints of the initial structure require relaxation. An illustration of the main results is provided by an example in section 4.

Notation

We shall use the following notation. Given a constrained optimization problem $(.)$, $(\bar{.})$ will denote its continous relaxation, FS(.) its feasible set, OS(.) its optimal set, i.e. the set of all its optimal solutions, and $v(.)$ its optimal value. Co(S) will denote the convex hull of a set S of $\mathbb{R}^n$ and the superscript t transposition.

2. Minimax constraints dualization

Consider the following linear minmax problem (P)

$$(P) \qquad \left[\begin{array}{l} \min \ \max \ F_i(x) \\ x \in \mathcal{S} \ \ i=1,p \end{array} \right.$$

where $\mathcal{S}$ is a discrete subset of $\mathcal{R}^n$ and the F_i's ,i = I,...,p, linear functions of the form $F_i(x) = f_ix+g_i$.

Let us now introduce a new variable δ to represent the maximum of the p functions $F_i(x)$, and let us rewrite (P) as

$$(P) \qquad \left[\begin{array}{l} \min \ \{ \delta \mid Fx+g \leq \delta e, x \in \mathcal{S} \} \\ x,\delta \end{array} \right.$$

where F is the p×n matrix $[f_i]_{i=1,p}$, g the vector $(g_i)_{i=1,p}$ and e the all one vector (1,...,1) of $\mathcal{R}^p$. The new constraints $Fx+g \leq \delta e$ destroy whatever structure the set $\mathcal{S}$ had initially, and render the resolution of problem (P) much harder.

The first basic results concern the comparison of bounds provided by various relaxations which are obtained by only dualizing the minimax constraints $Fx+g \leq \delta e$.

We introduce the set of multipliers $\mathcal{U} = \{ u \in \mathcal{R}^p \mid u \geq 0, \sum_{i=1}^{p} u_i = 1 \}$ and the maxmin dual problem (Q)

$$(Q) \qquad \left[\begin{array}{l} \max \ \min \ u(Fx+g) \\ u \in \mathcal{U} \ \ x \in \mathcal{S} \end{array} \right.$$

As $\mathcal{S}$ is not a convex set, the classical minimax inequality holds and we have the following inequalities

$$\begin{aligned} \upsilon(P) &= \min \{ \max \{ F_i(x) \mid i = 1,...,p \} \mid x \in \mathcal{S} \} \\ &= \min \{ \max \{ u(Fx + g) \mid u \in \mathcal{U} \} \mid x \in \mathcal{S} \} \\ &\geq \max \{ \min \{ u(Fx + g) \mid x \in \mathcal{S} \} \mid u \in \mathcal{U} \} = \upsilon(Q) \end{aligned}$$

Hence the minimax duality gap $\sigma = \upsilon(P) - \upsilon(Q)$ is due to the nonconvexity of $\mathcal{S}$. Compactness and convexity of $\mathcal{S}$ are sufficient but not necessary conditions to have $\sigma = 0$. When $\mathcal{S}$ is compact, arguments based on Lagrangean duality (Geoffrion, [3]) lead to the equivalence between (Q) and the following linear program (P*)

$$(P^*) \qquad \left[\begin{array}{l} \min \ \{ \delta \mid Fx+g \leq \delta e, x \in Co(\mathcal{S}) \} \\ x,\delta \end{array} \right.$$

Therefore the duality gap σ can be positive only when the feasible set $\{ \delta \mid Fx+g \leq \delta e, x \in Co(\mathcal{S}) \}$ has noninteger vertices.

We now compare the optimal value $\upsilon(Q)$ with lower bounds of (P) provided by three different relaxations of the minimax constraints $Fx+g \leq \delta e$.

Problem (P) is equivalent to (P°), in which we create multiple copies of δ as a p-vector z :

$$(P^\circ) \qquad \left[\begin{array}{l} \min \\ x,z \end{array} \right. \{ \tfrac{1}{p} e^t z \mid Fx+g \le z,\ x \in \mathcal{S},\ z = Ez \}$$

where E is a (p,p) cyclic permutation matrix such that: $E^j_i = \begin{cases} 1 & \text{if } j=i+1,\ i=1,\ldots,p-1 \\ 1 & \text{if } j=1,\ i=p \\ 0 & \text{otherwise} \end{cases}$

We now dualize the copy constraint z=Ez with multiplier u and obtain the Lagrangean decomposition subproblem

$$(LD(u)) \qquad \left[\begin{array}{l} \min \\ x,z \end{array} \right. \{\varphi(u)z \mid Fx+g \le z,\ x \in \mathcal{S} \}$$

where φ is the linear function on $\mathcal{R}^p$ defined by $\varphi(u) = u(I-E)+\frac{1}{p}e^t$ and I the identity matrix.

The corresponding Lagrangean dual (LD) is

$$(LD) \qquad \left[\begin{array}{l} \max \\ u \end{array} \right. \boldsymbol{\upsilon}(LD(u))$$

where $\boldsymbol{\upsilon}(LD(u)) = \sum_{j=1}^{p} \min \{ (\frac{1}{p} + u_j - u_{j-1}) z_j \mid F_j x + g_j \le z_j,\ x \in \mathcal{S} \}$

We also define the Lagrangean and the surrogate duals of (P) relative to the minimax constraints $Fx+g \le \delta e$:

$$(LR) \qquad \left[\begin{array}{l} \max \\ u \ge 0 \end{array} \right. \boldsymbol{\upsilon}(LR(u))$$

where $\boldsymbol{\upsilon}(LR(u)) = \min\{ \delta + u(Fx+g-\delta e) \mid x \in \mathcal{S} \} = ug + \min \{ (1-ue)\delta + uFx \mid x \in \mathcal{S} \}$,

and

$$(SD) \qquad \left[\begin{array}{l} \max \\ u \ge 0 \end{array} \right. \boldsymbol{\upsilon}(SD(u))$$

where $\boldsymbol{\upsilon}(SD(u)) = \min \{ \delta \mid u(Fx+g) \le \delta ue,\ x \in \mathcal{S} \}$.

The following theorem states that all the above mentioned duals of (P) provide identical lower bounds equal to $\boldsymbol{\upsilon}(Q)$

Theorem1 $\boldsymbol{\upsilon}(P) \ge \boldsymbol{\upsilon}(Q) = \boldsymbol{\upsilon}(LD) = \boldsymbol{\upsilon}(SD) = \boldsymbol{\upsilon}(LR)$ ◊

To conclude the section, we give sufficient conditions under which the duality gap σ equals zero.

Proposition 2.

Let u^* be an extreme point of the convex set $\mathcal{U} = \{ u \in \mathcal{R}^p \mid u \ge 0,\ \sum_{i=1}^{p} u_i = 1 \}$ and let h denote the index such that $u^*_h = 1$. If there exists an optimal solution x^* of the relaxation $(Q(u^*))$: $\min \{u^*(Fx+g) \mid x \in \mathcal{S} \}$ such that: $\max \{(Fx^*+g)_i \mid i=1,\ldots,p\} = (Fx^*+g)_h$ then $u^* \in OS(Q)$, $x^* \in OS(P)$ and $\sigma = 0$. ◊

3. Integer problems with special structure

In the section 2, we have assumed that the relaxed problems max{ $f_u(x) \mid x \in \mathbf{S}$ }, for linear functions $f_u(x) = u(Fx+g)$, could be solved, at least reasonably easily, so that no relaxation of the constraints defining $\mathbf{S}$ was necessary. In this section, we shall consider cases for which $\mathbf{S}$ contains so-called "complicating" constraints, requiring relaxation of some of the constraints in $\mathbf{S}$. The bounds may not be as strong as those described above, because of the two-level relaxation, yet they may still provide one of the best approaches to solve such problems.

So we consider now the case where the constraint set $\mathbf{S}$ can be partitioned into two subsets $\mathbf{S} = \{x \mid Ax \le b, Cx \le d, x \in \Omega\}$, where $\Omega \subset \mathcal{R}^m \times \mathcal{Z}^{n-m}$ is a discret subset and $Ax \le b$ *are the complicating constraints. The problem (P) is then equivalent to*

$$(P) \qquad \left[\begin{array}{l} \min\limits_{x,\delta} \{ \delta \mid Fx+g \le \delta e, Ax \le b, Cx \le d, x \in \Omega \} \end{array} \right.$$

First we compare the bounds obtained by relaxing the minimax constraints $Fx+g \le \delta e$ and the complicating constraints $Ax \le b$. The following relaxations differ in the way one dualizes the minimax constraints and the complicating constraints.

Problem (P) is equivalent to problem (P^i), i=1,2,3, in which we introduce multiple copies of δ as a p-vector $z = (z_1, z_2, \ldots, z_p)$, and one copy y of x :

$$(P^1) \qquad \left[\begin{array}{l} \min\limits_{x,y,z} \{ \frac{1}{p} e^t z \mid Fx+g \le z, Ay \le b, Cx \le d, x = y, z = Ez, x \in \mathcal{X}, y \in \mathcal{Y} \} \end{array} \right.$$

$$(P^2) \qquad \left[\begin{array}{l} \min\limits_{x,z} \{ \frac{1}{p} e^t z \mid Fx+g \le z, Ax \le b, Cx \le d, z = Ez, x \in \Omega \} \end{array} \right.$$

$$(P^3) \qquad \left[\begin{array}{l} \min\limits_{x,y,\delta} \{ \delta \mid Fx+g \le \delta e, Ay \le b, Cx \le d, x = y, x \in \mathcal{X}, y \in \mathcal{Y} \} \end{array} \right.$$

with $\mathcal{X} \cap \mathcal{Y} = \Omega$.

We dualize all the copy constraints in (P^1) and obtain a first Lagrangean dual

$$(LD^1) \qquad \left[\begin{array}{l} \max\limits_{u,\,v} \upsilon(LD^1(u,v)) \end{array} \right.$$

where $\upsilon(LD^1(u,v)) = \min \{\varphi(u)z - vx \mid Fx+g \le z, Cx \le d, x \in \mathcal{X}\} + \min \{vy \mid Ay \le b, y \in \mathcal{Y}\}$

In (P^2) , we dualize the copy constraints with multiplier u and the complicating constraints with multiplier w, to obtain the second Lagrangean dual

$$(LD^2) \qquad \left[\begin{array}{l} \max\limits_{u,\,w \ge 0} \upsilon(LD^2(u,w)) \end{array} \right.$$

where $\upsilon(LD^2(u,w)) = -wb + \min \{\varphi(u)z + wAx \mid Fx+g \le z, Cx \le d, x \in \Omega\}$

Finally, in (P^3) , we dualize the copy constraints and the minimax constraints and obtain the third Lagrangean dual

$$(LD^3) \qquad \left[\max_{u \geq 0 ,v} \upsilon(LD^3(u,v)) \right.$$

where $\upsilon(LD^3(u,v)) = ug + \min \{(1-ue)\delta + (uF - v)x \mid Cx \leq d , x \in \mathcal{X} \} + \min \{ vy \mid Ay \leq b, y \in \mathcal{Y} \}$

We also define the standard Lagrangean dual by relaxing the constraints $Fx+g \leq \delta e$ and $Ax \leq b$

$$(LR) \qquad \left[\max_{u \geq 0 ,w \geq 0} \upsilon(LR(u,w)) \right.$$

where $\upsilon(LR(u,w)) = -wb+ug+ \min \{ (1-ue)\delta+(uF+ wA)x \mid Cx \leq d, x \in \Omega \}$

The next proposition sums up the relationships between all these Lagrangean duals.

Proposition 3 $\qquad \upsilon(LD^1) = \upsilon(LD^3)$ and $\upsilon(LD^2) = \upsilon(LR)$ ◊

Since (LD^1) and (LD^3) are equivalent, we will call (LD) this one true Lagrangean decomposition of (P), and since (LD^2) and (LR) are equivalent, we still simply call (LR) this Lagrangean relaxation.

To discuss the quality of the lower bounds provided by (LD) and (LR), we introduce the LP relaxation of (P) denoted by $(\overline{P})$

$$(\overline{P}) \qquad \left[\min_{x,\delta} \{ \delta \mid Fx+g \leq \delta e, Ax \leq b, Cx \leq d, x \in co(\Omega) \} \right.$$

The main properties may be summarized as follows:

Theorem 4

(i) $\upsilon(Q) \geq \upsilon(\overline{P})$; if (P) has the Integrality Property:

$$\{ x \mid Ax \leq b, Cx \leq d, x \in co(\Omega) \} = co\{ x \mid Ax \leq b, Cx \leq d, x \in \Omega \}$$

then $\upsilon(Q) = \upsilon(\overline{P})$.

(ii) If $\mathcal{Y}$ is convex, then $\upsilon(LD) \leq \upsilon(LR)$.

(iii) If $\mathcal{Y}$ is convex and $co(\Omega) = co(\mathcal{X}) \cap \mathcal{Y}$, then $\upsilon(LD) \geq \upsilon(\overline{P})$.

(iv) $\upsilon(Q) \geq \max \{ \upsilon(LD), \upsilon(LR) \}$.

(v) If $\mathcal{X} = \Omega$ then $\upsilon(LR) \geq \upsilon(\overline{P})$; if (P) has the partial Integrality Property:

$$\{ x \mid Cx \leq d, x \in co(\Omega) \} = co\{ x \mid Cx \leq d, x \in \Omega \}$$

then $\upsilon(LR) = \upsilon(\overline{P})$.

(vi) If $\mathcal{X} = \mathcal{Y} = \Omega$ then $\upsilon(LD) \geq \upsilon(LR)$; if (P) has the partial Integrality Property:

$$\{ x \mid Ax \leq b, x \in co(\Omega) \} = co\{ x \mid Ax \leq b, x \in \Omega \}$$

then $\upsilon(LD) = \upsilon(LR)$. ◊

Theorem 4 is important for recognizing the cases where Lagrangean Decomposition could provide tighter bounds than Lagrangean or LP relaxations. This will happen frequently when $\mathcal{X} = \mathcal{Y} = \Omega$ and (P) has not the partial Integrality Property $\{ x \mid Ax \leq b, x \in co(\Omega) \} = co\{ x \mid Ax \leq b, x \in \Omega \}$.

The following theorem gives sufficient conditions to obtain $\upsilon(LD)$ equal to $\upsilon(P)$ and generalizes proposition 2.

Theorem 5 Let $(u^*,v) \in OS(LD)$ and $(x^*,y^*) \in OS(LD(u^*,v))$. If $x^* = y^*$ then

(i) For any multiplier $v \geq 0$, $\upsilon(LD(u^*,v)) = \upsilon(Q(u^*))$.

(ii) If $u^* \in OS(Q)$ then $\upsilon(LD) = \upsilon(Q)$.

(iii) If u^* is an extreme point of $\mathcal{U}$ and $\max\{(Fx^*+g)_i \mid i=1,...,p\} = (Fx^*+g)_h$ where h denotes the index such that $u^*_h = 1$, then $x^* \in OS(P)$ and $\upsilon(LD) = \upsilon(Q) = \upsilon(P)$. ◊

4. Example

The sufficient conditions given above are useful in practice and easy to check. Consider the following makespan problem $(P\alpha,\beta)$ with 2 machines and 2 jobs (see Escudero [1] for a full description of the problem and also [7,8]):

$$
\begin{array}{lll}
\min & \max\{4x_1+3x_3+4x_5+4x_7,\ 4x_2+2x_4+2x_6+3x_8\} & (Fx+g) \\
\text{s.t.} & & \\
 & x_1+x_2=1 \qquad x_3+x_4=1 & (Ax \leq b) \\
 & x_1 \leq x_5 \qquad x_3 \leq x_7 & \\
 & x_2 \leq x_6 \qquad x_4 \leq x_8 & \\
 & 4x_1+3x_3+4x_5+4x_7 \leq \alpha & (Cx \leq d) \\
 & 4x_2+2x_4+2x_6+3x_8 \leq \beta & \\
 & x_i \in \{0,1\} \quad i=1,...,8 &
\end{array}
$$

The first (resp. second) machine is available α (resp β) units of time, x_5 (resp. x_6) represents the assignment of job type 1 to machine 1 (resp. machine 2) with a potential setup time of 4 (resp. 2) units of time. Similarly, x_7 (resp. x_8) represents the assignment of job type 2 to machine 1 (resp . machine 2) with a potentiel setup time of 4 units (resp. 3). Finally x_1 (resp. x_2) correspond to the assignment of job 1 to machine 1 (resp. machine 2) with processing times of 4 on both machines. Similary, x_3 and x_4 play similar roles for job 2, with processing times of 3 and 2 units respectively.

For problem $(P_{7,6})$ the sufficient conditions of theorem 5 (iii) are satisfied by $u^*= (1,0)$, $v^*= (4,4,3,4,4,4,4,4)$ and $x^*= y^* =(0,1,1,0,0,1,1,0)$. One has $\upsilon(LD(u^*,v^*))=7$, thus $\upsilon(P) = \upsilon(Q) = \upsilon(LD) = 7$ and $x^* \in OS(P)$.

It is important to notice that alone condition $x^*=y^*$ of theorem 5 is not sufficient for optimality as in (Guignard, Kim [4,5,6]). Indeed consider $(P_{15,11})$; it is easy to show that :

. $\mathcal{U}(P)= 7$ and $OS(P) =\{(0,1,1,0,0,1,1,0)\}$

. $\mathcal{U}(Q) = \upsilon(Q(u^*)) = \frac{44}{7}$ with $u^* = (\frac{3}{7}, \frac{4}{7})$ which is not an extreme point of $\mathcal{U}$.

. $\upsilon(LD) = \upsilon(LD(u^*,v^*)) = \upsilon(Q)$ with $v^* = u^*F = \frac{1}{7}(12,16,9,8,12,8,12,12)$; $x^1 = (1,0,0,1,1,0,0,1)$ and $x^2 = (0,1,0,1,0,1,0,1)$ are such that (x^1,x^1) and (x^2,x^2) belong to $OS(LD(u^*,v^*))$, but $\upsilon(LD) < \upsilon(P)$.

5. Concluding remarks

We have presented various relaxations and decompositions wich can be applied to minmax integer programming problems. As an alternative to subgradient optimization, we have also extend column generation to solve Lagrangean decomposition duals ([2]). This technique has been applied to the minimization of excess capacity in loading problems and makespan in flexible manufacturing systems. All the proofs are contained in ([2]).

References

[1] L.F. Escudero (1987) "A mathematical formulation of a hierarchical approach for production planning in FMS", in A. Kusiak, ed. Modern Production Management Systems, North Holland , 231-245.

[2] A Fréville, M. Guignard (1990) "Relaxations for minimax problems", Research report, University of Valenciennes, France.

[3] A. M. Geoffrion (1974) "Lagrangean relaxation for integer programming", Mathematical Programming Study 2, 82-114.

[4] M. Guignard (1984) "Lagrangean decomposition: an improvement over Lagrangean and surrogate duals " , Departement. of Statistics, University of Pennsylvania, USA.

[5] M. Guignard, S.Kim (1987a) "Lagrangean decomposition for integer programming : theory and applications", RAIRO 21,307-324.

[6] M. Guignard, S.Kim (1987b) "Lagrangean decomposition: a model yielding stonger Lagrangean bounds", Mathematical Programming 32, 215-228.

[7] M. Guignard (1989) "Solving makespan minimization problems with Lagrangean decomposition ", Departement of Decision Sciences, University of Pennsylvania, USA

[8] M. Guignard; Hochang Lee (1989) "A hybrid bounding procedure for the workload allocation problem on parallel unrelated machines with setups ",Department of Decision Sciences, University of Pennsylvania, USA

COMBINATORIAL ACCELERATION OF THE BRANCH AND BOUND SEARCH FOR PROCESS NETWORK SYNTHESIS

F. Friedler[1,2] and L. T. Fan[2]

[1]Department of Systems Engineering, Research Institute of Chemical Engineering
Hungarian Academy of Sciences, Veszprém, Pf.: 125, H-8201, Hungary
Phone: (36)80-24-483, Fax: (36)80-28-275, E-mail: h727fri@ella.hu

[2]Department of Chemical Engineering, Kansas State University
Manhattan, Kansas 66506, U.S.A.

INTRODUCTION

The design of any process system for producing desired products from available raw materials almost always involves process network synthesis (PNS). A process system is a network of operating units, each of which transforms a specified number of input materials with known quality into a specified number of output materials by altering their physical, chemical, or biological properties. The importance of PNS arises from the fact that essentially every product of the chemical and allied industries is manufactured by such a network; moreover, the profitability of the same product from different networks may vary widely.

The MINLP model of PNS contains a large number of binary variables associated with the operating units. This renders the model difficult to solve by any available method without exploiting the specific features of process structures and the model. Although its complexity is exponential, the branch and bound method has the advantages of being independent of an initial structure; guaranteeing the optimality provided that the bounding algorithm exists; and being capable of incorporating combinatorial algorithms. Nevertheless, the general branch and bound method is inefficient in solving the MINLP model of PNS because a large number of NLP subproblems is generated and the number of free variables is unnecessarily large for each subproblem, i.e., many of such free variables are associated with operating units that need be excluded from any feasible solution of this subproblem.

Combinatorial analysis of the MINLP model of PNS and that of feasible process structures yield mathematical tools for exploiting the unique characteristics of PNS. These tools can accelerate the branch and bound search for the optimal solution by minimizing the number of subproblems to be solved and by reducing the size of an individual subproblem through exclusion of the binary variables and constraints of those operating units that must not be included in any feasible solution of the subproblem. This algorithm has been validated on the basis of combinatorial analysis of process structures and has been applied for solving industrial instances of PNS.

STRUCTURE REPRESENTATION IN PNS

The simple directed graph is effective in representing structures of general network problems [1]; however, it is unsuitable for PNS as demonstrated by simple examples [2]. Structure representation with enhanced sophistication is required for PNS.

Let M be a given set of objects, usually material species or materials that can be converted or transformed by the process under consideration. Transformation between two subsets of M occurs in an operating unit. It is necessary to link this operating unit to other operating units through the elements of these two subsets of M. The resultant structure can be described by a directed bipartite graph, termed a process graph or P-graph in short, which alleviates the difficulty encountered in representing a process structure by a simple directed graph.

Definition 1. Let M be a finite set, and let set $O \subseteq \wp(M) \times \wp(M)$ with $M \cap O = \varnothing$, where $\wp(M)$ denotes the power set of M. Pair (M, O) is defined to be a *process graph* or *P-graph*; the set of vertices of this graph is $M \cup O$, and the set of arcs is $A = A_1 \cup A_2$ with $A_1 = \{(x, Y) \mid Y = (\alpha, ß) \in O \text{ and } x \in \alpha\}$ and $A_2 = \{(Y, x) \mid Y = (\alpha, ß) \in O \text{ and } x \in ß\}$. P-graph (M', O') is defined to be a *subgraph* of (M, O), i.e., (M', O') $\subseteq$ (M, O), if $M' \subseteq M$ and $O' \subseteq O$. Let (M_1, O_1) and (M_2, O_2) be two subgraphs of (M, O). The *union* of (M_1, O_1) and (M_2, O_2) is defined by P-graph $(M_1 \cup M_2, O_1 \cup O_2)$ denoted by $(M_1, O_1) \cup (M_2, O_2)$; obviously, this union is a subgraph of (M, O). If $(\alpha, ß)$ is an element of O, then, set α is the *input-set* of $(\alpha, ß)$, while set ß is its *output-set*. The set of arcs incident into, out of, and to vertex x are denoted by $\omega^-(x)$, $\omega^+(x)$, and $\omega(x)$, respectively. The indegree, d^-, and the outdegree, d^+, of vertex x are defined by $d^-(x) = |\omega^-(x)|$ and $d^+(x) = |\omega^+(x)|$. The degree of vertex x is defined by $d(x) = d^-(x) + d^+(x)$. Since sets $\omega^-(x)$ and $\omega^+(x)$ do not intersect for a P-graph, we have $d(x) = |\omega(x)|$.

MINLP MODEL OF PNS

Let us consider a PNS problem in which the set of desired products is denoted by P; the set of raw materials, by R; and the set of available operating units, by $O = \{o_1, o_2, \ldots, o_n\}$. Moreover, let M be the set of materials belonging to these units, and assume that $P \cap R = \varnothing$, $P \subseteq M$, $R \subseteq M$, and $M \cap O = \varnothing$. Then, P-graph (M, O) contains the interconnections among units of O. Furthermore, each feasible solution of this problem corresponds to a subgraph of (M, O). For any $1 \le j \le n$, let $y_j = 1$ if o_j is contained in this subgraph and $y_j = 0$ otherwise. Thus, this subgraph is determined by the vector $(y_1, y_2, \ldots, y_n)$. Let $A = \{a_1, a_2, \ldots, a_r\}$ be the set of arcs and continuous variable x_k ($k = 1, 2, \ldots, r$) be assigned to arc a_k. The function, for which $\phi(\{a_{i_1}, a_{i_2}, \ldots, a_{i_t}\}) = (x_{i_1}, x_{i_2}, \ldots, x_{i_t})$ holds for any subset $\{a_{i_1}, a_{i_2}, \ldots, a_{i_t}\}$ of A, is denoted by ϕ. Finally, continuous variable z_j is assigned to operating unit o_j for $j = 1, 2, \ldots, n$.

The constraints on and the cost of operating unit o_j can be expressed, respectively, by

$$g_j(y_j, \phi(\omega(o_j)), z_j) \le 0, \qquad j = 1, 2, \ldots, n$$

$$f_j(y_j, \phi(\omega(o_j)), z_j), \qquad j = 1, 2, \ldots, n$$

where for a fixed value of y_j, both f_j and g_j are nonlinear, differentiable functions on the practically interesting domain for $j = 1, 2, \ldots, n$.

Similarly, the constraint on and the cost function of vertex m_i can be given, respectively, as follows:

$$g_i'(\phi(\omega(m_i))) \leq 0, \qquad i = 1, 2, \ldots, l$$

and

$$f_i'(\phi(\omega(m_i))), \qquad i = 1, 2, \ldots, l$$

In practice, g' and f' are usually linear. The cost function of the PNS problem is the sum of the costs of the materials and operating units involved.

COMBINATORIAL STRUCTURE OF PNS

In general, no arbitrary vector $(y_1, y_2, \ldots, y_n)$ $(y_i \in \{0, 1\}, i = 1, 2, \ldots, n)$ can define a feasible process structure. The feasible process structures have some common combinatorial properties [2] that have been expressed implicitly in the MINLP model. Since each feasible process structure must have these combinatorial properties, the set of subgraphs of (M, O), considered in solving the model, can be reduced to the set of combinatorially feasible process structures or *solution-structures* in short.

Definition 2. Subgraph (M', O') of P-graph (M, O) is defined to be a *solution-structure* of PNS given by set P of products and set R of raw materials if

(S1) $P \subseteq M'$, i.e., every final product is represented in P-graph (M', O');

(S2) $\forall x \in M'$, $d^-(x) = 0$ iff $x \in R$, i.e., a vertex from M' has no input if and only if it represents a raw material;

(S3) $\forall u \in O'$, $\exists$ path [u, v] in (M', O'), where $v \in P$, i.e., every vertex from O' has at least one path leading to a vertex representing a final product; and

(S4) $\forall x \in M'$, $\exists (\alpha, ß) \in O'$ such that $x \in (\alpha \cup ß)$, i.e., any vertex from M' must be an input to or output from at least one vertex from O'.

The set of solution-structures is denoted by S(P, R, O); its important properties are expressed by the following theorem, lemma, and corollaries.

Theorem 1. S(P, R, O) is closed under union.

Lemma. If $(M', O') \in S(P, R, O)$, then, $M' = \bigcup_{(\alpha,ß)\in O'} (\alpha \cup ß)$.

The direct consequence of this lemma is the following corollary.

Corollary 1. Let $(M', O') \in S(P, R, O)$; then, (M', O') is uniquely determined if set O' is given.

The maximal structure, defined below, plays an essential role in PNS.

Definition 3. Let us assume that $S(P, R, O) \neq \varnothing$. The union of all solution-structures of PNS is defined to be its *maximal structure*; it will be denoted by $\mu(P, R, O)$, i.e.,

$$\mu(P, R, O) = \bigcup_{\sigma\in S(P,R,O)} \sigma.$$

Since the set of solution-structures is finite and closed under union, the maximal structure also is a solution-structure; this leads to the following corollary.

Corollary 2. $\mu(P, R, O) \in S(P, R, O)$.

Naturally, the optimal solution need not be concerned with any operating unit not included in the maximal structure. Since any optimal solution is a solution-structure, the MINLP model of PNS can be based on the maximal structure. For this reason, let us suppose that $S(P, R, O) \neq \varnothing$, and also let us denote the maximal structure, $\mu(P, R, O)$, by (M', O').

A polynomial algorithm is available for the generation of the maximal structure [3].

BUILDING BLOCKS OF THE ACCELERATED BRANCH AND BOUND METHOD

Essentially, the branch and bound method yields the optimal solution of a mathematical programming problem by generating and solving some simplified subproblems. Suppose that we have three sets I_0, I_1, and I_f (any pair of them is disjunct) and that $I_0 \cup I_1 \cup I_f = \{1, 2, \ldots, n\}$. These sets define one subproblem of the branch and bound method. In this subproblem, I_0 and I_1 are the sets of indices of binary variables whose values are zero and one, respectively, and I_f is the set of indices for the free variables of this subproblem, i.e., the value of any of these variables is supposed to be in closed interval [0,1].

Subproblem Generation

The structures of some, or often most, subproblems, defined by I_0, I_1, and I_f, are not substructures of any solution-structure; these subproblems are said to be structurally infeasible, as will be delineated later. Only structurally feasible subproblems should be generated.

Definition 4. Let $\mu(P, R, O) = (M', O')$. Then, P-graph (m^*, o^*) is a *subsolution-structure* of PNS given by set P of products and set R of raw materials, if

(SS1) for $x \in m^*$, $d^-(x) = 0$, if $x \in R$;

(SS2) $o^* \subseteq O'$;

(SS3) $\forall u \in O'$, $\exists$ path [u, v] in (M', O'), where $v \in P$;

(SS4) $\forall x \in M'$, $\exists (\alpha, ß) \in O'$ such that $x \in (\alpha \cup ß)$.

Let $S^*(P, R, O)$ denote the set of subsolution-structures; note that $(\varnothing, \varnothing) \in S^*(P, R, O)$. If $(m^*, o^*) \in S^*(P, R, O)$, then, $(m^*, o^*) \subseteq \mu(P, R, O)$.

Theorem 2. $S(P, R, O) \subseteq S^*(P, R, O)$.

For a given subsolution-structure, $\sigma^* = (m^*, o^*)$ $(\in S^*(P, R, O))$, let us define set r^* such that

$$r^* = \{x \mid x \in (m^* \setminus R \cup P) \text{ and } d^-(x) = 0\}.$$

Theorem 3. Let $\sigma^* \in S^*(P, R, O)$; then, $\sigma^* \in S(P, R, O)$, if and only if $r^* = \varnothing$.

The accelerated branch and bound algorithm is based on algorithm SSG, given in Figure 1. Algorithm SSG generates each solution-structures exactly once and generates solution-structures only. It does so by determining the decision-mappings of some subsolution-structures (see the APPENDIX). These subsolution-structures define structurally feasible subproblems of the MINLP model of PNS. The validity of algorithm SSG has been proved by resorting to the following theorems.

Theorem 4. Decision-mapping $\delta[m]$ of algorithm SSG is consistent, and it is a subsolution-structure.

Theorem 5. Algorithm SSG generates all solution-structures whose decision-mappings are the extensions of $\delta[m]$.

Theorem 6. $\varnothing$ is a decision-mapping of algorithm SSG.

Theorem 7. No decision-mapping is generated more than once by algorithm SSG.

Theorem 8. Decision-mapping $\delta[m]$ of algorithm SSG is a solution-structure if and only if set p' of algorithm SSG is empty.

```
Input: P, R, M, o(x) (x∈M) ;
Comment: P⊆M, R⊆M, O⊆℘(M)×℘(M), P∩R=∅,
         o(x)={(α,ß) | (α,ß)∈O & x∈ß}, o(x)=∅ ⇔ x∈R ;
Output: all solution-structures of the PNS problem ;
Global variables: R, o(x) (x∈M) ;

begin
if P = ∅ then stop ;
SSG( P, ∅, ∅ ) ;
end

procedure SSG( p, m, δ[m] ) :
begin
if p = ∅ then begin write δ[m] ; return end
let x ∈ p ;
C := ℘ (o(x)) \ {∅} ;
For all c ∈ C do
                begin
                if ∀ y∈m, c∩(o(y) \ δ(y))=∅ & (o(x) \ c)∩δ(y)=∅
                   then
                        begin
                        δ[m∪{x}] := δ[m]∪{(x,c)} ;
                        SSG( p∪(  ∪  α ) \ (R∪m∪{x}), m∪{x}, δ[m∪{x}] ) ;
                             (α,ß)∈c
                        end
                end
return
end
```

Figure 1. Algorithm SSG

Theorem 9. Only one decision-mapping of algorithm SSG may belong to a solution-structure.

Subproblem Given by a Subsolution-Structure

Let us define a mapping, denoted by ind, that yields the set of indexes for the elements of a subset of O. Moreover, let $\delta[m]$ be a decision-mapping of subsolution-structure σ^*, and also let S' be the following set;

$S' = \{\sigma \mid \sigma \in S(P, R, O)$ and the decision-mapping of σ is an extension of $\delta[m]\}$.

If this set is not empty, then, $\delta[m]$ and the subproblem determined by $\delta[m]$ are defined to be structurally feasible.

Theorem 10. Suppose that $\delta[m]$ is a structurally feasible subsolution-structure and that structure σ is defined by $\sigma = \bigcup_{\sigma' \in S'} \sigma'$. Then, all solution-structures whose decision-mapping are the extensions of $\delta[m]$ are a substructure of σ. Conversely, σ is minimal with this property, i.e., for any structure ρ such that $\sigma \mathcal{C} \rho$, there exists solution-structure σ' such that $\sigma' \mathcal{C} \rho$.

Obviously, decision-mapping $\delta[m]$ can be extended to decision-mapping $\delta[M^*]$, a decision-mapping of S'. Then, sets I_1 and I_f of a subproblem defined by $\delta[m]$ are

$I_1 = \text{ind}(\bigcup_{x \in m} \delta(x))$ and

$I_f = \text{ind}(\bigcup_{x \in M} \delta'(x))$,

where $\delta'[M] = \delta[M^*] \setminus \delta[m]$.

ACCELERATED BRANCH AND BOUND ALGORITHM

Based on the building blocks mentioned above, an accelerated branch and bound algorithm, algorithm ABB has been developed for solving the MINLP model of a PNS problem (see Figure 2). This algorithm yields the optimal solution provided that the bounding

```
Input: P, R, M, o(x) (x∈M) ;
Comment: P⊆M, R⊆M, O⊆℘(M)×℘(M), P∩R=∅,
         o(x)={(α,ß) | (α,ß)∈O & x∈ß}, o(x)=∅ ⇔ x∈R ;
Output: optimal solution of the PNS problem ;
Global variables: R, o(x) (x∈M), U, currentbest ;

begin
U := ∞ ; currentbest := anything ;
if P = ∅ then stop ;
ABB( P, ∅, ∅ ) ; if U < ∞ then print currentbest else print 'there is no solution' ;
end

procedure ABB( p, m, δ[m] ) :
begin
let x ∈ p ; C := ℘( o(x)) \ {∅} ;
For all c ∈ C do
        begin
        if ∀ y∈m, c∩δ̄(y)=∅ & (o(x) \ c)∩δ(y)=∅
        then begin
             δ[m∪{x}] := δ[m]∪{(x, c)} ;
             p' := p∪(    ∪    α ) \ (R∪m∪{x}) ; m' := m ∪ {x} ;
                    (α,ß)∈c
             if p' = ∅
             then begin
                  U := min( U, BOUND( m', ∅, δ[m'] ) ) ; update currentbest ;
                  end
             else if RSG( p', m', δ[m'], M, δ[m'∪M] )
                  then if U ≥ BOUND( m', M, δ[m'∪M] ) then ABB( p',m', δ[m'] ) ;
             end
         end
return
end
```

Figure 2. Algorithm ABB

algorithm exists, by generating structurally feasible subproblems only. Moreover, the size of each subproblem is reduced by excluding the binary variables and constraints of those operating units that can not be included in any feasible solution of the subproblem (see Figure 3 for procedure RSG).

```
procedure RSG( p', m', δ[m'], M, δ[m'∪M] ) :
p := p' ;
M := ∅ ;
while p is not empty do
      begin
       x ∈ p ;
       M := M∪{x} ;
       δ(x) := o(x) \ ( ∪ δ̄(y)) ;
                     y∈m'
       if δ(x) = ∅ then return false ;
        p := p∪(      ∪      α ) \ (R∪m'∪M) ;
              (α,ß)∈δ(x)
        end
return true
end
```

Figure 3. Procedure RSG

Example

The accelerated branch and bound algorithm has generated 6325 subproblems for an industrial PNS problem involving 35 operating units in the worst case [2]. This is about one millionth of the number of the subproblems generated by the general branch and bound algorithm in the worst case. The reduction in the number of free variables of each subproblem of the accelerated branch and bound algorithm is also essential.

REFERENCES

[1] Minoux, M., Network Synthesis and Optimum Network Design Problems: Models, Solution Methods and Applications, Networks, **19**, 313-360 (1989).

[2] Friedler, F., K. Tarján, Y.W. Huang, and L.T. Fan, Graph-Theoretic Approach to Process Synthesis: Axioms and Theorems, Chem. Eng. Sci., **47**(8), 1973-1988 (1992).

[3] Friedler, F., K. Tarján, Y.W. Huang, and L.T. Fan, Combinatorial Algorithms for Process Synthesis, Computers chem. Engng, **16**, Suppl., S313-20 (1992).

APPENDIX

Decision Mappings

To generate a certain class of subgraphs of a graph, e.g., a set of feasible structures, a special technique, decision-mapping has been developed to organize the system of decisions.

Decision-mapping is a special tool to render our decisions consistent and complete in dealing with complex decision-problems such as PNS.

Let us suppose that for finite sets M and O, $O \subseteq \wp(M) \times \wp(M)$ holds; moreover, for $x \in M$, let us define set o(x) by $o(x) = \{(\alpha, ß) \mid (\alpha, ß) \in O \text{ and } x \in ß\}$; naturally, pair (M, O) is a P-graph.

Definition 5. Let us suppose that sets M and O satisfy $O \subseteq \wp(M) \times \wp(M)$ and set m be a subset of M. Moreover, let $\delta(x)$ be a subset of o(x), for $x \in m$. Then, $\delta[m] = \{(x, \delta(x)) \mid x \in m\}$ is defined to be a *decision-mapping* on its domain m.

Definition 6. Decision-mapping $\delta_1[m_1]$ is defined to be the *restriction* of decision-mapping $\delta_2[m_2]$ to m_1, if $m_1 \subseteq m_2$ and $\delta_1[m_1] = \{(x, \delta_2(x)) \mid x \in m_1\}$.

Definition 7. The *complement* of decision-mapping $\delta[m]$ is defined by $\delta^*[m] = \{(x, y) \mid x \in m \text{ and } y = o(x) \setminus \delta(x)\}$. Thus, $\delta^*(x) = o(x) \setminus \delta(x)$ for $x \in m$.

Definition 8. Decision-mapping $\delta[m]$ is *consistent* if $|m| \leq 1$ or $(\delta(x) \cap \delta(y)) \cup (\delta^*(x) \cap \delta^*(y)) = o(x) \cap o(y)$ for any $x, y \in m$.

Theorem 11. Decision-mapping $\delta[m]$ with $|m| \geq 1$ is consistent if and only if $\delta(x) \cap \delta^*(y) = \varnothing$ for all $x, y \in m$.

Theorem 12. Decision-mapping $\delta_1[m_1]$ is consistent if $\delta_1[m_1] \subseteq \delta_2[m_2]$ and $\delta_2[m_2]$ is a consistent decision-mapping.

Definition 9. For consistent decision-mapping $\delta[m]$, let $\mathbf{o} = \bigcup_{x \in m} \delta(x)$, $\mathbf{m} = \bigcup_{(\alpha,ß) \in \mathbf{o}} (\alpha \cup ß) \cup m$, and $\delta'[m] = \{(x, y) \mid x \in m \text{ and } y = \{(\alpha, ß) \mid (\alpha, ß) \in \mathbf{o} \text{ and } x \in ß\}\}$; then, $\delta'[m]$ is defined to be the *closure* of $\delta[m]$, and $\delta[m]$ is said to be *closed* if $\delta[m] = \delta'[m]$. The closure of a consistent decision-mapping is closed.

Theorem 13. Let $\delta'[m]$ be the closure of consistent decision-mapping $\delta[m]$; then, $\delta(x) = \delta'(x)$ for all $x \in m$, i.e. $\delta[m]$ is the restriction of $\delta'[m]$ to m.

Corollary 3. If $\delta'[m]$ is the closure of consistent decision-mapping $\delta[m]$, then $\delta[m] \subseteq \delta'[m]$.

Theorem 14. The closure of a consistent decision-mapping is consistent.

Definition 10. Two consistent decision-mappings are *equivalent* if they have common closure.

Naturally, a consistent decision-mapping is equivalent to its closure, and the relation "equivalent" is an equivalence relation.

Definition 11. m' is said to be an *active domain* of decision mapping $\delta[m]$ if $m' \subseteq m$, $\bigcup_{x \in m'} \delta(x) = \bigcup_{x \in m} \delta(x)$ and $\bigcup_{x \in m'} \delta^*(x) = \bigcup_{x \in m} \delta^*(x)$.

Note that m is always an active domain of decision-mapping $\delta[m]$, and a decision-mapping can have multiple active domains.

Theorem 15. Let $\delta[m]$ be a consistent decision-mapping; then, it is determined on its whole domain, m, if it is given only on one of its active domains.

Theorem 16. If a decision-mapping is consistent on one of its active domains, then, it is consistent.

Definition 12. Let $\delta_1[m_1]$ and $\delta_2[m_2]$ be consistent decision-mappings with their closures, $\delta_1'[m_1]$ and $\delta_2'[m_2]$, respectively. Then, $\delta_1[m_1]$ is defined to be an *extension* of

$\delta_2[m_2]$ if

(i) $m_1 \supseteq m_2$, $m_1 \supseteq m_2$,

(ii) $\delta_2[m_2]$ is the restriction of $\delta_1[m_1]$ to m_2, i.e., $\delta_1(x) = \delta_2(x)$ for $x \in m_2$, and

(iii) $\delta_1'(x) \supseteq \delta_2'(x)$ for $x \in m_2 \setminus m_2$.

A consistent decision-mappings is an extension of $\varnothing$. That $\delta_1[m_1]$ is an extension of $\delta_2[m_2]$ is denoted by $\delta_1[m_1] > \delta_2[m_2]$.

Theorem 17. $\delta'[m] > \delta[m]$ where $\delta'[m]$ is the closure of consistent decision-mapping $\delta[m]$.

Theorem 18. The relation extension is a partial order of the set of consistent decision-mappings.

Let P-graph (m, o) be a subgraph of P-graph (M, O)

Definition 13. m' is an *active set* of P-graph (m, o), if $m' \subseteq m$ and $ß \cap m' \neq \varnothing$ for any $(\alpha, ß) \in o$.

Definition 14. Let m' be an active set of P-graph (m, o); then, $\delta[m']$ is defined to be a *decision-mapping of P-graph* (m, o) if $\delta[m'] = \{(x, y) \mid x \in m' \text{ and } y = \{(\alpha, ß) \mid (\alpha, ß) \in o \text{ and } x \in ß\}\}$, i.e., $\delta(x) = \{(\alpha, ß) \mid (\alpha, ß) \in o \text{ and } x \in ß\}$ for $x \in m'$.

Theorem 19. The decision-mappings of a P-graph are consistent.

Theorem 20. Decision-mapping $\delta[m]$ of P-graph (m, o) is closed if set m is active.

Theorem 21. An active set of P-graph (m, o) is an active domain of its decision-mapping $\delta[m]$ if set m is active.

Theorem 22. The decision-mappings of a P-graph are equivalent provided that this P-graph has an active set.

Theorem 23. Let $\delta[m']$ be a consistent decision-mapping, $o = \bigcup_{x \in m'} \delta(x)$ and $m = \bigcup_{(\alpha,ß) \in o} (\alpha \cup ß)$. Then, (m, o) is a P-graph, m' is an active set of P-graph (m, o), and $\delta[m']$ is a decision-mapping of P-graph (m, o).

Definition 15. The *P-graph of consistent decision-mapping* $\delta[m']$ is defined to be (m, o), where $o = \bigcup_{x \in m} \delta(x)$ and $m = \bigcup_{(\alpha,ß) \in o} (\alpha \cup ß)$.

Theorem 24. An active domain of a consistent decision-mapping is an active set of its P-graph.

Theorem 25. Equivalent consistent decision-mappings have the same P-graph.

On a special class of linear programs with an additional reverse convex constraint

János Fülöp
Laboratory of Operations Research and Decision Systems,
Computer and Automation Institute, Hungarian Academy of Sciences,
H-1518 P.O.Box 63, XI. Kende u. 11-13. Budapest, Hungary.

Abstract: In the paper, a special class of linear programs with an additional reverse convex constraint is treated. The problems to be considered have the special property that the feasible set is the union of some faces of the polyhedron determined by the linear constraints. Several nonconvex programming problems can be written into this form, e.g. the minimum linear complementarity problem, the linear disjunctive programming problem, the linear bilevel programming problem, the problem of linear optimization over the efficient set, etc. We propose a finite method based on convexity and disjunctive cuts for solving such problems.

1. Introduction

The problems to be considered are given in the form

$$\min \; c^T x \;\text{ s.t. }\; x \in P,\; g(x) = 0, \tag{1.1}$$

where $P \subset R^n$ is a nonempty polyhedron, c is an n-vector and ^{T} denotes the transposition. In addition, $g : G \to R$ is a concave function such that $G \subseteq R^n$ is a convex set, $P \subseteq G$ and

$$g(x) \geq 0 \quad \text{for every} \quad x \in P.$$

Because of the last property, (1.1) is equivalent to

$$\min \; c^T x \;\text{ s.t. }\; x \in P,\; g(x) \leq 0, \tag{1.2}$$

which is the form of a linear program with an additional reverse convex constraint. Several methods have been published for solving linear programs with an additional reverse convex constraint, see e.g. [6,8] and the references therein. However, instead of applying one of these methods directly, we propose a modification of the algorithm presented in [6] for solving (1.2). This is motivated by the property that the possibly nonempty feasible set of (1.1) is the union of some faces of P. Consequently, if (1.1) has finite optimal value, there exists a vertex of P among the optimal solutions. For a linear program with a general reverse

convex constraint, we can state only that an optimal solution can be attained on an edge of P.

In Section 2, we propose a finite cutting plane method for solving (1.1). In Sections 3 through 6, we show that a wide range of nonconvex programming problems is transformable into the form of (1.1).

2. A finite cutting plane method

We assume that the polyhedron P is given by $P = \{x \in R^n \mid Ax = b,\ x \geq 0\}$, where A is an $m \times n$ matrix and b is an m-vector. Let X denote the feasible set of (1.1).

Proposition 2.1. *If $X \neq \emptyset$, then X is the union of some faces of P.*

Proposition 2.2. *If $X \neq \emptyset$, then exactly one of the following cases holds:*

(i) *Problem (1.1) has a finite optimal value and there exists an extreme point of P among the optimal solutions;*

(ii) *The objective function $c^T x$ is unbounded below over X and there exists an edge F of P such that $F \subseteq X$ and $c^T x$ is unbounded below over F.*

Consider first the case when $c^T x$ is bounded below over P, e.g. P is bounded. Let $V(P)$ denote the set of the extreme points of P. Consider the problem

$$\min\ c^T x \ \text{ s.t. } \ x \in V(P),\ g(x) = 0. \tag{2.1}$$

By Proposition 2.2, problems (1.1) and (2.1) have simultaneously feasible solution, and any optimal solution of (2.1) is optimal for (1.1) as well. If we are interested only in finding the optimal value and an optimal solution of (1.1), it is enough to solve (2.1). We shall deal with (2.1) in the sequel.

Any $x^0 \in V(P)$ is also a basic feasible solution of

$$Ax = b,\ x \geq 0. \tag{2.2}$$

For a feasible basis B of (2.2), let the simplex tabular form of (2.2) be determined by

$$x_i + \sum_{j \in I_B} \alpha_{ij} x_j = \alpha_{i0},\ \ i \in I_B, \tag{2.3}$$

where I_B and $\bar{I}_B$ denote the index sets of the basic and nonbasic variables, respectively.

Consider an $x^0 \in V(P)$ such that $g(x^0) > 0$. Then, x^0 is not a feasible solution of (1.1). We construct a convexity cut to exclude x^0 from the further search. Let

B be a feasible basis belonging to x^0. Using the coefficients of the simplex tabular form (2.3) determined by B, construct the vectors $z^j \in R^n$, $j \in \bar{I}_B$, defined by

$$z_k^j = \begin{cases} 1 & \text{for } k = j, \\ -\alpha_{kj} & \text{for } k \in I_B, \\ 0 & \text{otherwise,} \end{cases} \qquad k = 1, \ldots, n;\ j \in \bar{I}_B. \tag{2.4}$$

For every $j \in \bar{I}_B$, compute

$$\lambda_j = \sup \{\lambda \mid \bar{x} + \lambda z^j \in G,\ g(x + \lambda z^j) > 0\}. \tag{2.5}$$

Clearly, $\lambda_j \geq 0$ for every $j \in \bar{I}_B$.

Proposition 2.3. *Assume that $\lambda_j > 0$ for every $j \in \bar{I}_B$. Define $t \in R^n$ by*

$$t_j = \begin{cases} 1/\lambda_j & \text{for } j \in \bar{I}_B \text{ and } \lambda_j < \infty, \\ 0 & \text{otherwise,} \end{cases} \qquad k = 1, \ldots, n. \tag{2.6}$$

Then

$$t^T x^0 < 1 \quad \text{and} \quad g(x) > 0 \text{ for every } x \in P \cap \{x \in R^n \mid t^T x < 1\}. \tag{2.7}$$

Proposition 2.4. *If the vertex x^0 is nondegenerate, then $\lambda_j > 0$ for every $j \in \bar{I}_B$.*

Proposition 2.5. *If x^0 is an inner point of G, then $\lambda_j > 0$ for every $j \in \bar{I}_B$.*

Assume that we have an $x^0 \in V(P)$ such that $g(x^0) > 0$ and $\lambda_j > 0$ for every $j \in \bar{I}_B$. Then, the convexity cut

$$t^T x \geq 1 \tag{2.8}$$

defined by (2.6) cuts off x^0 but leaves the possible points of X. If we have $\lambda_j = 0$ for a $j \in \bar{I}_B$, then a convexity cut similar to (2.8) can be also generated at the expense of some extra efforts including the determination of the edges emanating from x^0 and solving a linear program [8].

In the latter case, an alternative and faster way of excluding x^0 from the further search is the generation of a disjunctive cut. We construct a cut of form (2.8) such that

$$t^T x^0 < 1 \text{ and } t^T x \geq 1 \text{ for every } x \in V(P) \setminus \{x^0\}. \tag{2.9}$$

Let $I_+ = \{i \mid x_i^0 > 0\}$. Then for any $x \in V(P) \setminus \{x^0\}$, there exists at least one $i \in I_+$ such that $x_i = 0$. The disjunctive cut is constructed based upon this disjunction.

Proposition 2.6. *Consider a simplex tabular form (2.3) belonging to x^0 and let*

$$t_j = \begin{cases} \max\{\alpha_{ij}/\alpha_{i0} \mid i \in I_+\} & \text{for } j \in \bar{I}_B, \\ 0 & \text{otherwise,} \end{cases} \qquad k = 1, \ldots, n. \tag{2.10}$$

Then the cut (2.8) defined by (2.10) fulfils (2.9).

Assume now that we have found an $x^0 \in V(P)$ such that $g(x^0) = 0$. Then x^0 is feasible to (1.1) and (2.1). Let $N(x^0)$ denote the set of those vertices of P which are adjacent to x^0. Examine whether there exists an $x^1 \in V(P)$ such that

$$x^1 \in N(x^0), \ g(x^1) = 0 \text{ and } \ c^T x^1 < c^T x^0. \tag{2.11}$$

If we find such an x^1, then we replace x^0 by x^1 and repeat the matter above. In this way, we step on feasible solutions of (2.1) meanwhile improving the objective function value.

After a finite number of improving steps, we obtain an x^0 feasible to (2.1) such that we cannot find an $x^1 \in V(P)$ fulfilling (2.11). It may also occur that there exists such an x^1 but x^0 is a degenerate vertex and we would like to spare the time needed for determining $N(x^0)$. We add the objective function cut

$$c^T x \leq \gamma, \tag{2.12}$$

where $\gamma = c^T x^0$, in order to exclude the points with objective function value greater than γ. However, since x^0 fulfils (2.12), we also generate a disjunctive cut presented above to exclude x^0 from the further search.

After adding one or two of the cuts presented above, we proceed with a new $x^0 \in V(P)$, if any, such that x^0 fulfils the cut constraints generated earlier. At a step of the algorithm, let $Q \subset R^n$ be the set of the points feasible to the cuts. Of course, Q is a polyhedron. The subproblem to be solved is now to find a point of $Q \cap V(P)$ or to prove $Q \cap V(P) = \emptyset$. This is a well-known problem of nonconvex programming. It was treated first by Majthay and Whinston [10] in a concave minimization context. They proposed a finite cutting plane method using a parametric programming technique. Their method was improved and extended by Fülöp [5] using a disjunctive programming technique. We suggest applying the finite method of [5] to find a point of $Q \cap V(P)$, if any.

The cutting plane method proposed for solving (2.1) is summarized below.

Algorithm 2.1:

Step 0: Set $\gamma \leftarrow \infty$ and $Q \leftarrow R^n$.

Step 1: If $Q \cap V(P) = \emptyset$, stop. Otherwise, find an $x^0 \in Q \cap V(P)$. If $g(x^0) = 0$, go to Step 2. Otherwise, go to Step 3.

Step 2: If we can find an x^1 fulfilling (2.11), then set $x^0 \leftarrow x^1$ and repeat Step 2. Otherwise, set $\gamma \leftarrow c^T x^0$, $x^* \leftarrow x^0$, generate the disjunctive cut (2.8) defined by (2.10) and set $Q \leftarrow Q \cap \{x \in R^n \mid t^T x \geq 1, c^T x \leq \gamma\}$. Go to Step 1.

Step 3: For every $j \in \bar{I}_B$ determine λ_j by (2.5). If $\lambda_j > 0$ for every $j \in \bar{I}_B$, generate the convexity cut (2.8) defined by (2.6). Otherwise, generate the disjunctive cut (2.8) defined by (2.10). Set $Q \leftarrow Q \cap \{x \in R^n \mid t^T x \geq 1\}$ and go to Step 1.

Proposition 2.7. *Algorithm 2.1 solves (2.1) in finite steps. If $\gamma = \infty$, then (2.1) has not feasible solution. Otherwise, γ is the optimal value and x^* is an optimal solution of (2.1).*

We turn now to the case when $c^T x$ is unbounded below over P. We have to check whether $c^T x$ is unbounded below over X as well. If (2.1) has not feasible solution, we are done since $X = \emptyset$. Otherwise, let γ be the optimal value of (2.1) and choose a $\bar{\gamma} > \gamma$ arbitrarily. Let $\bar{P} = P \cap \{x \in R^n \mid c^T x = \bar{\gamma}\}$.

Proposition 2.8. *The objective function $c^T x$ is unbounded below over X if and only if*

$$\min \{g(x) \mid x \in \bar{P}\} = 0. \tag{2.13}$$

It is clear that (2.13) holds if and only if there exists a vertex x of $\bar{P}$ such that $g(x) = 0$. Similarly to Algorithm 2.1, a finite algorithm based on convexity and disjunctive cuts can be proposed to verify (2.13).

3. The minimum linear complementarity problem

Consider the minimum linear complementarity problem given in the form

$$\min \ c^T x \ \text{ s.t. } \ Ax = b, \ x \geq 0, \ x_i x_{\bar{n}+i} = 0 \ \text{ for } \ i = 1, \ldots, \bar{n}, \tag{3.1}$$

where the sizes of matrix A and vectors x, b and c are the same as in the previous sections. We assume that $2\bar{n} \leq n$. Judice and Mitra [9] showed that several well-known mathematical programming problems can be transformed into (3.1). See [9] for a list of such problems and the details of the reformulations.

For an $x \in R^n$, let

$$g(x) = \sum_{i=1}^{\bar{n}} \min\{x_i, x_{\bar{n}+i}\}.$$

Then $g : R^n \to R$ is a concave function. Let $P = \{x \in R^n \mid Ax = b,\ x \geq 0\}$. It is easy to see that (3.1) is equivalent to (1.1).

We mention that the reformulation of some mathematical programming problems into (3.1) may result unrestricted variables x_j, $j = 2\bar{n}+1, \ldots, n$. The method presented in Section 2 can be easily modified for such problems. Another way is to write these unrestricted variables as differences of nonnegative variables.

4. The linear disjunctive programming problem

A mathematical programming problem is called linear disjunctive programming problem if the feasible set can be represented by a finite number of intersection and union operations on a finite number of closed halfspaces. In addition, a linear function is to be optimized over the feasible set. It can be shown [7] that any linear disjunctive program can be transcribed into an equivalent problem of following form:

$$\min\ c^T x \ \text{ s.t. } \ Ax = b,\ x \geq 0,\ \prod_{j \in I_i} x_j = 0 \ \text{ for } \ i = 1, \ldots, k, \tag{4.1}$$

where $I_i \subseteq \{1, \ldots n\}$ for $i = 1, \ldots, k$. Let the concave function $g : R^n \to R$ be defined by

$$g(x) = \sum_{i=1}^{k} \min\{x_j \mid j \in I_i\}.$$

Then (4.1) is equivalent to (1.1). It is also easy to see that (3.1) is a special case of (4.1) with $k = \bar{n}$ and $I_i = \{i, \bar{n} + i\}$ for $i = 1, \ldots, k$.

5. The linear bilevel programming problem

Consider the linear bilevel programming problem [1,4] stated as follows:

$$\max_{y}\ c^{(1)^T} y + c^{(2)^T} z, \ \text{ where } \ z \text{ solves} \tag{5.1}$$

$$\max_{z}\ d^{(1)^T} y + d^{(2)^T} z \tag{5.2}$$

$$\text{s.t. } \ A^{(1)} y + A^{(2)} z = b, \tag{5.3}$$

$$y \geq 0, \ \ z \geq 0, \tag{5.4}$$

where $c^{(1)}$, $d^{(1)}$ and y are n_1-vectors, $c^{(2)}$, $d^{(2)}$ and z are n_2-vectors, b is an m-vector, $A^{(1)}$ is an $m \times n_1$ matrix and $A^{(2)}$ is an $m \times n_2$ matrix. Let

$$x = \begin{bmatrix} y \\ z \end{bmatrix}, \ c = \begin{bmatrix} c^{(1)} \\ c^{(2)} \end{bmatrix}, \ d = \begin{bmatrix} d^{(1)} \\ d^{(2)} \end{bmatrix} \ \text{ and } \ A = \left[A^{(1)}, A^{(2)}\right].$$

Let $P = \{x \in R^{n_1+n_2} \mid Ax = b, x \geq 0\}$. We assume that $P \neq \emptyset$ and $d^T x$ is bounded above over P. For an $x \in R^{n_1+n_2}$, let

$$g(x) = -d^{(2)^T} z + \max\{d^{(2)^T} \bar{z} \mid A^{(2)} \bar{z} = b - A^{(1)} y, \bar{z} \geq 0\}, \tag{5.5}$$

where $g(x) = -\infty$ if the maximization problem in (5.5) has not feasible solution.

Let $G = \{x \in R^{n_1+n_2} \mid g(x) > -\infty\}$. It can be shown that G is convex, $P \subseteq G$, g is a piecewise linear, continuous and concave function over G and $g(x) \geq 0$ for every $x \in P$. In addition, (5.1)-(5.4) is equivalent to (1.1) with the modification that $c^T x$ is to be maximized now.

6. Linear optimization over the efficient set

Consider the multiple objective linear program

$$\text{`max'} \ Cx \ \text{s.t.} \ x \in P, \tag{6.1}$$

where C is a $k \times n$ matrix and $P \subset R^n$ is nonempty polyhedron. By definition [11], a point $x^0 \in P$ is an efficient solution of (6.1) if and only if there exists no $x \in P$ such that $Cx \geq Cx^0$ and $Cx \neq Cx^0$. Let $E(P)$ denote the set of the efficient solutions. Consider the problem

$$\min \ c^T x \ \text{s.t.} \ x \in E(P), \tag{6.2}$$

where c is an n-vector. Problem (6.2) has several applications in multiple objective programming, see [2,3,11] and the references therein.

For an $x \in R^n$, let $g(x)$ be defined by

$$g(x) = \max \ e^T C(y - x) \ \text{s.t.} \ Cy \geq Cx, \ y \in P. \tag{6.3}$$

where $g(x) = -\infty$ if (6.3) has not feasible solution and $g(x) = \infty$ if the objective function is unbounded above over the nonempty feasible set of (6.3). In (6.3), e is the k-vector whose every component is equal to 1.

Let $G = \{x \in R^n \mid g(x) > -\infty\}$. Clearly $P \subseteq G$. It can be shown that $E(P) \neq \emptyset$ if and only if $g(x)$ is finite for every $x \in G$. Assume that $E(P) \neq \emptyset$. Then g is a nonnegative, piecewise linear, continuous and concave function over G. In addition, problem (6.2) is equivalent to (1.1).

References

[1] Bard, J.F. (1984), Optimality conditions for the bilevel programming problem, *Naval Research Logistics Quarterly* **31**, 13-26.

[2] Benson, H.P. (1991), An all-linear programming relaxation algorithm for optimizing over the efficient set, *Journal of Global Optimization* **1**, 83-104.

[3] Benson, H.P. (1992), A finite, nonadjacent extreme-point search algorithm for optimizing over the efficient set, *Journal of Optimization Theory and Applications* **73**, 47-64.

[4] Candler, W. and Townsley, R. (1982), A linear two-level programming problem, *Computers and Operations Research* **9**, 59-66.

[5] Fülöp, J. (1988), A finite procedure to generate feasible points for the extreme point mathematical programming problem, *European Journal of Operational Research* **35**, 228-241.

[6] Fülöp, J. (1990), A finite cutting plane method for solving linear programs with an additional reverse convex constraint, *European Journal of Operational Research* **44**, 395-409.

[7] Fülöp, J. (1991), A finite cutting plane method for facial disjunctive programs, *ZOR - Methods and Models of Operations Research* **35**, 1-13.

[8] Horst, R. and Tuy, H. (1990), *Global Optimization: Deterministic Approaches*, Springer-Verlag, Berlin.

[9] Judice, J. and Mitra, G. (1988), Reformulation of mathematical programming problems as linear complementarity problems and investigation of their solution methods, *Journal of Optimization Theory and Applications* **57**, 123-149.

[10] Majthay, A. and Whinston, A. (1974), Quasi-concave minimization subject to linear constraints, *Discrete Mathematics* **9**, 35-59.

[11] Steuer, R.E. (1986), *Multiple Criteria Optimization: Theory, Computation, and Application*, Wiley, New York.

A STATISTICAL GENERALIZED PROGRAMMING ALGORITHM FOR STOCHASTIC OPTIMIZATION PROBLEMS

A. A. Gaivoronski (1), E. Messina (2), A. Sciomachen (2)
(1) Italtel, Central Research Laboratories
Castelletto di Settimo Milanese,
20019 Milan, Italy
(2) Dept. Information Sciences, University of Milan
via Comelico 39/41
20135 Milan, Italy

EXTENDED ABSTRACT

We consider here the following stochastic programming problem:

$$\begin{array}{lll} \text{minimize} & E_\omega f(x, \omega) & \\ \text{subject to} & p(x) \le 0, \quad x \in X & \end{array} \qquad (1)$$

and, more specifically, stochastic linear program with recourse ([WETS66], [BIRG86], [KALL76], [PREK73]) which is the problem (1) with $f(x,\omega) = c^T(\omega)x + Q(x,\omega)$, $p(x) = Ax\text{-}b$ and

$$Q(x, \omega) = \min_y \{q^T(\omega)y \mid W(\omega)y = h(\omega)\text{-}T(\omega)x\} \qquad (2)$$

where E_ω denotes expectation with respect to ω, an element of some probability space (Ω, B, P). We assume complete recourse, i.e. (2) always has a solution.

Several methods for solving this problem which combine Dantzig-Wolfe decomposition and statistical techniques were proposed recently ([HIGL91], [GAIV89]). The common feature of these methods is the necessity to solve on each iteration linear or quadratic programming problem which can be of considerable dimension.

In this paper we continue research in the direction of [GAIV89] and propose a specific algorithm for solution of problem (1)-(2) which combines stochastic quasigradient techniques with generalized linear programming.

Let us explain in more details the algorithm we use to solve problem (1)-(2). The generalized programming technique of Wolfe [DANT80] applied to (1) involves grid linearization of $Q(x) = E_\omega Q(x, \omega)$, that has been shown to be convex [WETS66], and requires coordinated solution of a master program and a Lagrangian subproblem defined as follows:

Master:

$$\min \sum_{j=1}^{k} cx^j\lambda_j + \sum_{j=1}^{k} Q(x^j)\lambda_j$$

s. t.

$$\pi^k : \sum_{j=1}^{k} (Ax^j)\lambda_j = b \qquad (3a)$$

$$\nu^k : \sum_{j=1}^{k} \lambda_j = 1$$

$$\lambda_j \geq 0$$

Where π^k, ν^k are the dual multipliers associated with the optimal solution of (3a) and $c = E_\omega c(x, \omega)$.

Subproblem: Find $x^{k+1} \in R^n$ such that $l \leq x^{k+1} \leq u$ and

$$\sigma^k x^{k+1} + Q(x^{k+1}) < \nu^k \qquad (3b)$$

by partially optimizing the problem:

$$\min_{l \leq x \leq u} \sigma^k x + Q(x) \qquad (3c)$$

where $\sigma^k \equiv (c - A^T \pi^k)$.

Hence, the essential features of the method consist of sending the prices of the master to the subproblem that uses them to identify an improved solution x^{k+1}, depending on the previous points $x^1 \ldots x^k$. In this way, a sequence of points $x^0 \ldots x^s$ is generated by the algorithm, which converges to the solution of the problem (1) in a certain probabilistic sense.

The problem is that it is not possible to compute the values of $Q(x^j)$ and its sugradients exactly, except in some rare cases [NAZA86]. More generally, $Q(x^j)$ can be approximated, for example by a sample procedures. What we

do is to replace $Q(x^j)$ by an estimate, say Q_k^j. This substitution does not affect the solution of the Master problem too much and for the Subproblem it is not relevant if we do not optimize at each iteration, but we just look for a point that satisfies relation (3b). This suggests to use statistical techniques for the subgradient estimation in order to find the next improving point x^{k+1}. In particular, we use stochasic quasi-gradient procedures because of its effectiveness in solving problems that has not to be pushed all the way to optimality ([ERMO76], [GAIV88]).

The fundamental steps of the proposed algorithm are illustred below.

STEP 1: (Initialize)

Choose a set of m grid points $x^1,...,x^m$ so that constraints

$$\begin{aligned} \sum_{j=1}^{k} (Ax^j)\lambda_j &= b \\ \sum_{j=1}^{k} \lambda_j &= 1 \\ \lambda_j &\geq 0 \end{aligned} \qquad (4)$$

have a feasible solution. The simplest way to find them is to fix the values of λ^j and start to solve the problem, of minimization of the objective function $\sum_{j=1}^{k} cx^j\lambda_j$, for example using the simplex algorithm. The basic solutions found at the first m iterations may be a good choice for initial points.

Set $k \leftarrow m$.

STEP 2: (form estimates)

Define a subset N_k of integers, $N_k \subset \{1,...,k\}$, which are the indices of the set of grid points for which the estimates will be made, and the number s(k) which controls the precision of estimates Q_k^j of $Q(x^j)$, in the following way:

- $s(m) = s_0$

- define a sequence $\{k_p\}_{p=1}^{\infty}$, with $k_{p+1} > k_p$, and

  IF $k = k_p$(for some k_p) THEN $N_k = \{1,...,k\}$

  $s(k) = s(k-1) + 1$

  ELSE $N_k = \{k\}$

  $s(k) = s(k-1)$

Elements of k_p define the iterations in which estimates have to be updated at all of the available grid points. During all other iterations only the estimation at the last point is performed.

- Q_k^j, with $j \in N_k$, is updated by:

  IF $k = m$ THEN ($\forall j < k$)

  $$Q_k^j = \frac{1}{s0}\sum_{i=1}^{s0} Q(x^j,\omega^i)$$ where ω^i are indipendend observations of ω.

  ELSE $$Q_k^j = \frac{1}{s(k)}\sum_{i=1}^{s(k)} Q(x^j,\omega^i) = \left(1-\frac{1}{s(k)}\right) Q_{k-1}^j + \frac{1}{s(k)} Q(x^j, \omega^{s(k)})$$

  for $j \neq k$.

- The estime at the last grid point to enter in the set is made as:

$$Q_k^k = \left(1-\frac{1}{s(k)}\right) Q_{k-1}^k + \frac{1}{s(k)}\sum_{i=1}^{s(k)} Q(x^k,\omega^i)$$

Such estimates have the property that $[Q(x^j) - Q_k^j] = \varepsilon_k \to 0$ a.s., as $s(k) \to \infty$.

STEP 3: (solve Master)

- Solve the master problem (3a) in order to obtain the value of the dual multpliers, π^k and ν^k, and of the primal variables λ_j^k.

At this point the set $\Lambda_k = \{j: \lambda_j^k > 0\}$ can be defined such that it is possible to redefine the set N_k avoiding to update the estimates in those points having $j \notin \Lambda_k$.

STEP 4: (Define a new grid point x^{k+1})

- Define $\sigma^k \equiv (c - A^T \pi^k)$ and consider the Lagrangian subploblem (3c).
- Fix the number S_k of iterations and compute a sequence of points using the stochastic quasi-gradient method as follows. Here d_s are the optimal dual variables associated with solution of the problem (2) for $x = x_k^s$.

$$\xi_k^s = \sigma_k - T^T(\omega^s) d_s$$
$$x_k^{s+1} = P_X(x_k^s - \rho_s \xi_k^s) \quad \text{with}$$
$$s = 0,...,S_k-1, \quad x_k^0 = x_k^k \quad \text{and} \quad x^{k+1} = x_k^{S_k}$$

where P_X is the projection operator over the set X of points belonging to the feasible region.

STEP 5: (Iterate)

$k \leftarrow k + 1$. Return to Step 2.

It is important to note that it has been possible to apply the generalized linear programming to stochastic problem introducing two important modifications, mentioned above, to the original decomposition method. Firstly, it does not require exact values of the objective function (step 2). It is only necessary to have estimates of the objective values at the grid points with their precision gradually increasing. Secondly, it is not necessary to minimize the Lagrangian subproblem at step 4, precisely, it is only necessary that current point x^{k+1} regulary comes to the vicinity of such a solution.

The convergence of this algorithm to the set of optimal solution of problem (1)-(2) with probability 1 follows from results contained in [GAIV89].

Methods using the decomposition techniques have as common feature the necessity to solve at each iteration a large scale linear problem. In order to speed up the computational process we have paid much attention to preprocessing of linear programming subproblems. In particular, we implemented this algorithm by calling each time a processing routine and a linear programming solver of the Optimization Subroutine Library (OSL) [OSL92]. This allows to reduce greatly the problem dimentions and to take advantage from similarity existing among problems arising at different iterations.

Results of numerical experiments are reported in the full paper.

REFERENCES

[BIRG86] Birge, J.R. and Wets, R.J.-B. (1986), "Designing Approximation Schemes for Stochastic Optimization Problems, in particular for Stochastic Programs with Recourse", Math. Programming Study 27, 54-102.

[DANT80] Dantzig, G. B. (1980), "Linear Programming and Extentions", Princeton University Press, Princeton, New Jersey.

[ERMO76] Ermoliev, Yu (1976), "Methods of Stochastic Programming", (in Russian), Nauka, Moscow.

[ERMO88] Ermoliev, Yu and R. J-B Wets (1988), "Numerical Techiniques for Stochastic Optimization", Springer-Verlag, Berlin.

[GAIV88] Gaivoronski A. A. (1988), "Stochastic Quasigradient Methods and their Implementation", in [ERMO88] 313-352.

[GAIV89] Gaivoronski A. A. and J. L. Nazareth, (1989) "Combining Generalized Programming and Sampling Techniques for Stochastic Programs with Recourse, in Resource Planning Under Uncertainty", Stanford University, (G. Dantzig and P. Glynn, eds.)

[HIGL91] Higle J. L. and Sen S. (1991), "Stochastic Decomposition: An Algorithm for Stage Linear Programs with Recourse", Mathematics of Operations Research, 16, pp. 650-669.

[KALL73] Kall P. (1973), "Stochastic Linear Programming", Springer Verlag, Berlin.

[NAZA86] Nazareth, J. L. (1986), "Algorithms Based Upon Generalized Linear Programming for Stochastic Programs with Recourse" [in F. Archetti (ed.), Stochastic Programming: Algorithms and Applications (Gargnano, Italy), Lecture Notes on Control and Information Science, 76, Springer-Verlag, Berlin.

[OSL92] "Optimization Subroutine Library" (1992), Guide and Reference IBM Corporation.

[PREK73] Prekopa A. (1973), "Contributions to the Theory of Stochastic Programming", Math. Programming 4, 202-221.

[WETS66] Wets, R. J-B (1966), "Approximation Formulas for Stochastic Linear Programming", SIAM Journal on Applied Mathematics, 14, 668-677.

A fast-start algorithm for multistage stochastic programs

Horand I. Gassmann
School of Business Administration
Dalhousie University, Halifax, Nova Scotia Canada

Current mailing address:
Division of Economics
Norwegian Institute of Technology
N-7034 Trondheim, Norway

This paper is concerned with the multistage stochastic linear programming problem, written in deterministic equivalent form as

$$\begin{aligned}
\min\ & c_0'x_0 + \sum_{k_1=1}^{K_1} p_{k_1} c_{k_1}' x_{k_1} + \sum_{k_2=K_1+1}^{K_2} p_{k_2} c_{k_2}' x_{k_2} + \cdots + \sum_{k_T=K_{T-1}+1}^{K_T} p_{k_T} c_{k_T}' x_{k_T} \\
\text{s.t. } & A_0 x_0 = b_0 \\
& B_{k_1} x_0 + A_{k_1} x_{k_1} = b_{k_1},\ k_1 = 1,\ldots,K_1 \\
& B_{k_2} x_{a(k_2)} + A_{k_2} x_{k_2} = b_{k_2},\ k_2 = K_1+1,\ldots,K_2 \\
& \qquad\qquad \ddots \qquad\qquad \vdots \\
& B_{k_T} x_{a(k_T)} + A_{k_T} x_{k_T} = b_{k_T},\ k_T = K_{T-1}+1,\ldots,K_T \\
& x_k \geq 0,\ k = 0,\ldots,K_T.
\end{aligned} \tag{1}$$

Nested Benders decomposition (Birge [1], Gassmann [2]) splits this problem into K_T+1 pieces, one for each node in the decision/event tree. Each of these subproblems takes the form

$$\begin{aligned}
\min\ & c_k x_k + \vartheta_k \\
\text{s.t. } & A_k x_k = b_k - B_k x_{a(k)} \\
& F_k x_k \leq f_k \\
& G_k x_k + 1\vartheta_k \leq g_k \\
& x_k \geq 0.
\end{aligned} \tag{2}$$

Here (F_k, f_k) defines *feasibility cuts*, generated by the subproblems beyond k's time stage to ensure their feasibility, and (G_k, g_k) are *optimality cuts* (1 is a column of ones) which cut off non-optimal parts of k's feasible region.

As the number of stages increases, information has to travel through more and more intermediate points to get from the first stage to the horizon and back, and one would expect this process to be quite time-consuming.

The fast-start method described in this paper selects one scenario, that is, a single path from the root of the event tree out to one of the leaf nodes and solves it as a regular LP of reasonable size. If the scenario represents an "average" set of realizations, then one would expect the optimal decision variables for the stochastic problem to be "near" the optimum values for the scenario problem.

In other, words, the scenario solution can be used as a reasonable starting point for the other problems. The important distinction between this approach and scenario aggregation and the progressive hedging algorithm of Rockafellar and Wets [4] is that in the present approach a scenario is not an indivisible unit but is merely seen as a means to an end, namely to find good starting bases for the node problems (2).

The main difficulty lies in disaggregating the scenario solution into the different stages.

Let's look at the two-stage problem first. We can assume without loss of generality that the first scenario consists of nodes 0 and 1, and we can separate the optimal solution x^* into five different components (an optimal solution must exist if the overall problem (1) is to have a solution):

- x^*_{0B} are first stage columns which are basic in first stage rows
- x^*_{0S} are first stage columns which are basic in second stage rows
- x^*_{0N} are first stage columns which are nonbasic (and have value 0)
- x^*_{1B} are basic second stage columns
- x^*_{1N} are nonbasic second stage columns

It is clear from the problem structure that all the components of x^*_{1B}, must be basic in second stage rows. Moreover, if $s = |S|$ is the number of components in the second group, then we need $s + 1$ cuts to force the solution of the node problem (2) to agree with the first stage decisions of the scenario problem. These cuts are derived from the second stage problem

$$\begin{aligned} &\min\ c_1 x_1 \\ &\text{s.t.}\ \ A_1 x_1 \ = b_1 - B_1 x_0 \\ &\qquad\ x_1 \ \geq 0, \end{aligned} \tag{3}$$

using $x_0 = x^*_0$ and the $2s$ perturbations $x^*_0 \pm \delta x^*_{0i}$, for $i \in S$ and some step length δ. At most one of the two directions will yield a cut to the node 0 problem, and the cuts can be both optimality and feasibility cuts. The starting basis for solving (3) is in all cases defined by x^*_{1B}, augmented with slacks corresponding to the deleted variables x^*_{0S}.

However, the cuts in node 0 are only valid for the single scenario problem, so optimality cuts have to be updated ("peeled back" in the language of Higle and Sen [3]) to reflect contributions from the other problems.

There are several ways to perform the update.

A. If the uncertainty is in the right hand side only and if the solved scenario corresponds to the mean value of the realizations (i.e. $b_1 = \sum_{i=1}^{K_1} p_i b_i$), then the cuts are valid without modification. (This is a simple application of Jensen's inequality.)

B. If L_i is a lower bound on the objective value for the node problem at node i, for $i = 1 \ldots K_1$, then the optimality cut

$$\pi_1^* B_1 x_0 + \vartheta_0 \leq \pi_1^* b_1$$

must be adjusted to read

$$p_1 \pi_1^* B_1 x_0 + \vartheta_0 \leq p_1 \pi_1^* b_1 + \sum_{i=2}^{K_1} p_i L_i.$$

C. If π_i^* are dual feasible solutions for all the second stage node problems, then the optimality cut (4) can be adjusted to

$$\sum_{i=1}^{K_1} p_i \pi_i^* B_i x_0 + \vartheta_0 \leq \sum_{i=1}^{K_1} p_i \pi_i^* b_1$$

This form is obviously tighter than B. but involves more work. If the A-matrices and cost coefficients are deterministic, it is of course permissible to use π_1^* throughout.

D. A simplified version of the algorithm dispenses with creating the cuts entirely and simply throws away information about the "superbasic" variables x_{0S}^*.

Multistage problems are similar; the schematic decision tree of figure 1a. may serve as an example with four stages.

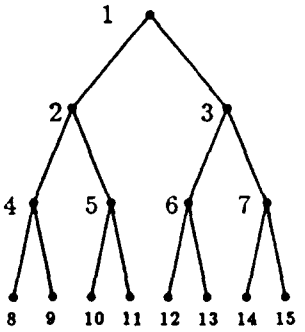


Figure 1a. A four-stage decision tree

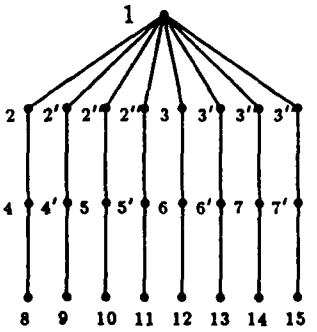


Figure 1b. A two-stage lower bound problem

The scenario problem (1 2 3 4) is solved first and cuts based on the second stage (2 3 4) are created and updated to be valid for the eight-scenario problem indicated in figure 1b. Because the value of information is always nonnegative, this defines a lower bound, and hence the cuts will also be valid for the configuration of figure 1a.

The optimal basis can be copied to the scenario problem (3 6 12), then both are separated out, the cuts are created and updated so as to be valid for each of the four

scenario problems lying downstream from nodes 2 and 3, then one works on the scenarios (4 8), (5 10), (6 12) and (7 14) in the same fashion, and finally one arrives at the fully separated problem with 15 nodes used in the nested Benders decomposition.

It is possible for one of the subsequent scenario problems (3 6 12), (5 10), (6 12), (7 14) to be infeasible. In that case, no optimality cuts should be derived from this scenario (or any of the subsequent scenarios imbedded in them), which may necessitate some modification to the updating of cuts if option C. is used.

Preliminary results on a set of standard test problems show a consistent reduction in CPU time of between 10 and 30% over standard nested decomposition methods.

References

[1] J. R. Birge. Decomposition and partitioning methods for multistage stochastic linear programs. *Operations Research*, 33:989–1007, 1985.

[2] H. I. Gassmann. MSLiP: A computer code for the multistage stochastic linear programming problem. *Mathematical Programming*, 47:407–423, 1990.

[3] J. L. Higle and S. Sen. Stochastic decomposition: An algorithm for two-stage linear programs with recourse. Technical report, Department of Industrial Engineering, University of Arizona, Tucson, Arizona, 1988.

[4] R. T. Rockafellar and R. J.-B. Wets. The principle of scenario aggregation in optimization under uncertainty. Technical Report Working paper WP–87–119, International Institute for Applied Systems Analysis, Laxenburg, Austria, 1987.

Some connections between integer programming and continuous optimization

F. Giannessi, Dept. of Mathematics, Univ. of Pisa, Italy

Abstract.

The purpose of this paper is to stress the importance of analyzing the relationships between combinatorial and continuous optimization. In order to attract more attention to this field some recent and less recent invesitgations will be touched.

First of all we discuss an equivalence property between two extremum problems of type:

$$\min f(x) \quad , \quad x \in Z \cap R, \tag{1}$$

and

$$\min[f(x) + \mu\varphi(x)] \ , \ x \in X \cap R, \tag{2}$$

where $Z \subseteq X$ represents a relaxation.

This property is used to establish an equivalence between a nonlinear integer programming problem of type:

$$\min f(x) \ , \ g(x) \geq 0 \ , \ x \in \{0,1\}^n \tag{3}$$

and a continuous nonlinear programming problem of type:

$$\min[f(x) + \mu x^T(e - x)] \ , \ g(x) \geq 0 \ , \ 0 \leq x \leq e, \tag{4}$$

where $e^T := (1, \ldots, 1)$. A quadratic objective function is considered as a special case.

By means of the preceding relationships it is easy to connect an integer programming problem with a complementarity problem in the form

$$F(x) \geq 0 \ , \ x \geq 0 \ , \ \langle F(x), x \rangle = 0, \tag{5}$$

with a variational inequality in the form

$$\langle G(x), \ y - x \rangle \ \geq 0 \ , \ \forall y \in K, \tag{6}$$

and with a fixed-point problem in the form

$$x = \Phi(x),$$

where the maps F, G, Φ and the set K are defined by the data of (3).

The results which connect (1) and (2) can be generalized and exploited to close the duality gap when the Lagrangean relaxation is applied to a facial constraint.

Some remarks are made about the resolution of problem (4) and of variational inequality (5).

In order to deepen the relationships between combinatorial and continuous optimization we introduce the concept of image of a constrained extremum problem. If this is given in the form

$$\min \varphi(x) \ , \ g(x) \geq 0 \quad x \in X, \ (\varphi : X \to \mathbb{R}; \ g : X \to \mathbb{R}^m), \tag{7}$$

then, given any $\bar{x} \in X$, the image of (7) is the set

$$\mathcal{K} := \{(u, v) \in \mathbb{R} \times \mathbb{R}^m : u = \varphi(\bar{x}) - \varphi(x) \ , \ v = g(x), \ x \in X\}.$$

Now the optimality of $\bar{x}$ is reduced to the disjunction of $\mathcal{K}$ and the set

$$\mathcal{H} := \{(u, v) \in \mathbb{R} \times \mathbb{R}^m : u > 0 \ , \ v \geq 0\}.$$

Separation arguments or alternative ones can be used to show $\mathcal{H} \cap \mathcal{K} = \emptyset$. The fact that separation and alternative may be considered equivalent but different languages for expressing the same thing enables one to reduce, to the same scheme, very different problems, for instance, problem (6). If K is given in the following form:

$$K := \{y \in X : g(y) \geq 0\},$$

where $g : X \to \mathbb{R}^m$, then obviously $x \in K$ solves (6) iff the system (in the variable y)

$$\langle G(x), \ x - y \rangle \ > 0 \ , \ g(y) \geq 0 \ , \ y \in X$$

is impossible. Hence it is easy to introduce the image of (6) as the set

$$\mathcal{K}(x) := \{(u, v) \in \mathbb{R} \times \mathbb{R}^n : u = \langle G(x), x - y \rangle, \ v = g(y), \ y \in X\}.$$

This leads us to associate, to problem (6), the following "gap function"

$$\psi(x) := \min_{\omega \in \Omega} \max_{y \in X} [\langle G(x), x - y \rangle + \gamma(g(y); w)],$$

where $\gamma : \mathbb{R} \times \Omega \to \mathbb{R}$ belongs to a wide class of functions (called separation functions), which includes the linear ones, and represents the generalized Lagrange multiplier approach. The crucial properties of ψ are: $\psi(x) \geq 0 \ \forall x$ and $\psi(x) = 0$ iff x solves (6). Hence we have "two ways connection" between optimization problems and variational inequalities. An interesting application of this is to the study of equilibrium in a network. In some real cases (6) is a better model than (7), since the equilibrium

is better interpreted by well known Wardrop principle. In this case the above equivalence may enable us to exploit the methods of combinatorial optimization for solving a "continuous model" like (6).

References.

* Giannessi F., Nicolucci F., "Connenctions between nonlinear and integer programming problems". Symposia Mathematica, vol. XIX, Academic Press, 1967.

* Hiriart–Urruty J.–B., Lemarechal C., "Testing necessary and sufficient conditions for global optimality in the problem of maximizing a convex quadratic function over a convex polyhedron". Research Paper, Univ. P. Sabatier, Toulouse, 1990.

* Larsen C., Tind J., "Lagrangean relaxation of a facial disjunctive program". Research Paper, AARUS Univ., 1991.

* Fukushima M., "Equivalent differentiable optimization problems and descent methods for asymmetric variational inequality problems". Math. Progr., vol. 53, 1992.

New Results for Aggregating Integer-Valued Equations

FRED GLOVER - School of Business, University of Colorado,
Boulder,CO 80309-0419, USA
DJANGIR BABAYEV - US West Advanced Technologies, 4001 Discovery Drive
Boulder, Colorado 80303, USA

§1. Introduction

Our goal in this paper is to provide new theorems for aggregating general integer-valued equations that can be shown to imply useful analytical results. We give theorems that provide new methods for aggregating equations, and show that each yields significant earlier results as special cases.

A number of references have been devoted to identifying rules for aggregating equations, which determine integer - valued weights for the equations so their linear combination yields a single equation with the same nonnegative integer solution set as the original collection. To emphasize this rigorous equivalence between the original system and the corresponding single equation we refer to this outcome as "integer equivalent aggregation " or IEA. The coefficients of the aggregated equation tend to become exceedingly large as the number of original equations increases, and hence it is desirable to identify weights so these coefficients will lie in a range as limited as possible. Babayev and Mamedov [2] have derived a novel result for integer-valued equations whose right hand sides are equal to 1, yielding what have been conjectured to be the smallest possible weights. Later this result was extended by Knyazev [5] to the case where right hand sides are equal to a common integer value $b \geq 1$. Our new results subsume these earlier results and also give a variety of additional ways to aggregate equations.

§2. Notation and General Results

Let $N = 1,2,...,n$, and X be a subset of the nonnegative n-dimensional integer vectors (as possibly constrained by additional inequalities or equalities of interest), and consider the equations

---

Acknowledgment This research is supported in part by the Joint Air Force Office of Scientific Research and Office of Naval Research Contract No 49620-90-C-0033, at the University of Colorado.

(1) $$\sum_{j \in N} a_{1j} x_j = a_{10}, \qquad x \in X,$$

(2) $$\sum_{j \in N} a_{2j} x_j = a_{20}, \qquad x \in X,$$

where the summations are over components of x, and the coefficients a_{1j} and a_{2j} are integers, not necessarily nonnegative. Define $a_{3j} = w_1 a_{1j} + w_2 a_{2j}$, where w_1 and w_2 are integer weights of the arbitrary sign. Then we seek conditions under which the equation

(3) $$\sum_{j \in N} a_{3j} x_j = a_{30}, \qquad x \in X,$$

has the same solution set as (1) and (2). A collection of equations can thereby be aggregated iteratively taking (3) in the role of (1) and letting each successive equation of the collection (except the first) take the role of (2).

Let X^1 and X^2 be two supersets of X (i.e., $X^1 \supseteq X$ and $X^2 \supseteq X$) and consider the two related equations :

(1a) $$\sum_{j \in N} a_{1j} x_j = a_{10}, \qquad x \in X^1,$$

(2a) $$\sum_{j \in N} a_{2j} x_j = a_{20}, \qquad x \in X^2.$$

Typically (1a) and (2a) each imply a number of linear inequalities in nonnegative coefficients, as, for example, simple upper bounds on some components of x. In general, we will represent any of these inequalities implied respectively by (1a) and (2a), as

(1°) $$\sum_{j \in N} b_{1j} x_j \le b_{10}, \qquad x \in X^1,$$

(2°) $$\sum_{j \in N} b_{2j} x_j \le b_{20}, \qquad x \in X^2,$$

where coefficients b_{1j} and b_{2j} are assumed nonnegative, though not necessarily integer.

Our first major result is

Theorem 1. *For* $k=1,2$ *let* $X_0^k = \{ x \in X^k : x_j = 0$ *for each* $j \in N$ *such that* x_j *is bounded from above by* $x \in X^k \}$ *and let* w_1 *and* w_2 *be relatively prime integers. Then* (3) *is equivalent to* (1) *and* (2) *if*

(4) $w_1 = \sum_{j \in N} a_{2j} x'_j$ *for some* $x' \in X_o^2$ *that violates* (2°), *i.e.,* $\sum_{j \in N} b_{2j} x'_j > b_{20}$,

(5) $w_2 = \sum_{j \in N} a_{1j} x''_j$ *for some* $x'' \in X_o^1$ *that violates* (1°), *i.e.,* $\sum_{j \in N} b_{1j} x''_j > b_{10}$. ♦

Our second result makes use of upper and lower bounds on the left hand sides of the equations (1) and (2).

We introduce a collection of eight inequality conditions by the following notation:

$$
\begin{aligned}
&C_{1u}^{+}: \quad -w_1 > U_2 - a_{20}\ ; && C_{1u}^{-}: \quad w_1 > U_2 - a_{20}\ ; \\
&C_{1l}^{+}: \quad w_1 > -(L_2 - a_{20})\ ; && C_{1l}^{-}: \quad -w_1 > -(L_2 - a_{20})\ ; \\
&C_{2u}^{+}: \quad w_2 > U_1 - a_{10}\ ; && C_{2u}^{-}: \quad -w_2 > U_1 - a_{10}\ ; \\
&C_{2l}^{+}: \quad -w_2 > -(L_1 - a_{10})\ ; && C_{2l}^{-}: \quad w_2 > -(L_1 - a_{10})\ ;
\end{aligned}
\tag{6}
$$

where

$$
U_i = \max\left(\max_{x \in X} \sum_{j \in N} a_{ij} x_j,\ a_{i0}\right), \quad L_i = \min\left(\min_{x \in X} \sum_{j \in N} a_{ij} x_j,\ a_{i0}\right), \qquad i = 1,2. \tag{7}
$$

Each of the inequalities of (6) is strict, with nonnegative right hand sides, so they provide lower bounds for the absolute values of the multipliers w_1 and w_2. The symbols u and l in (6) refer to inequalities based on upper and lower bounds, respectively. Allowing i to take the values 1 and 2, (7) determines U_i and L_i by reference to equations (1) and (2). If we employ the conventions

$$
\begin{aligned}
&+ \Leftrightarrow 1, \qquad u \Leftrightarrow 1, \qquad B_{i1} \Leftrightarrow U_i, \\
&- \Leftrightarrow 2, \qquad l \Leftrightarrow 2, \qquad B_{i2} \Leftrightarrow L_i,
\end{aligned}
$$

then the conditions of (6) can be succinctly represented in the composite form

$$
C_{ip}^{s}: \quad (-1)^{i+p+s} w_i > (-1)^{p-1} (B_{3-i,p} - a_{3-i,0}), \qquad i,p,s \in \{1,2,3\}. \tag{8}
$$

Theorem 2. *Equation* (3) *is equivalent to the system* (1) - (2) *if* w_1 *and* w_2 *are relatively prime integers that satisfy any pair of conditions* (C_{ip}^1, C_{jr}^2), *i.e.,* (C_{ip}^{+}, C_{jr}^{-}), *when the following relation is valid*

(9) $i \neq j \quad or \quad p \neq r.$

If $i = j$ *in the pair* (C^1_{ip}, C^2_{jr}), *then* w_{3-i} *can be given an arbitrary integer value, relatively prime with* w_i. ◆

N o t e 1. For aggregating it is not necessary that the system (1) - (2) be consistent. The theorems of this paper are also valid for systems that are inconsistent, i.e., that have an empty set of solutions.

In computational practice it may be more important to obtain a smaller maximum for the absolute values of the coefficients of the aggregated equation and its right hand side than to restrict the size of multipliers w_1 and w_2. In some cases this can be achieved by increasing the absolute value of one of the multipliers.

N o t e 2. The proof of Theorem 2 discloses that the validity of this theorem requires the left hand sides of equations (1) and (2) to take integer values, which is crucial for the proof. On the other hand, the proof does not rely on the signs of the variables or their integrality, or on the linearity of the left hand sides of the aggregated equations (1) and (2). That means that there is an analog of Theorem 2 that is valid for establishing integer equivalent aggregations of systems of a more general nature.

Let Y be a set of an arbitrary structure and let $S_i \equiv S_i(y), \quad i = 1, 2$ be real integer - valued functions defined for arguments $y \in Y$. Consider a system

(1a) $S_1(y) = a_{10}, \quad y \in Y,$

(2a) $S_2(y) = a_{20}, \quad y \in Y,$

and define

(7a) $U_i = \max(\max_{y \in Y} S_i(y), a_{i0}), \quad L_i = \min(\min_{y \in Y} S_i(y), a_{i0}), \qquad i = 1, 2.$

Then we may state

T h e o r e m 2a. *The equation*

(3a) $w_1 S_1(y) + w_2 S_2(y) = a_{30}$

is equivalent to the system (1a) - (2a) *if* w_1 *and* w_2 *are relatively prime integers that satisfy any pair of conditions* (C^1_{ip}, C^2_{jr}), *i.e.,* (C^+_{ip}, C^-_{jr}), *and relation* (9) *is valid.*

If $i = j$ *in the pair* (C^1_{ip}, C^2_{jr}), *then* w_{3-i} *can be given an arbitrary integer value, relatively prime with* w_i. ◆

N o t e 3. In the proof of Theorem 2 the values U_i and L_i were used for determining lower bounds for the absolute values of the multipliers w_1 and w_2. These bounds in general can be relaxed by obtaining smaller values for U_i and larger values for L_i, $i = 1,2$.

Define

(10) $U_i = \max(\max_{x \in X} S_i,\ a_{i0})$ s.t. $(-1)^{\alpha} S_{3-i} \le (-1)^{\alpha}[a_{3-i,0} - (-1)^{\alpha} sgn(w_i) w_i],\ i = 1,2,$

where $\alpha = 1 - i + s + [(1 + sgn(w_i)]/2$, and the symbols i and s refer to indexes of the condition $C^s_{3-i,p}$ (according to (8) U_i is contained in condition $C^s_{3-i,p}$). The same constraints apply also for determining L_i, $i = 1,2,$

(11) $L_i = \min(\min_{x \in X} S_i,\ a_{i0})$ s.t. $(-1)^{\alpha} S_{3-i} \le (-1)^{\alpha}[a_{3-i,0} - (-1)^{\alpha} sgn(w_i) w_i],\ i = 1,2.$

Then the following theorem is valid

T h e o r e m 2b. The *equation*

(3a) $$w_1 S_1(y) + w_2 S_2(y) = a_{30}$$

is equivalent to the system (1a) - (2a) *if* w_1 *and* w_2 *are relatively prime integers that satisfy any pair of conditions* (C^1_{ip}, C^2_{jr}), *i.e.*, (C^+_{ip}, C^-_{jr}), *when* U_i and L_i *are determined by* (10) *and* (11), *respectively, and relation* (9) *is valid.*

If $i = j$ *in the pair* (C^1_{ip}, C^2_{jr}), *then* w_{3-i} *can be given an arbitrary integer value, relatively prime with* w_i. ♦

C o m m e n t. Theorems 2, 2a and 2b integrate some ideas expressed in [3], [4] and [6].

In Theorem 1 the stipulation on w_1 was used to rule out $q > 0$. The case $q < 0$ was eliminated by the condition on w_2. In Theorem 2 conditions C^+_{ip} and C^-_{ip}, respectively serve the same purposes. From this observation it follows that the aggregation will be valid also under the proper combination of the conditions of Theorem 1 and Theorem 2. This conclusion constitutes the content of the following result.

T h e o r e m 3. *If* w_1 *and* w_2 *are relatively prime integers, then equation* (3) or (3a) *is equivalent to system* (1) - (2) *or* (1a) - (2a), *respectively, if any one of conditions* C^+_{ip} *replaces the stipulation on* w_1 *in Theorem 1, or if any one of conditions* C^-_{ip} *replaces the stipulation on* w_2 *in Theorem 1*. ♦

§ 3. Application to a Special Class of Equations.

We will now show how Theorems 1 and 3 can each be applied to generate the special results of [3] for aggregating the system of equations

$$x_j = b, \qquad j \in N, \tag{12}$$

where b is a positive integer. (In the case $b=1$ we have the result of [2].) The interest in aggregating this system is that the x_j variables can be replaced by any functions $f_j(x)$ that have nonnegative integer values for $x \in X$, thereby making it possible to replace the equations

$$f_j(x) = b, \qquad j \in N,$$

by a single equivalent equation. We state the corresponding result of [5] in an equivalent but slightly different form

Theorem 4[5]. *Define*

$$d_j^{(n)} = [(b+1)^n - (b+1)^{n-j}]/b, \qquad j \in N, \tag{13}$$

$$d_0^{(n)} = [(nb-1)(b+1)^n + 1]/b.$$

Then, for $n>1$ the equation

$$\sum_{j \in N} d_j^{(n)} x_j = d_0^{(n)} \tag{14}$$

has the unique solution, given by (14), *when the x_j variables are constrained to nonnegative integers.* ♦

To show that Theorem 4 is actually a consequence of Theorem 1, we take $X = \{x \geq 0 \text{ and integer } \}$, $X^1 = X^2 = X$ and make use of the following identities:

(i) $$d_j^{(j)} = [(b+1)^j - 1]/b,$$

(ii) $$d_j^{(q)} = (b+1)d_j^{(q-1)}, \quad \text{for } 1 \leq j < q$$

It is easy to show that Theorem 1 yields (14) for $n=2$. As (1) and (2) consider the following equations

$$x_1 = b, \tag{15}$$

$$x_2 = b. \tag{16}$$

The definitions of X and X^k imply $X_o^k = X^k$, $k = 1,2$. Equations (15) and (16) imply the following inequalities (15°) and (16°), which we take as (1°) and (2°):

(15°) $\qquad x_1 \le b,$

(16°) $\qquad x_2 \le b.$

The smallest value of w_1 satisfying (4) is obtained by taking $x_2' = b+1$, which implies $w_1 = b+1$. Then the smallest value of w_2 consistent with condition (5) and relatively prime with w_1 is obtained by assuming $x_1'' = b+2$, which implies $w_2 = b+2$. These values of w_1 and w_2 lead to the aggregating equation (as (3))

$$(b+1)x_1 + (b+2)x_2 = [(2b-1)(b+1)^2 + 1]/b,$$

which is (14) for $n = 2$.

In general, using mathematical induction, we take (1) to be (14) for $n = q-1$, (2) to be $x_q = b$, $X = \{x \ge 0 \text{ and integer } \}$, $X^1 = X^2 = X$ and show that Theorem 1 implies: (3) is (14) for $n = q$.

By Theorem 1 $w_1 = x_q^1$ and the smallest value of w_1 consistent with condition (4) is obtained by taking $x_q' = b+1$ (since $x_q \le b$), which gives $w_1 = b+1$. According to the relation (5)

$$w_2 = d_1^{(q-1)}x_1'' + d_2^{(q-1)}x_2'' + \ldots + d_{q-1}^{(q-1)}x_{q-1}''.$$

Coefficients $d_j^{(q-1)}$ for $j = 1,2,\ldots,q-2$ are divided by $(b+1)$. To obtain w_2 relatively prime with $w_1 = b+1$ it is necessary to take $x_{q-1}'' \ne 0$ and it suffices to choose a value of x_{q-1}'' that yields

$$d_{q-1}^{(q-1)}x_{q-1}'' = 1 \bmod (b+1)$$

or

$$d_{q-1}^{(q-1)}x_{q-1}'' = b \bmod (b+1).$$

As it can be seen from (i) for an arbitrary b the last inequality is obtainable and this occurs only for $x_{q-1}'' = b$. So we assume $x_{q-1}'' = b$. According to (5) x'' should violate (1°), which may be taken in the form $x_j \le b$ for any $j = 1,2,\ldots,q-2$. The smallest of the coefficients $d_j^{(q-1)}$ is $d_1^{(q-1)}$. To obtain a smaller value for w_2 we take $x_1'' = b+1$

and $x_2'' = x_3'' = \ldots = x_{q-2}'' = 0$. Thus $w_2 = d_1^{(q-1)}(b+1) + d_{q-1}^{(q-1)} b = d_q^{(q)}$. The last inequality is obtained with the use of identity (i) and the definition $d_j^{(n)}$, (15). Further (3) and the identity (ii) immediately yields (16) for $n = q$.

Now we shall derive Theorem 4 from Theorem 3. Renumber the variables x_j in (14) in the reverse order by introducing index $p = n - j + 1$ and define

$$c_p^{(n)} = d_{n-p+1}^{(n)} \quad \text{for} \quad p = 1, 2, \ldots, n. \tag{17}$$

Then (14) may be written as

$$\sum_{j \in N} c_p^{(n)} x_p = c_0^{(n)} \equiv d_0^{(n)} \tag{18}$$

Noting $n - j = p - 1$ (13) implies

$$c_p^{(n)} = d_{n-p+1}^{(n)} = [(b+1)^n - (b+1)^{p-1}]/b = (b+1)^{p-1}[(b+1)^{n-p+1} - 1]/b. \tag{19}$$

Then the relation

$$a^m - 1 = (a-1)(a^{m-1} + a^{m-2} + \ldots + 1)$$

for $a = b+1$ and $m = n - p + 1$ gives

$$[(b+1)^{n-p+1} - 1]/b = (b+1)^{n-p} + (b+1)^{n-p-1} + \ldots + (b+1) + 1$$

and from (19) it follows

$$c_p^{(n)} = (b+1)^{n-1} + (b+1)^{n-2} + \ldots + (b+1)^{p-1} = \sum_{k=p-1}^{n-1} (b+1)^k. \tag{20}$$

The last relation implies the following identities

$$c_p^{(p)} = (b+1)^{p-1}, \tag{iii}$$

$$c_p^{(q)} = (b+1)^{q-1} + c_p^{(q-1)}, \quad \text{for} \quad 1 \le p < q. \tag{iv}$$

We shall follow again the reasoning of mathematical induction. As can be seen by substitution for $n = 1$, (18) has a single solution $x_1 = b$. To obtain (18) for an arbitrary $2 \le q \le n$ consider the following system of two equations

$$\sum_{p=1}^{q-1} c_p^{(q-1)} x_p = c_0^{(q-1)}, \tag{21}$$

$$x_1 + x_2 + \ldots + x_q = qb, \tag{22}$$

where (21) is (18) for $n = q-1$ and is taken as (1) in Theorem 3, and (22) is taken as (2) in Theorem 3. Relation (22) is equivalent to $x_q = b$, where, by mathematical induction, it is assumed that equation (21) has a single solution: $x_p = b$ for $p = 1,2,\ldots,q-1$. As in the proof of Theorem 3 select $w_1 = 1$. Then

$$U_1 = \max \sum_{p=1}^{q-1} c_p^{(q-1)} x_p \quad \text{subject to} \quad \sum_{p=1}^{q} x_p \le qb-1. \tag{23}$$

The largest coefficient among $c_p^{(q-1)}$ is $c_1^{(q-1)}$. From (17) and (13)

$$c_1^{(q-1)} = d_{q-1}^{(q-1)} = [(b+1)^{q-1} - 1]/b,$$

In addition, the constraint in (23) implies

$$\sum_{p=1}^{q-1} x_p \le \sum_{p=1}^{q} x_p \le qb-1$$

and from (23) it follows that $U_1 = c_1^{(q-1)}(qb-1)$. Further

$$w_2 > U_1 - a_{10} = c_1^{(q-1)}(qb-1) - d_0^{(q-1)} = (b+1)^{q-1} - q. \tag{24}$$

Finally we require $w_2 = \sum_{p=1}^{q-1} c_p^{(q-1)} x_p''$ and select $x_p'' = 0$ for $p = 1,2,\ldots,q-2$ and $x_{q-1}'' = b+1$. This choice of x'' provides the smallest possible value of w_2, which meets condition (5) of Theorem 1 (violating $x_{q-1} \le b$ as (1°)) and satisfies (24), because as a result of the identity (iii) $w_2 = c_{q-1}^{(q-1)}(b+1) = (b+1)^{q-1}$. Weights $w_1 = 1$ and $w_2 = (b+1)^{q-1}$ for (21) and (22) and the identity (iv) lead straightforwardly to (18) for $n = q$.

Concluding Observation.

It is important to note that the results of [1], [2] and [5] for aggregating the system of n integer - valued equations were obtained from considerations which essentially differ from the approach outlined in §2, in which equations are aggregated step by step two equations at a time. In [1], [2] and [5] for the first time in the literature, analytical formulae were given for weights corresponding to each equation of the system to be aggregated (in [2] and [5] for common equal right hand sides and in [1] for the general case of arbitrary right hand sides). The results presented in this last section reveal

the existence of a tight linkage between the two different approaches to the problem of aggregating integer - valued equations.

REFERENCES

[1] Dj. Babayev, F. Glover, "Aggregation of Nonnegative Integer-Valued Equations", Discrete Applied Mathematics, **8**,125-130, 1984.

[2] Dj. A. Babayev, K. Sh. Mamedov, "Aggregation of a Class of Integer Equations", USSR Computational Mathematics and Mathematical Physics, **18**, 86 - 91, 1979.

[3] G. H. Bradley, "Transformation of Integer Programs to Knapsack Problem", Discrete Mathematics, **1**, No 1, 29 - 45,1971.

[4] Fred Glover and R. E. Woolsey, "Aggregating Diophantine Equations", Zeitschrift für Operations Research, **16**, 1 - 10, 1972.

[5] P. M. Knyazev, "Uniqueness of Integer Nonnegative Solutions of One Linear Equation", Dep. VINITI (USSR), No 4035 - 1979 (Russian).

[6] Manfred W. Padberg, " Equivalent Knapsack-type Formulation of Bounded Integer Linear Programs: an Alternative Approach", Nav. Res. Log. Quart., **19**, 699 - 708, 1972.

HEURISTIC ADVANCES IN OPTIMIZATION INTEGRATING TABU SEARCH, EJECTION CHAINS AND NEURAL NETWORKS

Fred Glover
School of Business, CB 419
University of Colorado
Boulder, CO 80309

Jadranka Skorin-Kapov
Harriman School for Management and Policy
State University of New York at Stony Brook
Stony Brook, NY 11794

1. Background.

Recent developments have shown the value of integrating metaheuristic approaches with special methods for generating new neighborhood structures, called ejection chain methods, and with processes derived from connection models embodied in neural networks. We focus on the implications and consequences of such integrated procedures, with particular attention to the tabu search framework. At the same time, we establish relationships between this framework and that of other metaheuristics, demonstrating ways to enhance "population combining models" (which include genetic algorithms) and "threshold based models" (which include simulated annealing), drawing on search paradigms from tabu search that offer ways to extend these other approaches. We also report developments by which tabu search has provided advances in the uses of neural network models in optimization. Computational studies are cited that confirm the practical merit of these advances.

2. Ejection Chain Processes.

Ejection chain methods give a useful way to build compound neighborhoods, with the goal of generating more powerful moves for solving discrete optimization problems. Ejection chains combine and generalize ideas from a number of sources, including classical alternating paths from graph theory, network related base exchange constructions in matroid optimization, and bounding form structures for solving integer programming problems. Each of these embodies a related theme whose incorporation into neighborhood search offers new approaches to combinatorial optimization applications.

An ejection chain is initiated by selecting a set of elements to undergo a change of state (e.g., to occupy new positions or receive new values). The result of this change leads to identifying a collection of other sets, with the property that the elements of at least one must be "ejected from" their current states. State-change steps and ejection steps typically alternate, and the options for each depend on the cumulative effect of previous steps (usually, but not necessarily, being influenced by the step immediately preceding). In some cases, a cascading sequence of operations may be triggered representing a domino effect. The ejection chain terminology provides a unifying thread that links a collection of useful procedures for exploiting structure, without establishing a narrow membership that excludes other forms of classification.

A number of methods deriving from this perspective recently have appeared in the literature. A *node (or block) ejection* procedure has been proposed by Glover (1991a) for traveling salesman problems, and extended to provide new approaches for quadratic assignment and vehicle routing problems. Laguna, et. al., (1991) introduce an ejection chain approach in conjunction with a tabu search procedure for multilevel generalized assignment problems, and demonstrate that ejection chains even of "small depth" produce highly effective results in this context. Ejection chain strategies are also proposed for clique partitioning of Dorndorf and Pesch (1992), similarly yielding good outcomes.

Recently, ejection chain strategies have been developed for traveling salesman problems that are founded on the notion of creating a reference structure to guide the generation of acceptable moves (Glover, 1992a). Such a structure can be controlled to produce transitions between tours with desirable properties, generating alternating paths (or collections of such paths) of a non-standard yet advantageous type. These paths yield *a combinatorial leverage effect* which provides solutions that are best among exponential numbers of alternatives, by the investment of a low polynomial degree of effort.

In particular, special algorithms enable solutions dominating $O(n2^n)$ alternatives to be obtained with $O(n^2)$ effort, and solutions dominating $O((n/2)!)$ alternatives to be obtained with $O(n^3)$ effort. We show tabu search strategies provide a way to extend the application of these results, leading to new solution procedures not only in the TSP setting, but also for a much broader collection of graph and network-related problems.

3. Links with Other Methods.

Relevant ways to visualize relationships between tabu search and other procedures like simulated annealing and genetic algorithms provide a basis for understanding similarities and contrasts in their philosophies, and for creating potentially valuable hybrid combinations of these approaches. We suggest how elements of tabu search can add a useful dimension to such approaches, drawing on observations from Glover and Laguna (1992). We assume the reader has a modest familiarity with the general form of these approaches as a foundation for the following discussion.

Simulated Annealing. Undoubtedly the most prominent contrast between simulated annealing and tabu search is the focus on exploiting memory in tabu search that is absent from simulated annealing. The introduction of this focus entails associated differences in search mechanisms, and in the elements on which they operate. Several elements in addition to memory are fundamental for understanding the relationship between the methods. We consider three such elements in order of increasing importance.

First, tabu search emphasizes careful probing of successive neighborhoods to identify moves of high quality, employing candidate list approaches. This contrasts with the simulated annealing approach of randomly sampling among these moves to apply an acceptance criterion that disregards the quality of other moves available. (Such an acceptance criterion provides the sole basis for sorting the moves selected in the SA method.) The relevance of this difference in orientation is accentuated for tabu search, since its neighborhoods include linkages based on history, and therefore yield access to information for selecting moves that is not available in neighborhoods of the type used in simulated annealing.

Next, tabu search evaluates the relative attractiveness of moves not only in relation to objective function change, but also in relation to factors of influence. Both types of measures are significantly affected in tabu search by the differentiation among move attributes, as embodied in tabu restrictions and aspiration criteria, and in turn by relationships manifested in recency, frequency, and sequential interdependence (hence, again, involving recourse to memory). Other aspects of the state of search also affect these measures, which depend on the direction of the current trajectory and the region visited.

Finally TS emphasizes guiding the search by reference to multiple thresholds, reflected in the tenures for tabu-active attributes and in the conditional stipulations of aspiration criteria. This may be contrasted to the simulated annealing reliance on guiding the search by reference to the single threshold implicit in the temperature parameter. The treatment of thresholds by the two methods compounds this difference between them. Tabu search varies its thresholds *non-monotonically*, reflecting the conception that multidirectional parameter changes are essential to adapt to different conditions, and to provide a basis for locating alternatives that might otherwise be missed. This contrasts with the simulated annealing philosophy of adhering to a temperature parameter that only changes monotonically.

Hybrids are now emerging that are taking preliminary steps to bridge some of these differences, particularly in the realm of transcending the simulated annealing reliance on a monotonic temperature parameter. A hybrid method that allows temperature to be strategically manipulated, rather than progressively diminished, has been shown to yield improved performance over standard SA approaches, as noted in the work by Osman (1992). A hybrid method that expands the SA basis for move evaluations also has been found to perform better than standard simulated annealing in the study by Kassou (1992). Consideration of these findings invites the question of whether removing the memory scaffolding of tabu search and retaining its other features may yield a viable method in its own right. A foundation for doing this by a "tabu thresholding method" is described in

(Glover, 1992b), and is reported in a study of graph layout and design problems by Verdejo and Cunquero (1992) to perform more effectively than the previously best methods for these problems.

Genetic Algorithms. Genetic algorithms offer a somewhat different set of comparisons and contrasts with tabu search. GAs are based on selecting subsets (usually pairs) of solutions from a population, called parents, and combining them to produce new solutions called children. Rules of combination to yield children are based on the genetic notion of crossover, which consists of interchanging solution values of particular variables, together with occasional operations such as random value changes. Children that pass a survivability test, probabilistically biased to favor those of superior quality, are then available to be chosen as parents of the next generation. The choice of parents to be matched in each generation is based on random or biased random sampling from the population (in some parallel versions executed over separate subpopulations whose best members are periodically exchanged or shared). Genetic terminology customarily refers to solutions as chromosomes, variables as genes, and values of variables as alleles.

By means of coding conventions, the genes of genetic algorithms may be compared to attributes in tabu search, or more precisely to attributes in the form underlying certain TS measures of frequency based memory. Introducing memory in GAs to track the history of genes and their alleles over subpopulations would provide an immediate and natural way to create a hybrid with TS.

Some important differences between genes and attributes are worth noting, however. Differentiation of attributes into *from* and *to* components, each having different memory functions, do not have a counterpart in genetic algorithms. This results because GAs are organized to operate without reference to moves (although, strictly speaking, combination by crossover can be viewed as a special type of move). Another distinction derives from differences in the use of coding conventions. Although an attribute change, from a state to its complement, can be encoded in a zero-one variable, such a variable does not necessarily provide a convenient or useful representation for the transformations provided by moves. Tabu restrictions and aspiration criteria handle the binary aspects of complementarity without requiring explicit reference to a zero-one x vector or two-valued functions. Adopting a similar orientation (relative to the special class of moves embodied in crossover) might yield benefits for genetic algorithms in dealing with issues of genetic representation, which currently pose difficult questions (see, e.g., Liepens and Vose (1990)).

A contrast to be noted between genetic algorithms and tabu search arises in the treatment of context, i.e., in the consideration given to structure inherent in different problem classes. For tabu search, context is fundamental, embodied in the interplay of attribute definitions and the determination of move neighborhoods, and in the choice of conditions to define tabu restrictions. Context is also implicit in the identification of amended evaluations created in association with longer term memory, and in the regionally dependent neighborhoods and evaluations of strategic oscillation.

At the opposite end of the spectrum, GA literature characteristically stresses the freedom of its rules from the influence of context. Crossover, in particular, is a *context neutral* operation, which assumes no reliance on conditions that solutions must obey in a particular problem setting, just as genes make no reference to the environment as they follow their instructions for recombination (except, perhaps, in the case of mutation). Practical application, however, generally renders this an inconvenient assumption, making solutions of interest difficult to find. Consequently, a good deal of effort in GA implementation is devoted to developing "special crossover" operations that compensate for the difficulties created by context, effectively reintroducing it on a case by case basis.

The chief method by which modern genetic algorithms and their cousins handle structure is by relegating its treatment to some other method. That is, genetic algorithms combine solutions by their parent-children processes at one level, and then a different method takes over to operate on the resulting solutions to produce new solutions. These new solutions in turn are submitted to be recombined by the GA processes. In these versions, pioneered by Mühlenbein, Gorges-Schleuter, and Krämer (1988) and also advanced by Davis (1991) and Ulder, et al. (1991), genetic algorithms already take the form of hybrid methods. Hence there is a natural basis for marrying GA and TS procedures in such approaches. But genetic algorithms and tabu search also can be joined in a more fundamental way.

Specifically, tabu search strategies for intensification and diversification are based on the following question: how can information be extracted from a set of good solutions to help uncover additional (and better) solutions? From one point of view, GAs provide an approach for answering this question, consisting of putting solutions together and interchanging components (in some loosely defined sense, if traditional crossover is not strictly enforced). Tabu search, by contrast, seeks an answer by utilizing processes that specifically incorporate neighborhood structures into their design.

Augmented by historical information, neighborhood structures are used in TS as a basis for applying penalties and incentives to induce attributes of good solutions to become incorporated into current solutions. Consequently, although it may be meaningless to interchange or otherwise incorporate a set of attributes from one solution into another in a wholesale fashion, as attempted in recombination operations, a stepwise approach to this goal through the use of neighborhood structures is entirely practicable. This observation provides a basis for creating structured combinations of solutions that embody desired characteristics such as feasibility (Glover, 1991b). The use of these structured combinations makes it possible to integrate selected subsets of solutions in any system that satisfies three basic properties. Instead of being compelled to create new types of crossover to remove deficiencies of standard operators upon being confronted by changing contexts, this approach addresses context directly and makes it an essential part of the design for generating combinations. The current trend of genetic algorithms seems to be increasingly compatible with this perspective, particularly in the work by Mühlenbein (1992), and this could provide a basis for a significant hybrid combination of genetic algorithm and tabu search ideas.

4. Recent Advances Using Tabu Search in Neural Networks.

Our examination of linking tabu search and neural networks in optimization derives from joint work by J. Chakrapani and J. Skorin-Kapov (1992a, 1992b, 1993). There are both benefits and challenges associated with using a massively parallel computer architecture for tabu search, as applied to solving NP hard combinatorial problems. Our starting point for addressing these issues is a connectionist model for solving the quadratic assignment problem (QAP), obtained by modifying Boltzmann machine used by Aarts and Korst (1989). This leads to a massively parallel tabu search algorithm for the QAP, implemented on the Connection Machine CM-2.

Not only is the connection machine extremely interesting as a massively parallel computer architecture for solving combinatorial optimization problems arising in different engineering applications, some optimization problems are also created in the attempt to use the machine as effectively as possible. Depending on the application, the communication time is not trivial. For the QAP, 55% of time is spent on interprocessor communication, which stems from a dynamic communication pattern. There is another class of applications for which the communication pattern is static, i.e. the memory locations defining the source and destinations of messages do not change, only the communicated data changes. In such applications, considerable time can be saved by allocating processors to chips according to the structure of their communication pattern (Dahl, 1990). A tabu search approach to the mapping and scheduling problem successfully improves communication time on a massively parallel system, as demonstrated by computation results on some data sets from the literature.

Connectionist models are constructed to follow the analogy with neural networks in the human brain, and consist of nodes representing neurons, and arcs representing a pattern of connectivity among the neurons. An activity level is associated with each node, and weights or connection strengths are associated with each arc. Activity levels and connection strengths can change according to functions directing the system's behavior. Depending on the values an activity level may take, connectionist model are classified as analog or binary. Boltzmann machines are connectionist models employing binary activity levels, determined probabilistically according to the Boltzmann equation.

With the proper setting of connection strengths, one can establish the equivalence between the objective function of a combinatorial problem and the function governing the behavior of the connectionist model. In such a case the equilibrium points of the system's function correspond to the local minima of the underlying combinatorial problem.

Instead of using simulated annealing to escape from local optima (as in Boltzmann machines), we have designed a related connectionist model in which tabu search is used. This represents the first study replacing simulated annealing with tabu search in a connectionist model, and the first study involving dynamically changing connection strengths for such problems. The results on the set of QAPs from literature show that a connection model based on tabu search performs better than such a model based on simulated annealing.

The connection model itself, though, still does not yield a framework for producing a method as effective as direct heuristic algorithms for the QAP that also utilize tabu search. The inefficiency is due to the fact that any binary matrix is a feasible configuration for a model, and in order to reach a permutation matrix (i.e. a feasible solution to the QAP), large bias connection strengths are needed, which in turn result in poor solutions to the QAP. In order to 'marry' the idea of massively parallel connectionistic approach, and the success of swap moves for the QAPs (which restricts the search to the feasible configurations only), we have progressed to the design of a massively parallel pairwise exchange algorithm implemented on the Connection Machine CM-2.

Apart from designing an elaborate algorithmic strategy using various elements of tabu search (e.g. aspiration, diversification, intensification and varying tabu list sizes), a straightforward implementation on the Connection Machine, obtained by identifying a pairwise exchange with a processor, produces a very inefficient utilization of both time and memory. On the other hand, an efficient implementation requires a fine grain decomposition of the problem into small identical subproblems suitable for data-level parallel computing. Assuming n*n processors, such a decomposition results in only a logarithmic increase in time per iteration as a function of the size of the problem. The logarithmic factor theoretically comes from finding the maximum of n numbers with n processors, which in CM-2 is done by the hardware, and is not significant (e.g. the time increases by 0.1 or 0.2 seconds as the problem size grows from 42 to 100). If the number of processors available is smaller than the neighborhood size, virtual processing is invoked, which requires increased time per iteration, opening possibilities for designing new strategies to handle larger problems.

After successfully solving QAPs of size up to 100, two questions remain: (1) How to 'push' the problem size even further; (2) How to use massively parallel computer architecture to gain more understanding of the tabu strategy itself. We undertake to give answers to these questions by addressing the problem of mapping tasks to processors in a multiprocessor system, in order to minimize communication time. We assume that communication among tasks follows a static pattern, and that all processors are identical and all tasks are similar. We stipulate that the number of processors equals the number of tasks, by introducing "dummy" tasks if there are more processors. The case where there are more tasks is treated within a virtual environment *as if* there are enough processors so that each task can be mapped to a single processor.

We therefore distinguish between the physical nodes of the multiprocessor system and the processors themselves. Each node may contain more than one processor (virtually or otherwise). Processors in the same node can communicate among themselves with minimal time spent in communication (assumed to be zero). Communication time for processors in different nodes is dependent on the architecture of the system. We approximate the communication time between two processors by the number of links that a message has to travel (dilation) and the total communication time by the sum of the individual dilations. Note that even when there are only two nodes, and the communication pattern is a graph with unit edge weights, the problem is NP-hard.

We formulate the problem as a quadratic assignment problem of mapping n tasks to n processors. Denote by D the distance matrix between processors and by F the matrix representing the communication between tasks. Such a formulation is general enough to cover different architectures and different communication patterns. Since this would be a preprocessing routine to an application, emphasis should be given to fast algorithms. Also, since the application will run on a parallel machine, parallel algorithms are preferable to reduce sequential bottlenecks.

We develop a heuristic algorithm based on tabu search for handling this problem. The heuristic employs a simple choice rule and neighborhood structure of iteratively selecting a pair of tasks in a greedy fashion, and swapping the processors to which they are mapped. In our parallel implementation two levels of parallelism are employed. First, the candidate tasks to be swapped are identified in parallel. Second, more than one pair of tasks is swapped in a single iteration. The computed effect of a single swapping is based on the assumption that no other swapping takes place at the current iteration. When performing multiple swaps, the cumulative effect of the swapping may not correspond to the sum of the individual effects. We show how this can lead to an inferior performance and illustrate the elements of our heuristic that makes it robust under these circumstances.

The heuristic is tested on the hypercube architecture, and implemented on CM-2. Computations are performed on data originating from finite element application and of size ranging from 8000 up to 64000 tasks. The result yields a new form of connection approach deriving from an integration with tabu search, and overcomes limitations of earlier connection models in this setting. We anticipate the value of integrating tabu search with connection models for additional types of applications in the future.

Acknowledgedment
This work is supported in part by the Joint Air Force Office of Scientific Research and Office of Naval Research Contract No. F49620-90-C-0033 at the University of Colorado.

References

Aarts, E.H.L. and J.H.M. Korst (1989). "Boltzmann machines for traveling salesman problems," *European Journal on Operational Research* 39, pp.79-95.

Chakrapani, J. and J. Skorin-Kapov (1992a). "A Connectionist Approach to the Quadratic Assignment Problem," *Computers and Operations Research* Vol.19, No.3/4, pp.287--295.

Chakrapani, J. and J. Skorin-Kapov (1992b). "Massively Parallel Tabu Search for the Quadratic Assignment problem," forthcoming in *Annals of Operations Research.*

Chakrapani, J. and J. Skorin-Kapov (1993). "Massively Parallel Heuristic for the Weighted Partitioning of Large, Sparse Graphs," in preparation.

Dahl. D. (1990). " Mapping and Compiled Communication on the Connection Machine System," *Proc. of the Fifth Distributed Memory Computing Conference DMCC*, Charleston.

Davis, L. (ed.) (1991) Handbook of Genetic Algorithms, Van Nostrand Reinhold.

Dorndorf, U. and E. Pesch (1992) "Fast Clustering Algorithms," INFORM and University of Limberg.

Glover, F. (1991a) "Multilevel Tabu Search and Embedded Search Neighborhoods for the Traveling Salesman Probem," to appear in *ORSA Journal of Computing*.

Glover, F. (1991b) "Tabu Search for Nonlinear and Parametric Optimization (with Links to Genetic Algorithms)," to appear in *Discrete Applied Mathematics*.

Glover, F. (1992a) "Ejection Chains, Reference Structures, and Alternating Path Methods for the Traveling Salesman Problem," Graduate School of Business and Administration, University of Colorado at Boulder.

Glover, F. (1992b) "Simple Tabu Thresholding in Optimization," Graduate School of Business and Administration, University of Colorado at Boulder.

Glover, F. and M. Laguna (1992) "Tabu Search," to appear in *Modern Heuristic Methods for Combinatorial Problems*, C.R. Reeves, ed., Blackwell Scientific Publishing.

Kassou, I. (1992) "Amerlioration d'Ordonnancements par des Methodes de Voisinage," doctoral thesis, INSA, Rouen, France.

Laguna, M., J.P. Kelly, J.L. Gonzalez-Velarde, and F. Glover (1991) "Tabu Search for the Multilevel Generalized Assignment Problem," Graduate School of Business and Administration, University of Colorado at Boulder.

Liepins, G. and M.D. Vose (1990) "Representational ISsues in Genetic Optimization," *Journal of Experimental and Theoretical Artificial Intelligence*, 2, 101-115.

Mühlenbein, H., M. Gorges-Schleuter, and O. Krämer (1988) "Evolution Algorithms in Combinatorial Optimization, *Parallel Computing*, 17, 619-632.

Mühlenbein, H. (1992) "Parallel Genetic Algorithms in Combinatorial Optimization," to appear in *Computer Science and Operations Research*, Osman Balci, ed., Pergamon Press.

Osman, I.H. (1992) "Metastrategy Simulated Annealing and Tabu Search Algorithms for the Vehicle Routing Problem," to appear in *Annals of Operations Research*.

Ulder, N.L.J., E. Pesch, P.J.M. Van Laarhoven, H.J. Bandelt, and E.H.L. Aarts (1991) "Genetic Local Search Algorithm for the Traveling Salesman Problem," *Parallel Problem Solving from Nature*, R. Maener and H.P. Schwefell, eds., Lectures in Computer Science 496, Springer-Verlag, 109-116.

Verdejo, V.V. and R.M. Cunquero (1992) "An Application of the Tabu Thresholding Techniques: Minimization of the number of Arc Crossing in an Acyclic Diagraph," Departamento de Estadistica e Investigacion Operativa, Universidad de Valencia, Spain.

An Overview of the Development of An Intelligent Mathematical Programming System

HARVEY J. GREENBERG
Mathematics Department
University of Colorado at Denver
PO Box 173364
Denver, CO 80217-3364

Abstract

Recent advances in computing technology have created a situation where we can solve larger problems than we can understand. This is true for most linear programs, and it is becoming increasingly true for nonlinear and integer forms. In addition, model management can be confounded by large models, especially when they are eclectic. To deal with this bottleneck in productive use of mathematical programming for decision support, a research project began in 1985 to develop an *Intelligent Mathematical Programming System* (IMPS).

Some of the problems we address in the IMPS project are:

- Find a reformulation that simplifies the model.
- Infer data relations that are necessary for the instance to be well posed (i.e., feasible and bounded).
- Give different views of the model, or an instance of it, that provide different insights.
- Answer questions of sensitivity: *What if...?*, *Why ...?*, and *Why not...?* Furthermore, form responses in English, graphics and other forms under user control.
- Provide aids for model debugging, such as why an instance is infeasible.
- Provide aids for documenting a model and scenarios.

The IMPS project is focused on producing new ideas for modeling and analysis support and has produced new approaches to model formulation, management and applications. After elaborating on this background, including our meaning of intelligence and opportunities for creating an intelligent computing environment, we present some examples of results, both positive and negative.

Analysis support is one of the areas the IMPS project has pursued extensively, producing an advanced software system, called ANALYZE, which includes rule-based interpretations of results. The results could be just a prototype instance

without a solution, where analysis looks for reductions and embedded structures in the interest of verification and documentation or other elements of model management. The results could be optimal solutions of scenarios, where analysis not only answers conventional sensitivity questions, but also probes more deeply into the meaning of the results. The results could be infeasible or anomalous, where analysis is debugging the runs to diagnose the cause of the infeasibility or anomaly.

A more recent development is a new approach to the pooling problem, which is a non-convex mathematical program, that gives exact answers to sensitivity questions, rather than the usual methods based on Lagrange multipliers, which can give erroneous answers. We shall demonstrate how this is done using computational geometry.

One of the negative results, which we illustrate, is the use of neural networks for assisting formulation. On the surface, the approach seems reasonable, but we failed to find an appropriate neural net to represent the problems we describe. Although this approach has not been abandoned completely, it has not worked so far, particularly compared with other approaches we have taken, notably syntax-directed modeling assistance.

After describing some of the results, both positive and negative, we summarize current and future activities within the IMPS project. An extensive bibliography is included to indicate the great amount of research and development activity that has emerged over the past few years to address the problems in modeling and analysis.

Equivalence Of Mathematical Programs

HARVEY J. GREENBERG
Mathematics Department
University of Colorado at Denver
PO Box 173364
Denver, CO 80217-3364

FREDERIC H. MURPHY
Department of Management
College of Business
Temple University
Philadelphia, PA 19122

Abstract

When are two mathematical programs equivalent? That is the question addressed in this paper. Since the beginnings of mathematical programming, people have found reformulations of mathematical programs that have certain desirable characteristics. Usually, the reformulation reveals a special structure, such as a network, that speeds up the numerical optimization. More recently, interest in reformulation is motivated by other characteristics, such as for better model management or understanding results more deeply for the decision problem represented by the mathematical program.

The usual way in which one establishes equivalence between the original formulation and another one is to provide mappings between the two formulations that preserve feasibility and optimality. We demonstrate that this mathematical approach to equivalence provides only a necessary condition, not a sufficient one, by showing that it also makes equivalent mathematical programs that really have no relation to each other. Besides the mathematical approach, which has been used for decades, there is a linguistic approach that has emerged more recently. Whereas the mathematical approach provides a necessary condition, the linguistic approach provides a sufficient one. The linguistic approach that has been proposed is too strong to contain necessary conditions. In particular, it does not allow variable re-definition, except in name. The first part of this paper elaborates on these approaches and describes some difficulties with a formal definition of equivalence through examples. Then, representative cases, mostly taken from the literature, are presented in order to build an intuition about equivalence.

The second part considers operational definitions of equivalence and examines their scope. The necessary condition of mappings, taken from the mathematical approach, is included, but a key difference pertains to separation of data from structure. Our motive for having a formal definition of equivalence is to create an artificially intelligent environment for modeling, where model matching can aid an initial formulation, and reformulation can be automatic.

Vladimir A. Gurvich

Extremal integer sequences with forbidden sums

Extended Abstract

1. **Basic notions and notations.** Let $\mathbb{Z}_+ = \{1,2,,,\}$ be the set of the integer positive numbers, $s = (s_1, s_2, \ldots, s_\ell)$ be a finite sequence of numbers from $\mathbb{Z}_+$, then let $\ell = \ell(s)$ be the *length*, $n(s) = s_1 + s_2 + \ldots + s_\ell$ be the *sum* and $m(s) = n(s)/\ell(s)$ be the *mean* of the sequence s. The sequence s' will be called an interval of s and denoted as $s' \subseteq s$, if there exist the numbers $i, j \in \mathbb{Z}_+$ such that $1 \le i \le j \le \ell(s)$ and $s' = (s_i, s_{i+1}, \ldots, s_j)$.

Then let $F \subseteq \mathbb{Z}_+$ be a finite set called the set of the *forbidden sums*. The sequence s will be called *F-excluding* if it does not contain an interval which sum belongs to F. We denote by $EX(\ell; F)$ the set of all the F-excluding sequences of the length l, that is

(1) $EX(\ell; F) = \{s \mid \ell(s) = \ell \text{ and } n(s') \notin F \ \forall\, s' \subseteq s\}$.

We denote by $Gex(F)$ the infinite F-excluding *lexicographically minimal* sequence and by $Gex(\ell; F)$ its initial interval of the length ℓ. "The greedy algorithm" realizes $Gex(F)$ by induction. Successively for each $\ell = 1, 2, \ldots$ the sequence $Gex(\ell; F)$ is obtained from the sequence $Gex(\ell - 1; F)$ by adding the *minimal* number $s_\ell \in \mathbb{Z}_+$ such that $(s_{\ell'} + s_{\ell'+1} + \ldots + s_\ell) \notin F$ for any $\ell' \le \ell$. In particular

(2) $Gex(\ell'; F) \subseteq Gex(\ell; F) \subset Gex(F) \quad \forall\ \ell, \ell' \in \mathbb{Z}_+ \mid \ell' \le \ell$.

In the present paper we shall study the following two functions

(3) $gex(\ell; F) := n(s)$, where $s = Gex(\ell; F)$ and

(4) $ex(\ell; F) := min\,(n(s) \mid s \in EX(\ell; F))$,

realized respectively by the *lexicographically and additively minimal F-excluding sequences* of the length l.

2. **Extremal sequences with forbidden sums and the extremal graph theory.** We can obtain a natural generalization if we replace a set

of the forbidden sums F by a set of the forbidden intervals $\mathcal{F}$. But the most of the results given below will not generalize this case. If $\mathcal{F} = \mathcal{F}(F) := \{s \mid n(s) \in F\}$ then we obtain our problem.

Note an apparent analogy between $ex(\ell; \mathcal{F})$ and the function with the same name studied by extremal graph theory; see for example [1]. Then the function $ex(\ell; F)$ is analogous to that considered in [2].

3. **Basic results.**

Theorem 1. Both functions ex and gex are uniform, that is

(5) $ex(i\ell; iF) = i\, ex(\ell; F)$, $gex(i\ell; iF) = i\, gex(\ell; F)$ $\forall\ i, \ell, F$,

where $iF := \{ia_1, ia_2, \ldots, ia_m\}$ provided $F = \{a_1, a_2, \ldots, a_m\}$; see §7.

Theorem 2. The ratio gex to ex can be estimated by inequalities

(6) $1 \le gex(\ell; F)/ex(\ell; F) \le (\#F + 1)/2$ $\forall\ \ell, F$,

which can not be sharpened for any $\#F$. In particular, if $\#F = 1$ then the equation $gex(\ell; F) \equiv ex(\ell; F)$ holds and if $\#F > 1$ then the second inequality in (6) can be replaced by the strict one; see §§ 4.2, 4.7.

Theorem 3. For any F the sequence $Gex(F)$ is quasiperiodical, that is some its infinite interval is periodical; see § 8.

The *symmetrical* sets defined by the condition

(7) $\exists\ n = n(F) \mid\ a \in F \Leftrightarrow n - a \in F$.

will play an important role. For the symmetrical sets we shall give an asymptotically exact estimation of the functions gex and ex, in other words we shall determine the limits

(8) $m(F) = \lim (ex(\ell; F)/\ell)$ and

$m_g(F) = \lim(gex(\ell; F)/\ell)$ for $\ell \to \infty$.

Theorem 4. The limits (8) exist for any F . In the case of a symmetrical F they are realized by two periodical sequences such that in each one the sum of all the numbers in each period is equal to $n(F)$; see § 6.1.

In particular condition (7) holds for $\#F = 1$ and $\#F = 2$.

Theorem 5. For any set F there exists a symmetrical set F' (respectively F'_g) such that $F \subseteq F'$ (respectively $F \subseteq F'_g$), functions $ex(\ell; F)$ and $ex(\ell; F')$ (respectively $gex(\ell; F)$ and $gex(\ell; F'_g)$) are asymptotically equivalent and realized by the same periodical sequence. Moreover $n(F')$ is always equal to the sum of some two *different* numbers from F.

But $n(F'_g)$ can be much greater; see § 4.6.

4. **Examples**. 4.1. For the set $F = (1,2,\ldots,j)$ we obtain

$Gex(F) = (j+1,\ j+1,\ldots) = (j+1)^{\infty}$; $Gex(\ell;\ F)) = (j+1)^{\ell}$; $EX = \{Gex\}$;

$ex(\ell;\ (1,2,\ldots,j)) = gex(\ell\ ;\ (1,2,\ \ldots,j)) = \ell(j+1)$;

$m(1,2,\ldots,j) = m_g(1,2,\ldots,j) = j+1$;

$ex(i\ell\ ;\ (i,2i,\ldots,ji)) = gex(i\ell\ ;\ (i,2i,\ldots,ji)) = i\ell(j+1)$.

The last formula demonstrates that the functions ex and gex are uniform; see §§ 3,7. In particular for $i = j = 1$ we obtain

$ex(\ell;\ (\ell)) = gex(\ell;\ (\ell)) = 2\ell$; $Gex(\ell;\ (\ell)) = (1)^{\ell-1}(\ell+1)$.

4.2. Let $F = (i)$ be the single-element set. Then

$ex(\ell;\ (i)) = gex(\ell;\ (i)) = 2i\lfloor \ell/i \rfloor + \ell \pmod{i}$;

$Gex((i)) = (1^{\ell-1}(i+1))^{\infty}$; $m(i) = m_g(i) = 2$.

4.3. Let F be an arbitrary set of *odd* numbers. If $1 \in F$ then

$Gex(F) = (2,2,\ldots) = (2)^{\infty}$; $ex(\ell;\ F) = gex(\ell;\ F) = 2\ell \quad \forall\ \ell$.

In any case $m(F) = m_g(F) = 2$. The exclusion of odd sums is analogous to the exclusion of subgraphs which chromatic numbers are not less than 3; see [1].

4.4. The functions ex and gex can be different and even asymptotically not equivalent. The simplest examples are given by the sets (4,7), (4,9), (5,8).

$Gex(4,7) = (1\ 1\ 1\ 8)^{\infty} = (1^3\ 8)^{\infty}$, $m_g(4,7) = 11/4$;

$Gex(4,9) = (1^3\ 5^2)^{\infty}$, $Gex(5,8) = (1^4\ 9)^{\infty}$, $m_g(4,9) = m_g(5,8) = 13/5$.

At the same time $m(4,7) = 11/5$, $m(4,9) = m(5,8) = 13/6$.

The functions $ex(\ell; (4,7))$, $ex(\ell; (4,9))$, $ex(\ell; (5,8))$ are asymptotically realized respectively by the periodical sequences

$(1\ 2\ 3\ 3\ 2)^\infty$, $(1\ 2\ 3\ 2\ 3\ 2)^\infty$, $(1\ 2\ 1\ 3\ 3\ 3)^\infty$

which contain no intervals with forbidden sums. Note that

$ex(5i; (4,7)) = 11i$; $ex(6i; (4,9)) = ex(6i; (5,8)) = 13i \quad \forall\ i$.

4.5. According to Theorem 3, for any set F the sequence $Gex(F)$ is quasiperiodical, that is $Gex(F) = (s_g^0(F))(s_g(F))^\infty$ and the initial interval $s_g^0(F)$ can be not empty. For example,

$Gex(2,4,7) = (1\ 5)(3)^\infty$; $Gex(3,6,10) = (1\ 1\ 7)(4\ 1\ 4\ 4)^\infty$;

$Gex(2,11,12) = (1\ 3\ 1\ 3\ 1\ 5)(4\ 4\ 1\ 4\ 1\ 3\ 1\ 4\ 1)^\infty$.

Introduce the notations $n_g(F) = n(s_g(F))$, $\ell_g(F) = \ell(s_g(F))$. Then $m_g(F) = m(s_g(F)) = n(s_g(F))/\ell(s_g(F)) = n_g(F)/\ell_g(F)$.

4.6. The sequence Gex can have a "very long" period. For example,

$Gex(3,7,10) = ((1\ 1\ 4)\ 8\ (1\ 4\ 1)\ 8\ (4\ 1\ 1)\ 11)^\infty$, $m_g(3,7,10) = 15/4$;

$Gex(3,8,11) = (1\ 1\ 4\ 1\ 9\ 4\ 1\ 1\ 4\ 9\ 1\ 4\ 1\ 1\ 12)^\infty$, $m_g(3,8,11) = 18/5$;

$Gex(4,9,13) = ((1^3\ 5)\ 10\ (1^2\ 5\ 1)\ 10\ (1\ 5\ 1^2)\ 10\ (5\ 1^3)\ 14)^\infty$;

$Gex(j,\ 2ji + 1,\ 2ji + j + 1) = (((1^{j-1}\ (j + 1))^i\ (2ji + 2)$

$(1^{j-2}(j + 1)\ 1)^i\ (2ji + 2)\ldots(1^{j-m}\ (j + 1)\ 1^{m-1})^i\ (2ji + 2)\ldots$

$(1\ (j + 1)\ 1^{j-2})^i\ (2ji + 2)\ ((j + 1)\ 1^{j-1})^i\ (2ji + j + 2))^\infty =$

$= ((\prod_{m=1}^{j}\ ((1^{j-m}\ (j + 1)\ 1^{m-1})^i\ (2ji + 2))) \circledast j\ )^\infty$,

where the sign $\circledast$ means that j is added to the last number of the word. Thus for $F = (j,\ 2ji + 1,\ 2ji + j + 1)$ we obtain

$n_g(F) = j(4ij + 3)$, $\ell_g(F) = j(ij + 1)$, $m_g(F) = 4 - 1/(ij + 1)$.

In general we have no good upper estimate for $n_g(F)$ and $\ell_g(F)$. If the set F is symmetrical then $n_g(F)$ is a divider of $n(F)$.

4.7. "Big ratios" gex/ex are realized by the arithmetic progressions $F = (2i,\ 4i - 1,\ 6i - 2,\ \ldots,\ 2ji - j + 1)$ that have the length $\#F = j$, the step $2i - 1$ and the initial number $2i$. Then $Gex(F) = (1^{2i-1}(2ji - j + 2))^\infty$ and $ex(\ell; F)$ is asymptotically realized by the periodical sequence $(1\ 2\ (2^{i-2}\ 3)^j\ 2^{i-1})^\infty$. Thus

$n_g(F) = n(F) = 2ij+2i-j+1$, $\ell_g(F) = 2i$, $\ell(F) = ij+i-j+1$;

$m_g(F)/m(F) = \ell(F)/\ell_g(F) = (j+1)/2 - (j-1)/(2i)$;

$gex(n_0; F) / ex(n_0; F) = (j+1)/2 - (j-1)/(2i)$ where

$n_0 = \ell_g(F)\ \ell(F)\ n(F) = 2i_(ij+i-j+1)\ (2ij+2i-j+1)$;

$(j+1)/2 - (j-1)/(2i)$ $(j+1)/2$ provided i ∞ ; $j = \#F$.

4.8. Let us explain some notations of §§ 4.1-4.7. The sequences Gex and $Ex \in EX$ are given as words in the alphabet $\mathbb{Z}_+$. Some intervals are marked by the parentheses. The power i over a parenthesis means that the corresponding interval is repeated i times. The power over the last parenthesis can take the value ∞ .

5. **Minimal F-excluding, absolutely F-excluding and F-critical sequences; the asymptotic realization of the function** ex **by an infinite periodical sequence.**

5.1. A finite F-excluding sequence s will be called *minimal* (mFe) if it realizes the value of the function ex , that is if $n(s) = ex(\ell(s); F)$.

Lemma 1. Any mFe sequence s realizes a lower estimate $m(F) \geq m(s) = n(s)/\ell(s)$.

Proof is based on the following evident but important inequality

(9) $ex(i\ell; F) \geq i\ ex(\ell; F)$ $\forall$ i, ℓ, F.

Thus for any i, ℓ, F we obtain

$ex(i\ell; F)/(i\ell) \geq i\ ex(\ell; F)/(i\ell) = ex(\ell; F)/\ell = n(s)/\ell(s) = m(s)$.

5.2. A finite sequence s will be called *absolutely* F-excluding (aFe) if the infinite periodical sequence $(s)^\infty$ is F-excluding. Any aFe sequence is F-excluding but not vice versa.

Lemma 2. Any aFe sequence s realizes an upper estimate $m(F) \leq m(s) = n(s)/\ell(s)$.

Proof. For any i, ℓ, F we obtain $ex(\ell(s); F) \leq n(s)$; $ex(i\ell(s); F) \leq in(s)$; $ex(i\ell(s); F)/(i\ell(s)) \leq in(s)/(i\ell(s)) = m(s)$.

Lemma 3. The following eight properties of the finite sequence s are equivalent:

f) s is aFe ; f′) $(s)^\infty$ is F-excluding;

b, b′) $(s)^i$ is aFe for any (for some) $i \in \mathbb{Z}_+$;

c, c′) the sequence $\sigma(s)$ is aFe for any (for some) *cyclic* permutation σ of the sequence s ;

d) the inverse to s sequence $\bar{s}$ is aFe ;

e) for any number $a \in F$ and interval $s' \subseteq s$ the following inequalities hold: $n(s') \neq a \pmod{n(s)}$ and $n(s') \neq (-a) \pmod{n(s)}$.

5.3. A finite sequence will be called *F-critical* if it is mFe and aFe simultaneously. Lemmas 1 and 2 result in

Proposition 1. For any F-critical sequence s the equation $m(F) = m(s)$ holds. Thus s realizes the asymptotically exact estimate of the function $ex(\ell; F)$.

5.4. Let s be an arbitrary aFe sequence such that $m(s) = m(F)$. Then as a rule the sequence $(s)^i$ will be F-critical for some $i \in \mathbb{Z}_+$. For example the sequence $(3\ 3\ 4)^i$ is F-critical for the set $F = (1,8,9)$ if $i = 5$, but not for $i \leq 4$.

A PC-program of computations of $m(F)$ was realized by S. Tarassov. This method is rather efficient but unfortunately sometimes it fails. The simplest example is given by the set $F = (1,2,6,11)$. In this case there are infinitely many aFe sequences that realize the value $m(F) = 4$; for example, $(4)^\infty$, $(3\ 4\ 5)^\infty$, $(3\ 5\ 4)^\infty$, $(3\ 4\ 3\ 5\ 5)^\infty$, $((3\ 4\ 3)(5\ 4\ 3)^i(5\ 4\ 5))^\infty$ $\forall\ i \in \mathbb{Z}_+$ etc.; see Lemma 3. However neither these sequences nor their powers are not mFe. Really let us consider infinite quasiperiodical sequence $(3\ 4\ 3)(5\ 4\ 3)^\infty$. Inequality $m(s) < 4$ holds for any *initial* interval of s . For the considered set $F = (1,2,6,11)$ there exists no F-critical sequence at all.

5.5. However according to Theorem 5 the value $m(F)$ can be determined by the following formula

$m(F) = min\ (m(F')|\ F'\colon\ F \subseteq F' \in SYM\ ,\ n(F') \le 2max(a|\ a \in F))$.

In other words, the minimum of $m(F')$ is taken over all the *symmetrical* sets F', that contain the set F and satisfy to the inequality $n(F') \le 2max(a|a \in F)$.

In the next section we shall see that it is not difficult to determine the values m and m_g in the case of symmetrical sets.

6. **Symmetrical sets of the forbidden sums.**

6.1. Let F be a symmetrical set; see (7). Consider the set of all the sequences which sums are equal to $n(F)$, that is

$SN(F) = \{s|\ n(s) = n(F)\}$.

Lemma 4. A sequence $s \in SN(F)$ is F-excluding if and only if it is aFe.

Denote the corresponding subset of $SN(F)$ by $SNE(F)$. Chose in $SNE(F)$ the *lexicographically minimal* sequence s_g and an arbitrary sequence s of the *maximal length*. Let $\ell(F) := \ell(s)$, $\ell_g(F) := \ell(s_g)$. Evidently $\ell(F) \ge \ell_g(F)$.

Proposition 2. The sequence s is F-critical and $Gex(F) = (s_g)^\infty$.

This statement results in the equations

(10) $\quad m(F) = n(F)/\ell(F)\ ,\quad m_g(F) = n(F)/\ell_g(F)$.

Thus the periodical sequences $(s)^\infty$ and $(s_g)^\infty$ asymptotically realize the functions ex and gex respectively. Note that by the definition $n(s) = n(s_g) = n(F)$ in accordance with Theorem 4.

6.2. Let $N(F) := \{1, 2, \ldots, n(F) - 1\}$ and $G := N(F) - F$. Evidently the set G is also symmetrical and $N(F) = N(G)$, $n(F) = n(G)$.

Theorem 6. For any symmetrical set the following inequality holds

(11) $\quad n(F) = n(G) \ge \ell(F)\ \ell(G)$.

Formulae (10) and (11) result in

(12) $\quad m(F) = n(F)/\ell(F) \ge \ell(G)\ ,\quad m(G) = n(G)/\ell(G) \ge \ell(F)$;

(13) $\quad m(F)\ m(G) = n^2/(\ell(F)\ \ell(G)) \ge n(F) = n(G)$.

Proposition 3. For any set F we obtain

(14) $m(F) = m_g(F) = 1$ if $F = \varnothing$; $2 \le m(F) \le m_g(F)$ if $F \ne \varnothing$.

Proof for a symmetrical F immediately results from (12).

6.3. Any set F such that $\#F \le 2$ is symmetrical. The case $\#F = 1$ was already considered in § 4.2. Let $F = (i, j)$, then $n(F) = i + j$.

According to Theorem 1 both functions ex and gex are uniform. This statement results in the following equations

(15) $m(i, j) \equiv m(ik, jk)$, $m_g(i, j) \equiv m_g(ik, jk)$ $\forall\ i, j, k \in \mathbb{Z}_+$.

Then $m(i, j) = m_g(i, j) = 2$, if both numbers i and j are odd: see § 4.3. Thus without a loss of generality we can assume that

f) $j \ge i$; b) $GCD(i,j) = 1$; c) i is even, j is odd or vice versa.

Proposition 3. The following formulae hold

$n(i,j) = n_g(i,j) = i+j$;

$\ell(i,j) = \lfloor (i+j)/2 \rfloor$; $\ell_g(i,j) = \lfloor (i+j)/2 \rfloor - \lfloor (j-i)/2 \rfloor (mon\ i)$,

where $a(mon\ b) := min\ (b(mod\ a), -(b+1)(mod\ a)) \in \{0,1,\ldots, \lceil a/2 \rceil - 1\}$;

$m(i,j) = n(i,j)/\ell(i,j) = (i+j)/\lfloor (i+j)/2 \rfloor = 2 + 1/\lfloor (i+j)/2 \rfloor$;

$m_g(i,j) = n_g(i,j)/\ell_g(i,j) = (i+j)/(\lfloor (i+j)/2 \rfloor - \lfloor (j-i)/2 \rfloor (mon\ i))$;

$m_g(i,j)/m(i,j) = 1 / (1 - ((\lfloor (j-i)/2 \rfloor (mon\ i))/\lfloor (i+j)/2 \rfloor) < 3/2$.

The upper estimate $3/2$ can not be sharpened; see § 4.7.

The important peculiar property of the case $\#F = 2$ is the uniqueness of the F-critical sequence. More exactly

Proposition 4. For any i, j there exists the unique (up to the cyclic permutations) (i,j)-excluding sequence s that has the length $\ell(s) = \lfloor (i+j)/2 \rfloor$ and the sum $n(s) = i + j$.

7. **Proof of Theorem 1 (sketch).** Fix an arbitrary sequence $s = (s_1, s_2, \ldots, s_\ell)$ and number $i \in \mathbb{Z}_+$. Consider the sequence

$si := ((1)^{i-1}(s_1 i - i + 1)\ (1)^{i-1}(s_2 i - i + 1)\ \ldots\ (1)^{i-1}(s_\ell i - i + 1))$.

Evidently $\ell(si) = i\ \ell(s)$, $n(si) = i\ n(s)$, $m(si) = m(s)$.

The sequence s is a) F-excluding, b) mFe, c) lexicographically minimal F-excluding if and only if the sequence si is respectively

a') iF-excluding, b') m(iF)e, c') lexicographically minimal iF-excluding. Let us prove for example the implication a) $\Rightarrow$ a'). For any $j = 1,2,\ldots, \ell$ the following equations and inequalities hold

$$s_j i - i + 1 > i(s_j - 1), \quad (s_j - i + 1) + 2(i-1) = i(s_j + 1) - 1 < i(s_j + 1).$$

Thus if for an interval $s' \subseteq si$ the sum $n(s')$ is a multiple of i, then $s' = si''$, where s'' is an interval of s. But $n(s'') \notin F$ because the sequence s is F-excluding. Thus $n(s') = in(s'') \notin iF$.

Consider also the implication b) $\Rightarrow$ b'). The sequence s is mFe. We shall prove that si is m(iF)e. It is proved already that si is iF-excluding. Assume that it is not minimal, that is there exists a sequence s' such that $\ell(s') = i\ell$ and $n(s') < in(s)$. Consider all the initial intervals $s'_j \subseteq s'$ and numbers $n(s'_j) \pmod i$; $j = 1, 2, \ldots, i\ell$. There are $i\ell$ numbers which take only i values. There are two possibilities: either each value occurs exactly ℓ times or there is a value that occurs not less then $\ell+1$ times. In both cases in s' we can outline ℓ successive disjunctive intervals which sums are multiple to i. Replace each interval by the corresponding sum. We obtain a sequence s'' such that $\ell(s'') = \ell$ and $n(s'') < n(s)$. Moreover the sequence s'' is F-excluding because s' was (iF)-excluding. But the sequence s was mFe. Contradiction

8. **Proof of Theorem 3.** Introduce the set of sequences

$$S(F) = \{s \mid n(s) \geq \max\{a \mid a \in F\} > n(s') \quad \forall\, s' \subseteq s\}. \tag{16}$$

It is finite. At the same time the sequence $Gex(F)$ contains infinitely many intervals from S. Consequently $Gex(F)$ contains two equal intervals $s' \cong s''$. Then the interval between the beginnings of s' and s'' is the period of $Gex(F)$. It will be clear if we compare the definition of $S(F)$ by (16) and the definition of $Gex(F)$ by the greedy algorithm; see § 1.

The author thanks S. Tarassov for the PC-program computing ex.

References.

1. Bela Bollobas. Extremal graph theory, Academic Press, 1978.

2. A.I. Golberg and V.A. Gurvich. On the maximum number of edges for a graph with n vertices in which every subgraph with k vertices has at most ℓ edges. Soviet Math. Reports, 1987, v.293, N 1, p.27-32 (Russian). English translation in Sov. Math. Dokl., v. 35 (1987) N 2.

Vladimir A. Gurvich

Russian Academy of Sciences,
International Institute of Earthquake Prediction Theory
and Mathematical Geophysics,

Warshavskoye sh. 79, k.2 , 113556 , Moscow, Russia.

Telefax (095) 310 70 32, telex 411628 MITPA SU;
E-mail mitpan@node.ias.msk.su

Home. Prospect Mira 116 A, 11, Moscow, 129626, Russia;
telephone (095) 287 73 59.

7.11.92

A branch and bound algorithm for discrete programming using the interior point method and the simplex method

M. Hajian

R. Levkovitz

G. Mitra

Brunel, The University of West London

Uxbridge Middlesex

U.K.

1.Introduction

The use of the interior point method (IPM) for the solution of linear programming (LP) problems provides a number of benefits which are summarized below. For large or highly degenerate LPs IPM is usually faster than the simplex solver. Whereas simplex based algorithms require considerable adaptation and control parameter tuning from one model class to another, default settings of IPM are sufficient to process a wide class of LPs. IPM is not only robust in this way, its progress is not hindered by the degeneracy or the stalling problem of the simplex; indeed it reaches the "near optimal" solution very quickly. Simplex algorithms, in contrast, are not affected by the boundary conditions which slow down the convergence of IPM. The fast initial convergence of IPM to a near optimal solution can be followed up by the superior near optimal to optimal convergence of simplex algorithms to create an hybrid IPM-simplex system.

In this paper we review the current use of IPM for the solution of integer programming (IP) problems. We extended our hybrid IPM simplex system (Levkovitz et al. 92) and investigate the role of the IPM within a simplex based branch and bound (B&B) algorithm to discrete programming. An IPM based heuristic is developed whereby the IPM search includes a non-integrality penalty for the discrete variables. The IPM solution is than used to determine an alternative tree search criteria for a simplex based B&B algorithm.

The rest of the paper is organized as follows. In section 2 we review the use of IPM for solving IP problems within the branch and bound algorithm. In section 3 we outline our

proposed heuristic for finding a good starting IP solution. In section 4 we describe the integration of the IPM solution in the B&B algorithm.

2. Interior Point Method in the B&B algorithm

Since the branch and bound (B&B) algorithm creates LP sub-problems that are closely related to each other, a sub-problem is solved by using an optimal basis of a previously solved problem. In this way, the solution of a sub-problem requires only few simplex iterations. A similar efficient warm start procedure for IPM, however, is not yet available (Lustig et al. 90).

The use of IPM in the solution of IP problems was examined by several researchers notably Karmarkar et al. (89) Kamath et al. (89, 91) Mitchell and Todd (91), and Mitchel and Borchers (91). We identify several theoretical advantages of IPM when used in the context of the B&B algorithm. The key advantage of IPM is the ability to reach a near optimal solution or to determine in a relatively short time that such a solution does not exist. If a primal dual algorithm is used, the near optimal solution also provides an upper and a lower bound for the optimal solution at every iteration. A sub-problem can be abandoned (and consequently the branch of the tree in B&B) if the problem is found to be infeasible or if the previously found integer solution is lower than the dual bound. Further, the quick convergence to the solution means that individual variables reach their near optimal value quickly and then converge steadily to the optimal one. This allows the use of indicator functions to distinguish between the dormant variables which converge to their bounds (to an integer solution) and the active variables which remain fractional (El-barki et al 91). The smooth convergence of variables to their optimal value mean that a convergence to a fractional value (active variable) can be discovered early in the iterative process. Using this information we can decide if the problem should be solved to optimality or terminated early. Another important advantage of primal dual IPM is the generation of the solution points by following the path of centres, (Levkovitz 92). The path of centres can be perturbed in a way that will attract it to feasible integer lattice points.

Consider the following pure 0-1 integer programming problem:

$$\begin{aligned} &\min\ c^T x \\ &s.t.\ Ax=b \\ &\quad 0\le x\le u \\ &\quad x_j\in\{0,1\},\ \forall j\in N_1 \end{aligned} \tag{2.1}$$

where $N_1 \subseteq \{1,..,n\}$, the set of indices of the binary variables.

The linear relaxation of this problem and its corresponding dual problem are:

Primal:		*Dual*:	
Min	$c^T x$	*Max*	$b^T y-u^T w$
s. t.	$Ax=b$	*s. t.*	$A^T y+z-w=c$
	$x+s=u,\ x,s\ge 0$		$w,z\ge 0$
	$0\le s_j,x_j\le 1,\ \forall j\in N_3$		

(2.2)

where, $A\in\mathbf{R}^{m\times n}$, $x,s,w,z,c,u\in\mathbf{R}^n$, $y,b\in\mathbf{R}^m$.

Borchers and Mitchell (91) solves (2.1) by applying a specially designed B&B algorithm which utilizes the primal dual predictor corrector IPM to solve the generated sub-problems. The algorithm starts by solving the LP relaxation of (2.2). The optimal solution of this problem provides a lower bound for the best integer solution that can be found. The algorithm then chooses the first sub-problem by using a 'depth first' search tree. Other sub-problems are chosen according to their estimated objective function value such that the sub-problems with lower estimation are solved first.

For every sub-problem, the following operations are carried out:

Algorithm 2.1

1. *Restart the sub problem from a previously saved IPM solution.*
2. *Solve the LP sub problem until* $\dfrac{c^T x-b^T y+u^T w}{|b^T y-u^T w|}<0.1$

   If infeasibility is detected in the process then STOP
3. *If* $b^T y+u^T w > t$ *where* t *is the best integer solution found so far then STOP*
4. *Declare which variables are fractional by using the following indicators: let* $k+1$ *be the current iteration of the IPM algorithm then if one of the following*

conditions holds the 0,1 variable is declared fractional:

$$\left|\frac{x_j^{k+1}}{x_j^k}-1\right|<0.1,\ \left|\frac{s_j^{k+1}}{s_j^k}-1\right|<0.1,\ \frac{w_j^{k+1}}{w_j^k}<0.6,\ \frac{z_j^{k+1}}{z_j^k}<0.6.$$

Mitchell and Borchers implemented this algorithm and compared it to an alternative algorithm that uses cold start IPM and solves every sub-problem to completion. These two algorithms are also compared to the IP module of IBM/OSL (Forrest and Tomlin 90). The preliminary results show that although the algorithm presented above is superior to an algorithm that uses IPM in a simple form, it is still inferior to the OSL/IP algorithm. Analysis of the results show that the performance gap is created by a combination of a longer processing time for sub-problems and a larger number of sub-problems processed in the IPM algorithm.

3. An IPM based heuristic for finding an integer starting point

In our investigations, we use the primal dual IPM in a heuristic to find a feasible or a near feasible starting point for pure 01 IP problems. This method is designed for inclusion in a combined IPM/B&B algorithm where the primal dual iteration data is used as decision criteria in the B&B framework. In our method we try to encourage the binary variables to converge to their bounds by augmenting the objective function with a penalty function on the binary variables that remain fractional. In addition, to avoid the stalling boundary behaviour of IPM, the variables are shifted such that the required optimal solution is in the interior of the LP feasible region.

By this approach, the movement of a variable towards the binary solution can be detected and the binary variables can be fixed to their appropriate value earlier. The heuristics fixes the binary variables one by one while attempting to maintain primal feasibility. Consider the pure 01 IP problem presented in (2.1). We reformulate the problem in the following way : Let $0\le\bar{x}_j\le 1,\ j\in N_I$ be the relaxation of the binary variables $x_j\in\{0,1\},\ j\in N_I$. These variables are shifted by $\epsilon>0$ in the following way:

$$\forall j\in N_I:\quad x_j=\epsilon+\bar{x}_j \rightarrow \epsilon\le x_j\le 1+\epsilon. \tag{3.1}$$

This requires the update of the right hand side values

$$b=\bar{b}+A\xi, \quad \xi_j=\begin{cases}\epsilon & j\in N_I \\ 0 & otherwise\end{cases} \tag{3.2}$$

The bounds of the new variables are further relaxed such that

$$\forall j\in N_I:\ l_j=0,\ u_j=1+2\times\epsilon,\ \epsilon>0. \tag{3.3}$$

It is clear that a solution to the relaxed problem is feasible for the original problem if the variables that represent the binary variables are fixed to either ϵ or $\epsilon+1$.

To achieve this end, we augment the objective function with a penalty function based on the sum of fractions:

$$\min\ c^T x+M\times\sum_{j\in N_I}(x_j-\epsilon)\times(1+\epsilon-x_j), \tag{3.4}$$

where big M is a weight such that $M>c^T x$

The solution process of the new problem is described in algorithm 3.1:

Algorithm 3.1: IPM heuristic for finding an IP solution

1. Initialize: Calculate the first feasible point for the LP relaxation problem in (2.2).

2. Execute IPM (predictor corrector).
If (all the binary variables are fixed or infeasibility is detected) then STOP

3. Find a variable for fixing $v\in G$: $x_v=\min(\min_{j\in G}\{|x_j-\epsilon|\},\min_{j\in G}\{|1+\epsilon-x_j|\})$

4. Remove the variable from the problem
if $|x_v-\epsilon|\le|1+\epsilon-x_v|$ *then* $x_v=\epsilon$ *else* $x_v=1+\epsilon$

set $G=G\setminus\{v\}$, $A=A\setminus\{a_j\}$, $b=b-\epsilon\times a_j$, $c=c\setminus\{c_j\}$

5. Reinstate the dual problem

6. Go to step 2

The procedure described in algorithm 3.1 continues until all the integer variables are fixed or if it is established that the heuristic does not lead to an integer feasible solution.

In both cases the last feasible solution is converted to simplex using the basis retrieval procedure. Preliminary results of our heuristic on a small set of problems with binary variables are presented in Table 3.1 The first four columns give the model name, number of rows, number of columns and number of binary variables respectively. The columns marked 'LP sol' and 'IP sol' give the optimal LP solution and the best integer solution respectively. The column marked IPMH presents the result of running the IPM heuristic to termination. The last column in the table indicates the sum of infeasibilites where 10^{-8} is the feasibility tolerance. These results, although preliminary, indicate that the use of IPM as an alternative heuristic to generate a first feasible integer solution in the branch and bound procedure offers some advantages.

Model	Rows	Cols	Binary	LP Sol	IP Sol	IPMH	Inf.
Gray2	35	48	24	185.55	202.3	241.55	10^{-8}
Gray9	63	96	48	256.61	280.95	312.75	10^{-8}
Sam1	8	15	15	8 0	125	130	0
Sam2[1]	46	67	21	24 0	375.0	565.0	10^{-4}
Glbmod	292	422	98	$.204*10^{8}$	$.207*10^{8}$	$.208*10^{8}$	10^{-8}
Bpgas1[1]	211	75	12	-6.79	-5.91	-5.53	10^{-3}

[1] Feasible solution not found

Table 3.1 Result of IPM integer heuristic

4. Using IPM within B&B to Construct a Good Starting Solution

The integration of IPM and simplex has become a leading applied research topic in the area of large scale LPs, see for example (Forest and Tomlin 90) and (Levkovitz 92). The integration at the level of IP tree search with IPM, however, has not been considered or reported so far. In section 2, we noted that the results reported by Borchers and Mitchell (91) are not comparable to those of OSL, mainly because IPM cannot be efficiently warm-started to re-optimise a series of sub-problems. Taking advantage of our basis recovery procedure mainly from IPM to simplex and also making use of a fixing agenda (fix-mix) given by the discrete solution obtained by IPM heuristic, we have created the following procedure for constructing a partially specified search tree (Hajian 92).

The choice of a variable which is partitioned to discrete values for creating the corresponding subproblems and the choice of a sub-problem taken together determine the structure and size of a search tree. We added to our existing B&B solver an algorithmic facility by which a part of the search tree is created rapidly from an initial agenda of the global entities. Such an agenda may be obtained from various sources such as: (i) an existing schedule in case of scheduling, (ii) applying an IPM heuristic described in section, (iii) rounding off a quasi integer solution, or by any other means.

A given agenda does not necessary provide an integer feasible solution but it has to be a near feasible integer solution in order to provide any benefits. A B&B code which is designed to take advantage of a partial or a full agenda of discrete variables may perform more efficient tree search than the one without such a facility. It is obvious that, using a given integer feasible solution in a B&B saves the time spent on the heuristic for obtaining the first integer solution. We call this procedure *fix-mix* which is equivalent to *warm-starting* the B&B algorithm. To fix-mix procedure is described in algorithm 4.1. In this algorithm t^c_{IP} is the value of a possible integer feasible solution (if one gained), t^f_{IP} is the given integer feasible (or near feasible) solution in the fix-mix agenda.

Algorithm 4.1 Fix-Mix (Hajian 92)

0. Initialize: *Solve the first LP relaxation problem. If the solution is integer feasible then STOP.*
else set the lower bound on the objective function to t^f_{IP} (if one exists).

1. Fix: *If the fix-mix agenda is empty then go to step 4, else choose a variable from the fix-mix agenda and fix it to the appropriate value.*

2. Store *Create a subproblem by fixing the variable in previous step to its complementary value and store the subproblem together with the current basis in the stack.*

3. Solve *Solve the current subproblem. If (the solution is integer feasible and the objective function value t^c_{IP}: $t^f_{IP} > t^c_{IP}$) or (the solution is not integer) then go to step 1.*

4. *Execute the standard B&B algorithm.*

The flow chart of Algorithm 4.1 is illustrated in figure 4.1

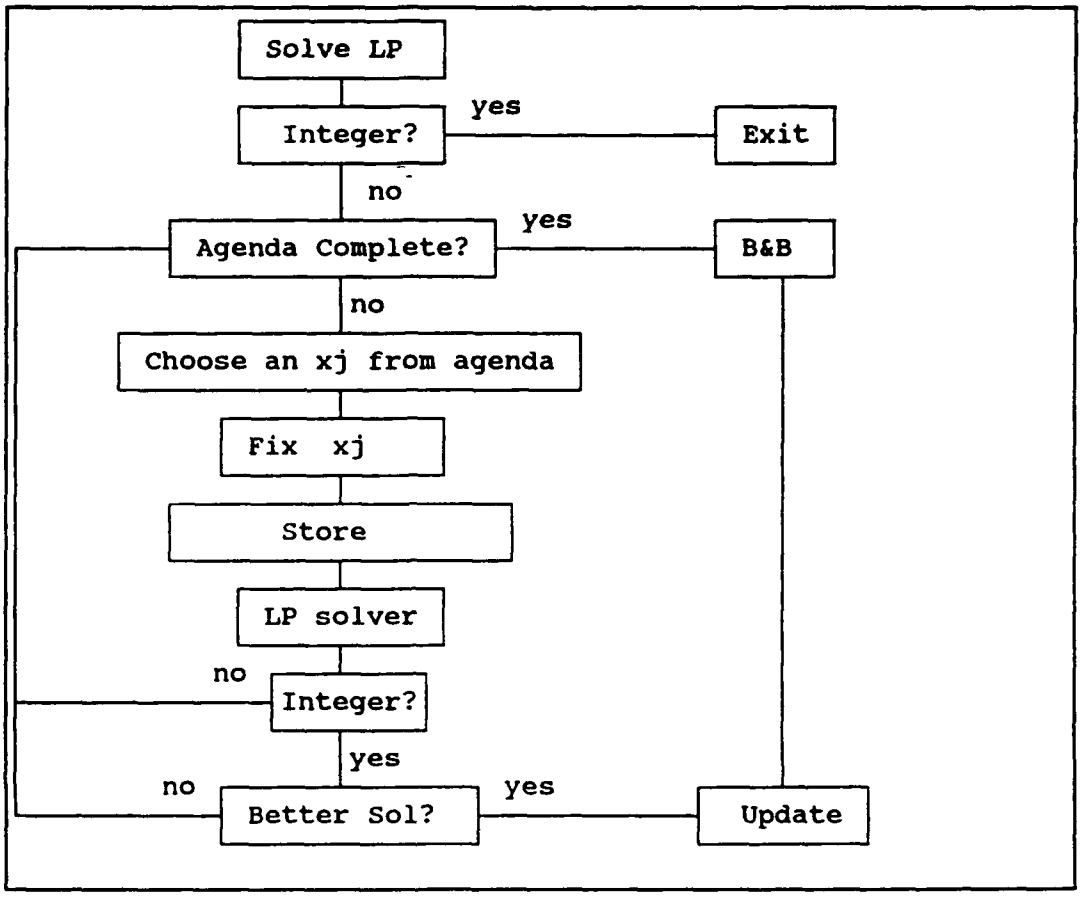


Figure 4.1 The B&B algorithm with Fix-Mix

5. References

(Borchers and Mitchel 91) B. Borchers, J. E. Mitchel, Using an Interior point method in a branch and bound algorithm for integer programming, R.P.I Maths report No. 195, Rensselaer Polytechnic Institute, Troy NY 12180 USA, July 1992.

(El-Barky et al. 91) A.S. El-Barky, R.A. Tapia, Y. Zhang, A study of indicators for identifying zero variables in interior point methods, presented to ICIAM 91, washington D.C. july, 1991.

(Forrest and Tomlin 90) J. Forrest, J.A. Tomlin, OSL Optimization subroutine library version 2.0, user guide and reference, IBM, 1990.

(Hajian 92) M.T. Hajian,Computational methods for discrete programming problems,Ph.D. Thesis, Maths Department, Brunel University Uxbridge MDDX UB8 3PH U.K., December

1992.

(Kamath et al. 89) A.P. Kamamth, N.K. Karmarkar, K.G. Ramakrishnan, M.G.C. Resende, Computational experience with an interior point algorithm on the satisfiability problem, AT&T Bell laboratories, Murray Hill New Jersey 07974, October 1989.

(Kamath et al. 91) A.P. Kamamth, N.K. Karmarkar, K.G. Ramakrishnan, M.G.C. Resende, A continuous approach to inductive inference, AT&T Bell laboratories Murray Hill New Jersey 07974, April 1991.

(Karmarkar et al. 89) N.K. Karmarkar, M.G.C. Resende, K.G. Ramakrishnan, An interior point algorithm to solve computationally difficult set covering problems, Mathematical sciences research center, AT&T bell laboratories, Murray hill, NJ 07974, USA, December 1989.

(Levkovitz 92) R. Levkovitz, An investigation of the interior point method for large scale linear programming, theory and computational algorithms, Ph.D. Thesis, Mathematics department, Brunel University, Uxbridge MDDX UB8 3PH UK., October 1992.

(Levkovitz et al. 92) R. Levkovitz, M. Tamiz, G. Mitra, Experimental investigations in combining IPM and simplex based LP solvers, Mathematics department, Brunel University, Uxbridge MDDX UB8 3PH UK, Presented to the APMOD93 conference, Budapest, January 1993.

(Lustig et al. 90) I. J. Lustig, R. E. Marsten, David F. Shanno, Starting and restarting the primal dual interior point method, Rutcor research report # 61-90, Rutcor, Rutgers University, New Brunswick ,NJ 08903 USA, October 1990.

(Mitchell and Todd 90) J.E. Mitchell, M. J. Todd, Solving matching problems using Karmarkar's algorithm, Contemporary mathematics, Vol. 114, 1990.

A LOGARITHMIC BARRIER CUTTING PLANE METHOD FOR CONVEX PROGRAMMING [1]

(Extended Abstract)

D. den Hertog, J. Kaliski, C. Roos, T. Terlaky [2]

Faculty of Technical Mathematics and Informatics
Delft University of Technology
P.O. Box 5031, 2600 GA Delft, The Netherlands

Key Words: interior point method, linear programming, logarithmic barrier function, convex programming, cutting plane methods, decomposition, nondifferentiable optimization, column generation.

1 Introduction

In this paper we consider the problem of maximizing a linear function over a convex, compact set, defined by convex functions. A logarithmic barrier cutting plane algorithm is developed for solving such problems. Like other cutting plane methods, a linear relaxation of the convex programming problem is generated in each stage of the algorithm. Instead of solving these relaxations, we just try to follow their central path with a long-step logarithmic barrier method. If the next iterate would be out of the feasible region or close to the boundary, then we do not make the step. Then we add a new constraint in the point (or "close" to it) where the step violates a constraint. Obviously, this way we combine Interior Point Methods (IPM's) with some new cutting plane (decomposition) method. As it is known that a convex programming problems is in fact a semi-infinite programming problem, therefore our algorithm can be regarded as an IPM for semi-infinite programming problems.

The first, and still probably the most popular, cutting-plane algorithm for convex programs was developed by Kelley [11]. Here an LP relaxation is solved and an infeasible point is generated in each step. Kelly's method does not work well in practice. The so-called "central cutting plane methods" of Elzinga and Moore [2] and Goffin and Vial [4] are considered quite efficient (see e.g. Kortanek and Ho [14], Goffin, Haurie and Vial [3] and Bahn, Goffin, Vial and Du Merle [1]. They calculate a certain "center" of the LP relaxation. If the center is feasible, then they add an objective cut, if the center is infeasible, then a new separating hyperplane is generated. Therefore these methods generate feasible and infeasible points during the algorithm, although this might not be the case. Contrary to these methods, our algorithm remains in the interior of the feasible set.

The third area to be considered in this paper is the theory of IPM's. Karmarkar [12] has initiated the explosively developing field of IPM's. These methods not only have theoretical interest, but are practically efficient, especially for large and degenerate problems. Jarre [10], Nesterov and Nemirovski [15] and Den Hertog, Roos and Terlaky [7, 8] generalized logarithmic barrier methods to smooth convex programming.

In [6] a build-up strategy for the long-step logarithmic barrier method for LP is presented. In [9] the effect of adding and deleting constraints in the logarithmic barrier method for LP is studied. These algorithms start with a (small) subset of the constraints, and follows the corresponding central path until the iterate is close to (or violates) one of the other constraints. In fact, to be

[1] This work is completed under the support of a research grant of SHELL.

[2] On leave from the Eötvös University, Budapest, and partially supported by OTKA No. 2116.

more precise, if some slack value s_i satisfies $s_i < 2^{-t}$, for some 'proximity' parameter t, then the i-th constraint, if not already in the current system, is added to the current system. This process is repeated until the iterate is close to the optimum.

As far as notations are concerned, e shall denote the vector of all ones, e_i the i-th unit vector and I the identity matrix. Given an $m \times n$ matrix A, its columns are denoted by $a_i, i = 1, \ldots n$. Given an n-dimensional vector s we denote by s^T the transpose of the vector s and the same notation holds for matrices. Finally $\|s\|$ denotes the l_2 norm.

We will consider the following convex programming problem.

$$(CP) \quad \max \quad b^T y$$
$$\text{s.t.} \quad y \in \mathcal{F}.$$

where $y, b \in R^m$ and

$$\mathcal{F} := \{y \in \mathbb{R}^m \ : f_i(y) \leq 0, \ 1 \leq i \leq n\}.$$

The functions $f_i(y)$, $0 \leq i \leq n$, are assumed to be convex. Without loss of generality we may assume that the objective function is linear and we also assume that $\|b\| = 1$ and $\mathcal{F}$ is compact. Further we assume that $\mathcal{F}^0$, the interior of $\mathcal{F}$, is not empty. This condition is equivalent to the Slater condition used by Elzinga and Moore [2].

2 Central Cutting Plane Algorithms

The essence of central cutting plane algorithms can be given as follows.

Central Cutting Plane Algorithm

Input:
$\mathcal{P}$ is a convex polytop such that $\mathcal{F} \subseteq \mathcal{P}$.
z is lower bound for the objective value on $\mathcal{F}$.
ϵ is any small number, the stopping tolerance.
τ is initially a large number (the distance of the center to the boundary).

begin
  while $\epsilon < \tau$ **do**
    Compute (approximately) the "center" y^c of the polytope $\mathcal{P} \cap \{y : b^T y \geq z\}$;
    $\tau := b^T y^c - z$;
    if $y^c \notin \mathcal{F}$, **then Feasibility Cut;**
      else Objective Cut;
    endif
  end
end.

Objective Cut

begin
$z := b^T y^c$, the new objective cut will be

$$b^T y \geq z.$$

end.

Feasibility Cut

begin
$f_i(y^c) > 0$, i.e. f_i is a violated constraint. Add the cut

$$f_i(y^c) + \nabla f_i(y^c)(y - y^c) \leq 0$$

to the current system. $\mathcal{P} := \mathcal{P} \cap \{ y : f_i(y^c) + \nabla f_i(y^c)(y - y^c) \leq 0\}$
end.

Modifying Elzinga and Moore's [2] proof of convergence, we can give a convergence proof for Goffin and Vial's [4] central cutting plane method. We will assume, that an algorithm is available to find the appropriate center in these algorithms. Up to now in the implementations of the algorithm of Elzinga and Moore the simplex method was used to find the center [14] (using the simplex method is not essential, IPM's can be used as well), while Goffin and Vial used the projective method [1].

The convergence proof is based on the following observation. Sonnevend [17] proved, that for any polyhedron $\mathcal{P} \subset \mathbb{R}^m$ there exist two ellipsoids E and E' with $E' = mE$ such that $y^c + E \subset \mathcal{P} \subset y^c + E'$, where y^c is the analytic center of the polytop. Now let us inscribe a sphere into $\mathcal{P}$. It is obvious, that the radius of the largest inscribed sphere is larger or equal than the smallest axis of E and not greater than the smallest axis of E'.

Elzinga and Moore [2] calculate the "ball center" of the actual polytope. This is defined as the center of the largest inscribed sphere. Formally, for a polyhedron $\{y : A^T y \leq c\}$, the ball center is the solution of the following LP problem at iteration k:
$\sigma^k = \max\{\sigma : a_i^T y + ||a_i|\sigma \leq c_i, \ i = 1, \ldots, n\}$,

Goffin and Vial [4] use the so-called "analytic center" [17] of the polytope. The analytic center is the unique maximizer of the logarithmic barrier function:
$\max\{\sum_{i=1}^n \log(c_i - a_i^T y) : A^T y \leq c\}$,

Let ρ^k be the smallest radius of the largest inscribed ellipsoid, centered at the analytic center at iteration k. (It is known [17] that there exists always an inscribed ellipsoid centered in the analytic center, and $\rho^k \leq \sigma^k \leq m\rho^k$ for a given polyhedra.) The following can be proved.

Theorem 1 *(i) For the sequence of the generated centers $\{y^k\}_{k=1}^n$ the sequence ρ^k converges to zero.*
(ii) If $\hat{y} \in \mathcal{F}^0$ then there exists a $\hat{\rho} > 0$ such that $\hat{y}$ is $\hat{\rho}$ feasible for any constraint cut.
(iii) If $\lim_{k\to\infty} \rho^k \to 0$ and $z < z^*$ then there is a $\tilde{k}$ such that $u^k \in \mathcal{F}$ for all $k > \tilde{k}$.
(iv) If $z < z^*$ then every convergent subsequence converges to an optimal solution.

3 A Logarithmic Barrier Cutting Plane Method for Convex Programming

We remind some well-known results for path-following methods for LP [16]. We also use similar ideas then that are used in the Build-Up and Down algorithm of [9]. The algorithm and its analysis is based on the results about the effect of adding and deleting constraints in logarithmic barrier methods, studied in [6, 9].

Now we present our cutting plane algorithm for convex programming problems, which is a straightforward application of the Build-Up and Down Algorithm of [6]. Therefore we try to solve subsequent LP relaxations of (CP). The LP problem under consideration is always the actual relaxation

(localization) of (CP). The algorithm starts from an interior feasible solution. Then, as in path-following methods line searches are performed in the Newton direction of the logarithmic barrier function of the actual LP problem. If the new iterate violates a constraint or is "close" to its boundary, then we stay in the previous point and a new supporting hyperplane is added to improve the approximation. If the current iterate is centered, with a careful deletion rule we try to eliminate redundant constraints and so keep the size of the LP relaxation as small as it is possible.

We assume, that box constraints are included in the problem and index set J refers to the index set of the box constraints. The algorithm goes as follows.

A Logarithmic Barrier Cutting Plane Algorithm

Input:
$\mathcal{P}$ is convex polytop (the initial LP approximation) such that $\mathcal{F} \subseteq \mathcal{P}$;
$\mu := \mu_0$ is a barrier parameter value;
t_c is a convergence parameter;
θ is a reduction parameter, $0 < \theta < 1$;
Q is an initial index set of the (linear) constraints of $\mathcal{P}$;
$y \in \mathcal{F}^0$ is a given interior feasible point such that $\delta_Q(y, \mu) \leq \frac{1}{4}$;

begin
  while $\mu > 2^{-t_c}$ **do**
    begin
      Delete–Constraints;
      $\mu := (1 - \theta)\mu$;
      Center–and–Add–Cut
    end
end.

Procedure Delete–Constraints

Input:
$t_d \geq 4$ is a 'deleting' parameter;

begin
  for $i := 1$ to n **do**
    if $i \in Q \setminus J$ **and** $s_i \geq t_d$ **then**
    begin
      $Q := Q \setminus \{i\}$;
      if $\|p_Q\| \geq \frac{1}{4}$ **then Center–and–Add–Cut**;
    end
end.

Procedure Center–and–Add–Cut

Input:
t_a is an 'adding' parameter;

begin
  while $\delta_Q(y,\mu) > \frac{1}{4}$ **do**
    begin
      $\tilde{y} := y$;
      $\bar{\alpha} := \arg\max_{\alpha>0} \left\{ f_Q(y + \alpha p_Q, \mu) : s_i - \alpha a_i^T p_Q > 0, \forall i \in Q \right\}$;
      $y := y + \bar{\alpha} p_Q$;
      if $\exists k \; : \; -f_k(y) < 2^{-t_a}$ **then Add–Cut**
    end
  $\hat{y} := y$
end.

Procedure Add–Cut

Input:
$\tilde{y} \in \mathcal{F}$ and $y \notin \mathcal{F}$ or y is "close" to the boundary of $\mathcal{F}$;

begin
  If $y \notin \mathcal{F}$ **then** y^b is the boundary point of $\mathcal{F}$ on the line segment $(\tilde{y}, y)$;
  If $y \in \mathcal{F}$ **then** y^b is the boundary point of $\mathcal{F}$ on the halfline $\tilde{y} + \vartheta(y - \tilde{y})$, $\vartheta \geq 0$;
  f_k is a constraint with $f_k(y^b) = 0$;
  $a := \frac{\nabla f_k(y^b)}{\|\nabla f_k(y^b)\|}$, $\gamma = \frac{\nabla f_k(y^b)^T y^b}{\|\nabla f_k(y^b)\|}$;
  $\mathcal{P} := \mathcal{P} \cap \{y : a^T y \leq \gamma\}$
end.

About Convergence

Up to now we could not get a rigorous proof for convergence. The key problem is how to prove that an inner cycle is finite. Namely, if we are centered, then the barrier parameter is reduced. The problem is to prove that after a finite number of iterates we will be centered again. To prove this, probably the results of Den Hertog, Roos and Terlaky [8, 9] can be used.

A straightforward convergence proof for a variant of the above algorithm can be obtained by using the results about the discretization of semi-infinite programming problems (see e.g. Gustafson [5]).

Comparing with Other Cental Cutting Plane Methods

Kelley's [11] cutting plane method generates generates infeasible points and it is known to be instabile, inefficient in practice. Elzinga and Moore's [2] central cutting plane method eliminates these disadvantages. Centering ensures some stability and at least some of the iterates are feasible, therefore the algorithm might be stopped earlier with some useful information in hand. Kortanek and No [14] reports some encouraging computational results. They also used the simplex method to solve the LP relaxations. The central cutting plane method of Goffin and Vial [4] has the advantages of Elzinga and Moore's method. They calculate the analytic center of the polytop with the projective algorithm. The analytic center has some nice properties, which predicts that this method might be even more efficient than the cutting plane method of Elzinga and Moore. Good computational results are reported in [1].

Our method shares the advantages of the above central cutting plane methods. We we start from a feasible point and feasibility is preserved while try to follow the central path of the actual LP relaxation. This provides the centering component in our approach and provides also stability

as above. By adding new cuts, the actual center moves in the opposite direction, but the change cannot be arbitrarily large (see [8, 9]). The same holds if a loose cut is deleted. The main difference comparing with the other approaches is that in the other central cutting plane methods the LP relaxation is fixed while the new center is calculated. We dynamically refine the LP approximation of (CP) in the iterations as we try to get close to the central path.

4 Computational Experience

The implementation of the Logarithmic Barrier Cutting Plane Algorithm was developed to run both in a PC environment (using Microsoft FORTRAN) and in a mainframe platform. The results presented below were computed using FORTRAN on an IBM 3090-200e with vector processing. Similar results (in terms of iteration counts and solution accuracies) are also found on workstation or by the PC based system.

About the Test Problems

We solved 19 test problems. The first 14 problems, taken from [14, 1], are examples of geometric programming problems. The exponential variable transformation ($t_i = e^{y_i}$) is used for each of the problems thereby eliminating the need for explicitly maintaining the positivity constraint $t_i > 0$. Also the objective functions for each of these problems has been made linear. The final 5 problems are examples from [13]. Problems 15,16, and 18 are geometric programs; problem 17 is a semi-infinite program. The majority of the problems which we investigated were small enough to allow an initial feasible point to be found through inspection. Problems 10, 11, 12, and 13 required the use of phase 1.

About the Implementation

In this subsection we discuss several of the important implementation techniques we used in developing our path-following cutting plane system for solving CP. This discussion will center around the main activities performed by the system:

- generating of search directions;
- linesearching and generating cuts;
- determining an initial interior point;
- setting the required parameters;
- terminating the algorithm.

Computational Results

Although the results presented here originate from our system running on the IBM 3090-200e, similar results may be found with the system on the PC and workstation environment.

There are several points which we wish to highlight from Table 1. First there seems to be a remarkable consistency in the major iteration counts listed. This consistency is a direct result from the strategy we use to reduce the log barrier parameter μ. A truer measure of the work required to solve the problem may be found in the number of normal matrix formulations and factorizations required to obtain a given level of solution accuracy. It is apparent that this measure indicates a wide variance in the difficulty the algorithm had in solving the different problems.

Also in Table 1 we see that duality gaps of 10e-9 to 10e-12 can typically be achieved with this method. Although machine accuracy is not obtained, the method presented here does consistently and significantly outperform the method of [14] and our results are also better than presented in [1] in terms of solution accuracy. Note that for problem 13 our method had difficulties obtaining the usual level of precision; a gap of only 9.4e-7 was possible. In this problem, there was an enormous range on the coefficients which exacerbated the numeric difficulties.

Table 1: Problem Results Overview								
Prob. No.	Rows	Col's Iters.	Max. Q Size Factor.	Number of Major Steps	Matrix	Average Recenter Sec.	Duality Gap	Total CPU
1	2	1	15	13	35	2.7	1.2e-12	0.04
2	4	2	25	13	51	3.9	1.8e-12	0.06
3	3	1	20	14	79	5.6	1.3e-11	0.10
4	4	1	30	12	102	8.5	1.7e-10	0.17
5	11	3	60	12	120	10.0	5.7e-9	0.35
6	4	3	25	12	59	4.9	1.4e-9	0.07
7	8	7	45	15	87	5.8	4.0e-12	0.16
8	8	7	45	15	113	7.5	3.8e-12	0.21
9	7	7	56	14	190	13.6	4.7e-11	0.44
10	7	4	53	17	155	9.1	2.8e-9	0.33
11	7	4	45	18	153	8.5	4.5e-10	0.32
12	7	4	53	20	162	8.1	4.9e-11	0.36
13	10	7	94	15	242	16.1	9.4e-7	0.48
14	22	36	161	13	304	23.4	1.1e-9	1.78
15	2	2	10	13	28	2.2	7.2e-12	0.02
16	2	1	9	13	18	1.4	1.7e-12	0.02
17	2	∞	12	13	31	2.4	1.0e-12	0.06
18	3	3	15	13	34	2.6	6.1e-12	0.03
19	3	4	15	14	44	3.1	2.2e-12	0.05

References

[1] Bahn, O., Goffin, J.L., Vial, J.P., and Du Merle, O. (1991), Implementation and Behavior of an Interior Point Cutting Plane Algorithm for Convex Programming: an Application to Geometric Programming, Working Paper, Université de Genève, Genève, Switzerland. To appear in *Annales of Discrete Mathematics*.

[2] Elzinga, J. and Moore, T.G. (1975), A Central Cutting Plane Algorithm for the Convex Programming Problem, *Mathematical Programming*, 8, 134–145.

[3] Goffin, J.L., Haurie, A. and Vial, J.P. (1992), Decomposition and Nondifferentiable Optimization with the Projective Algorithm, *Management Science*, 38, No. 2, 284–302.

[4] Goffin, J.L., and Vial, J.P. (1988), Cutting Planes and Column Generation Techniques with the Projective Algorithm, *Journal of Optimization Theory and Applications*, 65, No. 3, 409–429.

[5] Gustafson, S.-A. (1979), On Numerical Analysis in Semi-Infinite Programming, In "Lecture Notes in Control and Information Sciences, Vol. 15." *Semi-Infinite Programming*, ed. Hettich, R. Springer Verlag, New York, 51–65.

[6] Den Hertog, D., Roos, C., Terlaky, T. (1991), A Build-Up Variant of the Path-Following Method for LP, Report No. 91–47, Faculty of Mathematics and Informatics/Computer Science, Delft University of Technology, Delft, Holland. To appear in *Operations Research Letters*.

[7] Den Hertog, D., Roos, C., Terlaky, T. (1992), A Large Step Analytic Center Algorithm for a Class of Smooth Convex Programming Problems, *SIAM Journal on Optimization*, 2, 55–70.

[8] Den Hertog, D., Roos, C., Terlaky, T. (1992), On the Classical Logarithmic Barrier Method for a Class of Smooth Convex Programming Problems, *JOTA*, 73, 1, 1-25.

[9] Den Hertog, D., Roos, C., Terlaky, T. (1992), Adding and Deleting Constraints in the Logarithmic Barrier Method for Linear Programming Problems, SHELL Report AMER 001. Koniklijke Shell Laboratorium Amsterdam, The Netherlands. Submitted to *Mathematical Programming*.

[10] Jarre, F. (1989), The Method of Analytic Centers for Smooth Convex Programs, Dissertation, Institut für Angewandte Mathematik und Statistik, Universität Würzburg, Würzburg, West-Germany.

[11] Kelley, J.E. Jr. (1960), The Cutting-Plane Method for Solving Convex Programs, *Journal of the Society for Industrial and Applied Mathematics*, 8, 703-712.

[12] Karmarkar, N. (1984), A New Polynomial-Time Algorithm for Linear Programming, *Combinatorica* 4, 373-395.

[13] Kortanek, K.O. (1991), Semi-Infinite Pogramming Duality for Order Restricted Statistical Inference Models, Working Paper, No. 91-18, College of Business Administration, The University of Iowa, Iowa City, Iowa. I

[14] Kortanek, K.O. and No, H. (1992), A Central Cutting Plane Algorithm for Convex Semi-Infinite Programming Problems. Working Paper, No. 90-08, revised February 1992. College of Business Administration, The University of Iowa, Iowa City, Iowa.

[15] Nesterov, Y.E., and Nemirovsky, A.S. (1989), Self-Concordant Functions and Polynomial Time Methods in Convex Programming, Moscow.

[16] Roos, C. and Vial, J.-Ph. (1989), Long Steps with the Logarithmic Penalty Barrier Function in Linear Programming, in *Economic Decision-Making: Games, Economics and Optimization*, dedicated to Jacques H. Drèze, edited by J. Gabszevwicz, J.-F. Richard and L. Wolsey, Elsevier Science Publisher B.V., 433-441.

[17] Sonnevend, Gy. (1985). An "Analytic Centre" for Polyhedrons and New Classes of Global Algorithms for Linear (Smooth, Convex) Programming. *Lecture Notes in Control and Information Sciences*, Springer Verlag, New York, 84, 866-876.

A Unifying Investigation of Interior-Point Methods for Convex Programming [1] [2]

D. den Hertog[a], F. Jarre[b], C. Roos[c] and T. Terlaky[c]

[a]CQM, Vonderweg 11, Building HCZ-3, P.O. Box 414, 5600 AK Eindhoven, NL.

[b]Institut für Angewandte Mathematik und Statistik, Universität Würzburg, Am Hubland, 8700 Würzburg, Germany.

[c]Faculty of Technical Mathematics and Computer Science, Delft University of Technology, P.O. Box 5031, 2600 GA Delft, NL.

Key words: interior-point method, barrier function, dual geometric programming, (extended) entropy programming, primal and dual l_p-programming, relative Lipschitz condition, scaled Lipschitz condition, self-concordance.

1 Introduction

The efficiency of a barrier method for solving convex programs strongly depends on the properties of the barrier function used. A key property that is sufficient to prove fast convergence for barrier methods is the property of self-concordance introduced in [17]. This condition not only allows a proof of polynomial convergence, but numerical experiments in [1, 11, 14] and others further indicate that numerical algorithms based on self-concordant barrier functions are of practical interest and effectively exploit the structure of the underlying problem.

A well-known barrier function for solving convex programs is the logarithmic barrier function, introduced by Frisch [4] and Fiacco and McCormick [3]. To describe the logarithmic barrier function more precisely, we will first give a general form for the classes of problems considered in this paper:

$$(\mathcal{P}) \quad \begin{cases} \min\ f_0(x) \\ f_i(x) \le 0,\ i = 1, \cdots, m, \\ Ax = b, \end{cases}$$

where A is $p \times n$ matrix and b an p-dimensional vector. The logarithmic barrier function for this program is given by

$$\phi(x, \mu) = \frac{f_0(x)}{\mu} - \sum_{i=1}^{m} \ln(-f_i(x)),$$

where $\mu > 0$ is the barrier parameter. We show that for several classes of convex problems for which interior-point methods were presented in the literature the logarithmic barrier function is self-concordant. These classes are: dual geometric programming, (extended) entropy programming, primal and dual l_p-programming. Since for dual geometric programming and dual l_p-programming no complexity results are known in the literature, these self-concordance proofs enlarge the class of problems for which polynomiality can be proved. (In [12] only a convergence analysis is given.) Moreover, we show that some other smoothness conditions used in the literature (relative Lipschitz condition [9, 7], scaled Lipschitz condition [25, 13], Monteiro and Adler's condition [16]) are also covered by this self-concordance condition.

[1]The fourth author is on leave from the Eötvös University, Budapest. Research partially supported by OTKA No. 2116.

[2]This work is completed with the support of a research grant from SHELL.

These observations allow a unification of the analyses of interior-point methods for a number of convex problems.

The article is divided in three parts. In Section 2 we give the definition of self-concordance and state some basic lemmas about self-concordant functions. In Sections 3 - 6 we prove self-concordance for the classes of problems treated in [5, 12, 23], and in Section 7 we show that the smoothness conditions used in [7, 9, 13, 16, 25] imply self-concordance of the barrier function.

2 Some general composition rules

Let us first give the precise definition of self-concordance as given by Nesterov and Nemirovsky [17]:

Definition of self-concordance: Let $\mathcal{F}^0$ be an open convex subset of $\mathbb{R}^n$. A function $\varphi : \mathcal{F}^0 \to \mathbb{R}$ is called κ-self-concordant on $\mathcal{F}^0$, $\kappa \geq 0$, if φ is three times continuously differentiable in $\mathcal{F}^0$ and if for all $x \in \mathcal{F}^0$ and $h \in \mathbb{R}^n$ the following inequality holds:

$$\nabla^3\varphi(x)[h,h,h] \leq 2\kappa \left(h^T\nabla^2\varphi(x)h\right)^{\frac{3}{2}},$$

where $\nabla^3\varphi(x)[h,h,h]$ denotes the third differential of φ at x and h.

Intuitively, since $\nabla^3\varphi$ describes the change in $\nabla^2\varphi$, and since $\nabla^3\varphi$ is bounded by a suitable power of $\nabla^2\varphi$, this condition implies that the *relative* change of $\nabla^2\varphi$ is bounded by 2κ. The associated norm to measure the relative change is given by $\nabla^2\varphi(x)$, i.e. for $h \in \mathbb{R}^n$ the norm associated with the point x is $\|h\|_{\nabla^2\varphi(x)} := (h^T\nabla^2\varphi(x)h)^{1/2}$. (See [10] and [6] for example, where also a brief analysis is given, showing that the property of self-concordance of the barrier function of a convex program is sufficient to prove polynomial convergence. A more detailed analysis that includes certain nonconvex programs and that uses an additional condition relating the first and second derivatives of φ is given in [17].)

The following lemma gives some helpful composition rules for self-concordant functions. It follows immediately from the definition of self-concordance.

Lemma 1 *(Nesterov and Nemirovsky [17])*

- *(addition and scaling) Let φ_i be κ_i-self-concordant on $\mathcal{F}_i^0$, $i = 1,2$, and $\rho_1, \rho_2 \in \mathbb{R}$ then $\rho_1\varphi_1 + \rho_2\varphi_2$ is κ-self-concordant on $\mathcal{F}_1^0 \cap \mathcal{F}_2^0$, where $\kappa = max\{\frac{\kappa_1}{\sqrt{\rho_1}}, \frac{\kappa_2}{\sqrt{\rho_2}}\}$.*
- *(affine invariance) Let φ be κ-self-concordant on $\mathcal{F}^0$ and let $\mathcal{B}(x) = Bx + b : \mathbb{R}^k \to \mathbb{R}^n$ be an affine mapping such that $\mathcal{B}(\mathbb{R}^k) \cap \mathcal{F}^0 \neq \emptyset$. Then $\varphi(\mathcal{B}(.))$ is κ-self-concordant on $\{x : \mathcal{B}(x) \in \mathcal{F}^0\}$.*

The next lemma gives a sufficient condition for an objective function f to guarantee that f "combined" with the logarithmic barrier function for the positive orthant of $\mathbb{R}^n$ is self-concordant.

Lemma 2 *Let $f(x) \in C^3(\mathcal{F}^0)$ be convex. If there exists a β such that*

$$|\nabla^3 f(x)[h,h,h]| \leq \beta h^T\nabla^2 f(x)h\sqrt{\sum_{i=1}^n \frac{h_i^2}{x_i^2}}, \tag{1}$$

$\forall x \in \mathcal{F}^0$ and $\forall h \in \mathbb{R}^n$, then

$$\varphi(x) := f(x) - \sum_{i=1}^{n} \ln x_i$$

is $(1 + \frac{1}{3}\beta)$-self-concordant on $\mathcal{F}^0$, and

$$\psi(\nu, x) := -\ln(\nu - f(x)) - \sum_{i=1}^{n} \ln x_i$$

is $(1+\frac{1}{3}\beta)$-self-concordant on $\hat{\mathcal{F}}^0$. Here, $\hat{\mathcal{F}}^0 \subset \mathbb{R} \times \mathcal{F}^0$ is the set $\{(\nu, x) \mid x \in \mathcal{F}^0,\ \nu > f(x)\}$.

Some discussion of property (1) may be useful. Let $\phi(x) = -\sum_{i=1}^{n} \ln x_i$ be the logarithmic barrier for $\mathbb{R}^n$. Observe that

$$\sqrt{\sum_{i=1}^{n} \frac{h_i^2}{x_i^2}} = \sqrt{h^T \nabla^2 \phi(x) h} = \|h\|_{\nabla^2 \phi(x)}.$$

We recall that (as mentioned above) the canonical norm associated with some barrier function ϕ at a point x is given by $\nabla^2 \phi(x)$. Loosely speaking, property (1) tells us that for $\|h\|_{\nabla^2 \phi(x)} = 1$, the spectral norm of the third derivative $\nabla^3 f$ is bounded by a multiple β of the spectral norm of the second derivative $\nabla^2 f$. This property is defined in [17] as f being compatible with ϕ, and, as we have seen, it implies self-concordance of the combined barrier functions φ and ψ.

Clearly, if f satisfies (1), then so does $\frac{1}{\mu} f$ for any (fixed) parameter $\mu > 0$. In particular, this implies that also the function $f(x)/\mu - \sum \ln x_i$ is $(1 + \frac{1}{3}\beta)$-self-concordant. Finally we note that for any parameter $q \geq 1$, the above proof also holds true for $-q \ln(\nu - f(x)) - \sum_{i=1}^{n} \ln x_i$. This observation can be used to prove that for the classes of problems considered in this paper not only the logarithmic barrier function but also the center function of Huard [8] (also used in e.g. [21, 9, 10, 6]) is self-concordant.

3 The dual geometric programming problem

Let $\{I_k\}_{k=1,\cdots,r}$ be a partition of $\{1, \cdots, n\}$ (i.e. $\cup_{k=1}^{r} I_k = \{1, \cdots, n\}$ and $I_k \cap I_l = \emptyset$ for $k \neq l$). The dual geometric programming problem [2] is then given by

$$(\mathcal{DGP}) \quad \begin{cases} \min\ c^T x + \sum_{k=1}^{r} \left[\sum_{i \in I_k} x_i \ln x_i - \left(\sum_{i \in I_k} x_i \right) \ln \left(\sum_{i \in I_k} x_i \right) \right] \\ Ax = b \\ x \geq 0, \end{cases}$$

where A is an $m \times n$ matrix and c and b are n- and m-dimensional vectors, respectively. For this problem we can prove the following lemma.

Lemma 3 *The logarithmic barrier function of the dual geometric programming problem $(\mathcal{DGP})$ is 2-self-concordant*[3].

[3] This corrects a remark in [12], in which it is claimed that the self-concordance property does not hold for this problem.

4 The extended entropy programming problem

The extended entropy programming problem is defined as

$$(\mathcal{EEP}) \quad \begin{cases} \min\ c^T x + \sum_{i=1}^n g_i(x_i) \\ Ax = b \\ x \geq 0, \end{cases}$$

where A is an $m \times n$ matrix and c and b are n- and m-dimensional vectors, respectively. Moreover, it is assumed that the scalar functions $g_i \in C^3$ satisfy $|g_i'''(x_i)| \leq \kappa_i \frac{g_i''(x_i)}{x_i}$, $i = 1, \cdots, n$. This class of problems is studied in Ye and Potra [23] and Han et al[4] [5]. In the case of entropy programming we have $g_i(x_i) = x_i \ln x_i$, for all i, and $\kappa_i = 1$. Self-concordance for the logarithmic barrier function of this problem simply follows from the following lemma.

Lemma 4 *Suppose that* $|g_i'''(x_i)| \leq \kappa_i \frac{g_i''(x_i)}{x_i}$, $i = 1, \cdots, n$, *then the logarithmic barrier function for the extended entropy programming problem* $(\mathcal{EEP})$ *is* $(1 + \frac{1}{3}\max_i \kappa_i)$*-self-concordant.*

5 The primal l_p-programming problem

Let $\{I_k\}_{k=1,\cdots,r}$ be a partition of $\{1, \cdots, m\}$ (i.e. $\cup_{k=1}^r I_k = \{1, \cdots, m\}$ and $I_k \cap I_l = \emptyset$ for $k \neq l$). Let $p_i \geq 1$, $i = 1, \cdots, m$. Then the primal l_p-programming problem [18, 22] can be formulated as

$$(\mathcal{PL}_p) \quad \begin{cases} \max\ \eta^T x \\ \sum_{i \in I_k} \frac{1}{p_i} |a_i^T x - c_i|^{p_i} + b_k^T x - d_k \leq 0,\ k = 1, \cdots, r, \end{cases}$$

where (for all i and k) a_i, b_k, and η are n-dimensional vectors, and c_i and d_k are real numbers. Nesterov and Nemirovsky [17] treated a special case of this problem, namely the so-called l_p-approximation problem. We will reformulate $(\mathcal{PL}_p)$ such that all problem functions remain convex, contrary to Nesterov and Nemirovsky's reformulation.

In a first step, the primal l_p-programming problem can be reformulated as:

$$\begin{cases} \max\ \eta^T x \\ \sum_{i \in I_k} \frac{1}{p_i} t_i + b_k^T x - d_k \leq 0,\ k = 1, \cdots, r \\ \left.\begin{array}{l} s_i^{p_i} \leq t_i \\ a_i^T x - c_i \leq s_i \\ -a_i^T x + c_i \leq s_i \end{array}\right\} i = 1, \cdots, m \\ s \geq 0. \end{cases} \tag{2}$$

In the same way as we will prove Lemma 5, it can be proved that the logarithmic barrier function for this reformulated l_p-programming problem is $(1 + \frac{1}{3}\max_i |p_i - 2|)$-self-concordant, i.e. the concordance parameter depends on p_i. We can eliminate this dependence as follows. Replace the constraints $s_i^{p_i} \leq t_i$ by the equivalent constraints $s_i \leq t_i^{\pi_i}$, where $0 < \pi_i := \frac{1}{p_i} \leq 1$, and replace the (redundant) constraints $s \geq 0$ by $t \geq 0$. So, we obtain the following reformulated l_p-programming problem:

[4] In [5] it is conjectured that these problems do not satisfy the self-concordance condition. The lemma shows that this conjecture is not true.

$$(\mathcal{PL}'_p) \quad \left\{ \begin{array}{l} \max\ \eta^T x \\ \sum_{i\in I_k} \frac{1}{p_i} t_i + b_k^T x - d_k \le 0,\ k = 1,\cdots,r \\ \left.\begin{array}{l} s_i \le t_i^{\pi_i} \\ a_i^T x - c_i \le s_i \\ -a_i^T x + c_i \le s_i \end{array}\right\} i = 1,\cdots,m \\ t \ge 0. \end{array} \right. \tag{3}$$

Observe that the transformed problem has $4m+r$ constraints, compared with r in the original problem $(\mathcal{PL}_p)$. Now we have the following lemma.

Lemma 5 *The logarithmic barrier function for the reformulated l_p-programming problem $(\mathcal{PL}'_p)$ is $\frac{5}{3}$-self-concordant.*

6 The dual l_p-programming problem

Let q_i be such that $\frac{1}{p_i} + \frac{1}{q_i} = 1$, $1 \le i \le m$, and let the rows of a matrix A be a_i, $i = 1,\cdots,m$, and the rows of a matrix B be b_k, $k = 1,\cdots,r$. Then, the dual of the l_p-programming problem $(\mathcal{PL}_p)$ is (see [18]-[22])

$$(\mathcal{DL}_p) \quad \left\{ \begin{array}{l} \min\ c^T y + d^T z + \sum_{k=1}^r z_k \sum_{i\in I_k} \frac{1}{q_i} \left|\frac{y_i}{z_k}\right|^{q_i} \\ A^T y + B^T z = \eta \\ z \ge 0. \end{array} \right.$$

(If $y_i \neq 0$ and $z_k = 0$, then $z_k|y_i/z_k|^{q_i}$ is defined as ∞.) The above problem is equivalent to

$$\left\{ \begin{array}{l} \min\ c^T y + d^T z + \sum_{i=1}^n \frac{1}{q_i} t_i \\ s_i^{q_i} z_k^{-q_i+1} \le t_i,\ \ i \in I_k,\ \ k = 1,\cdots,r \\ y \le s \\ -y \le s \\ A^T y + B^T z = \eta \\ z \ge 0 \\ s \ge 0. \end{array} \right. \tag{4}$$

It can be proved that the logarithmic barrier function of this reformulated dual l_p-programming problem is $(1 + \frac{\sqrt{2}}{3}\max_i(q_i+1))$-self-concordant. Again, the dependence on q_i can be eliminated: the constraints $s_i^{q_i} z_k^{-q_i+1} \le t_i$ are replaced by the equivalent constraints $t_i^{\rho_i} z_k^{-\rho_i+1} \ge s_i$, where $0 < \rho_i := \frac{1}{q_i} \le 1$, and the redundant constraints $s \ge 0$ are replaced by $t \ge 0$. The new reformulated dual l_p-programming problem becomes:

$$(\mathcal{DL}'_p) \quad \left\{ \begin{array}{l} \min\ c^T y + d^T z + \sum_{i=1}^n \frac{1}{q_i} t_i \\ s_i \le t_i^{\rho_i} z_k^{-\rho_i+1},\ \ i \in I_k,\ \ k = 1,\cdots,r \\ y \le s \\ -y \le s \\ A^T y + B^T z = \eta \\ z \ge 0 \\ t \ge 0. \end{array} \right. \tag{5}$$

Note that the original problem $(\mathcal{DL}_p)$ has r inequalities, and the reformulated problem $(\mathcal{DL}'_p)$ $4m + r$. We now have the following lemma.

Lemma 6 *The logarithmic barrier function of the reformulated dual l_p-programming problem $(\mathcal{DL}'_p)$ is 2-self-concordant.*

7 Other smoothness conditions

Relative Lipschitz condition

Jarre [9] introduced the following relative Lipschitz condition (also used in e.g. [7]) for the Hessian matrix of the problem functions $f_i(x)$, $0 \leq i \leq m$, of (CP):

$$\exists M > 0 : \ \forall v \in I\!R^n \ \ \forall x, x+h \in \mathcal{F}^0 :$$

$$|v^T(\nabla^2 f_i(x+h) - \nabla^2 f_i(x))v| \leq M\|h\|_H v^T \nabla^2 f_i(x) v, \tag{6}$$

where H is the Hessian matrix of the corresponding logarithmic barrier function. As shown in Jarre [10], if the Hessians of the problem functions f_i of $(\mathcal{P})$ fulfil this relative Lipschitz condition with parameter M, and if $f_i \in C^3$, then the associated logarithmic barrier function is $(1+M)$-self-concordant. (The converse is not true.) Moreover, in [10] it is shown that the relative Lipschitz condition for the logarithmic barrier function is equivalent to self-concordance if the underlying function is three times continously differentiable.

Monteiro and Adler's condition

Monteiro and Adler [16] considered minimization problems with linear equality constraints and a separable convex objective function on the positive orthant of $I\!R^n$. The objective function $f(x) = \sum_i g_i(x_i)$ must satisfy the following condition:

There exist positive numbers T and p such that for all reals $x > 0$ and $y > 0$ and all $i = 1, \cdots, n$, we have

$$y|g_i'''(y)| \leq T \max\left\{(\frac{x}{y})^p, (\frac{y}{x})^p\right\} g_i''(x).$$

Using Lemma 2 and substituting $y = x$ in the above condition, it is easy to see that g_i satisfies (1) with $\beta = T$, i.e. that the logarithmic barrier function for such a problem is $(1 + \frac{1}{3}T)$-self-concordant. Using Lemma 2 we may simplify the condition of [16] to the (weaker) condition that there exists a positive number T such that for all reals $y > 0$ and all $i = 1, \cdots, n$, we have

$$y|g_i'''(y)| \leq T g_i''(x).$$

This condition is not only simpler, also the dependence on some extra parameter p is eliminated.

Scaled Lipschitz condition

In [25] and [13] interior-point methods are given and analyzed for problems with linear equality constraints and convex objective function $f(x)$ on the positive orthant of $I\!R^n$. The objective function has to satisfy the following scaled Lipschitz condition:

There exists $M > 0$, such that for any γ, $0 < \gamma < 1$,

$$\|X(\nabla f(x+\Delta x) - \nabla f(x) - \nabla^2 f(x)\Delta x)\| \leq M \Delta x^T \nabla^2 f(x) \Delta x, \tag{7}$$

whenever $x > 0$ and $\|X^{-1}\Delta x\| \leq \gamma$. (Here, $\|.\|$ is the Euclidean norm.)

This condition is also covered by the self-concordance condition if f is three times continuously differentiable in the interior of the feasible domain. More precisely, the next lemma states that the corresponding logarithmic barrier function is $(1 + \frac{2}{3}M)$-self-concordant.

Lemma 7 *Suppose $f(x) \in C^3$ fulfils the scaled Lipschitz condition with parameter M. Then the logarithmic barrier functions φ and ψ from Lemma 2 are $(1 + \frac{2}{3}M)$-self-concordant.*

Before we conclude we would like to briefly point out a class of problems considered by Mehrotra and Sun [15] (and also by Zhang [24]) which does not have a self-concordant logarithmic barrier function. Mehrotra and Sun introduced a curvature constraint of the following form: There exists a number $\kappa \geq 1$ such that for all x, y and h in $\mathbb{R}^n$

$$h^T \nabla^2 f_i(x) h \leq \kappa h^T \nabla^2 f_i(y) h.$$

For constraint functions f_i satisfying this condition, they present a polynomial time interior-point algorithm (that needs at most $O(\kappa^5 \sqrt{m} \ln \epsilon)$ Newton iterations to reduce the error by a factor of ϵ). Clearly, there are constraints with self-concordant barriers that do not satisfy this condition, and conversely, this condition covers some constraint functions that do not have a self-concordant barrier function. For most applications however, we believe that the self-concordance condition is more practical.

References

[1] Alizadeh, F. (1992), Optimization over the Positive Definite Cone: Interior-Point Methods and Combinatorial Applications, in: P. Pardalos (ed.), *Advances in Optimization and Parallel Computing*, North Holland, Amsterdam, 1–25.

[2] Duffin, R.J., Peterson, E.L. and Zener, C. (1967), *Geometric Programming*, John Wiley & Sons, New York.

[3] Fiacco, A.V. and McCormick, G.P. (1968), *Nonlinear Programming, Sequential Unconstrained Minimization Techniques*, John Wiley & Sons, New York.

[4] Frisch, R. (1955), The Logarithmic Potential Method for Solving Linear Programming Problems, Memorandum, University Institute of Economics, Oslo.

[5] Han, C.-G., Pardalos, P.M. and Ye, Y. (1991), On Interior-Point Algorithms for Some Entropy Optimization Problems, Working Paper, Computer Science Department, The Pennsylvania State University, University Park, Pennsylvania.

[6] Hertog, D. den (1992), Interior Point Approach to Linear, Quadratic and Convex Programming: Algorithms and Complexity, Ph.D. Thesis, Delft University of Technology, Faculty of Technical Mathematics and Computer Science, Delft, The Netherlands.

[7] Hertog, D. den, Roos, C. and Terlaky, T. (1992), On the Classical Logarithmic Barrier Method for a Class of Smooth Convex Programming Problems, *Journal of Optimization Theory and Applications* 73, 1, 1–25.

[8] Huard, P. (1967), Resolution of Mathematical Programming with Nonlinear Constraints by the Methods of Centres, in: J. Abadie (ed.), *Nonlinear Programming*, North-Holland Publishing Company, Amsterdam, 207–219.

[9] Jarre, F. (1989), The Method of Analytic Centers for Smooth Convex Programs, Ph.D. Thesis, Institut für Angewandte Mathematik und Statistik, Universität Würzburg, Würzburg, Germany.

[10] Jarre, F. (1990), Interior-Point Methods for Convex Programming, Technical Report SOL 90-16, Systems Optimization Laboratory, Stanford University, Stanford, California. To Appear in *Applied Mathematics and Optimization*.

[11] Jarre, F. and Saunders, M.A. (1991), Practical Aspects of an Interior-Point Method for Convex Programming, Technical Report SOL 91-9, Systems Optimization Laboratory, Stanford University, Stanford, California.

[12] Kortanek, K.O. and No, H. (1990), A Second Order Affine Scaling Algorithm for the Geometric Programming Dual with Logarithmic Barrier, *Optimization* 23, 501-507.

[13] Kortanek, K.O. and Zhu, J. (1991), A Polynomial Barrier Algorithm for Linearly Constrained Convex Programming Problems, Revised Working Paper Series No. 90-17, College of Business Administration, The University of Iowa, Iowa City, Iowa. To Appear in *Mathematics of Operations Research.*

[14] Lustig, I.J., Marsten, R.E. and Shanno, D.F. (1990), On Implementing Mehrotra's Predictor-Corrector Interior Point Method for Linear Programming, Technical Report SOR 90-03, Department of Civil Engineering and Operations Research, Princeton University, Princeton, New Jersey. To Appear in *SIAM Journal on Optimization.*

[15] Mehrotra, S. and Sun, J. (1990), An Interior Point Algorithm for Solving Smooth Convex Programs Based on Newton's Method, *Contemporary Mathematics* 114, 265-284.

[16] Monteiro, R.D.C. and Adler, I. (1989), An Extension of Karmarkar Type Algorithms to a Class of Convex Separable Programming Problems with Global Rate of Convergence, *Mathematics of Operations Research*, 15, 3, 408-422.

[17] Nesterov, Y.E. and Nemirovsky, A.S. (1989), Self-Concordant Functions and Polynomial Time Methods in Convex Programming, Technical Report, Central Economical and Mathematical Institute, Academy of Science, Moscow. To Appear in *Lecture Notes in Mathematics.*

[18] Peterson, E.L. and Ecker, J.G. (1970), Geometric Programming: Duality in Quadratic Programming and l_p Approximation I, in: H.W. Kuhn and A.W. Tucker (eds.), *Proceedings of the International Symposium of Mathematical Programming*, Princeton University Press, Princeton, New Jersey.

[19] Peterson, E.L. and Ecker, J.G. (1967), Geometric Programming: Duality in Quadratic Programming and l_p Approximation II, *SIAM Journal on Applied Mathematics* 13, 317-340.

[20] Peterson, E.L. and Ecker, J.G. (1970), Geometric Programming: Duality in Quadratic Programming and l_p Approximation III, *Journal of Mathematical Analysis and Applications* 29, 365-383.

[21] Renegar, J. (1988), A Polynomial-Time Algorithm, Based on Newton's Method, for Linear Programming, *Mathematical Programming* 40, 59-93.

[22] Terlaky, T. (1985), On l_p programming, *European Journal of Operational Research* 22, 70-100.

[23] Ye, Y. and Potra, F. (1990), An Interior-Point Algorithm for Solving Entropy Optimization Problems with Globally Linear and Locally Quadratic Convergence Rate, Working Paper Series No. 90-22, Department of Management Sciences, The University of Iowa, Iowa City, Iowa. To Appear in *SIAM Journal on Optimization.*

[24] Zhang, S. (1992), The Convergence Property of Iri-Imai's Algorithm for Some Smooth Convex Programming Problems, Research Memorandum nr. 485, Institute of Economic Research, Faculty of Economics, University of Groningen, Groningen, The Netherlands.

[25] Zhu, J. (1992), A Path Following Algorithm for a Class of Convex Programming Problems, *Zeitschrift für Operations Research, Methods and Models of Operations Research*, 36, 4, 359-377.

IP, MIP AND LOGICAL MODELING USING LPL

T. Hürlimann

November 1992

INSTITUTE FOR AUTOMATION AND OPERATIONS RESEARCH

University of Fribourg

CH-1700 Fribourg / Switzerland

Bitnet: HURLIMANN@CFRUNI51

Key-words: Model Building, Modeling Language, Mixed Integer Programming.

Abstract: This paper gives an overview of different modeling techniques in the realm of integer and mixed integer programming as well as in logical modeling. The modeling language LPL (Linear Programming Language) is use as vehicle to express such models. The idea of having a unique representation scheme for mathematical and logical models is powerful but its advantages are not yet widely recognized.

Initially, LPL was built to formulate the structure of bigger LP models to overcome the model management difficulties of big real-live LP models. In the meantime, the language specification has been extended several times to manage more complex models such as logical models.

IP and MIP techniques are applicable to a surprisingly large number of (logical) problems too. Methods are given to convert these models to pure MIP models. The LPL compiler translates pure logical sentences into IP restrictions such that a linear MIP solver can solve them.

1 INTRODUCTION

"One of the reasons why IP has not been applied anywhere near as widely as it might to practical situations is the failure to recognize when a problem can be cast in this mould."
Williams H.P.

This paper supposes a reader familiar with the basics of the LPL modeling language which allows a modeler to formulate an LP-model in the usual mathematical notation using indexing mechanism as described in [Hürlimann 1992]. A brief overview of LPL can also be found in [Hürlimann 1992a].

To summarize, the main features of LPL (Linear Programming Language) are:

- a simple syntax of models with indexed expressions close to the mathematical notation, and directly applicable for documentation
- formulation of both small *and* large LP's with optional separation of the data from the model structure
- availability of a powerful index mechanism, making model structuring very flexible
- an innovative and high-level Input and Report Generator
- intermediate indexed expression evaluation (much like matrix manipulation)
- automatic or user-controlled production of row- and column-names
- tools for debugging the model (e.g. explicit equation listing)
- built-in text editor to enter the LPL model
- fast production of the MPS file
- open interface to most LP/MIP solver packages.

Recently, LPL has been enhanced by several logical operators to exploit the power of integer programming. This extension of the new version 3.9 of LPL is summarize in the second part of this abstract.
It is well known that logical statements can be translated into IP-constraints containing 0-1 variables. The paper hold at the conference will present the translation rules of logical statement into IP constraints used by the LPL compiler and expose a number of applications.

A surprisingly wide class of practical problems can be modeled using integer programming. There are applications in OR/MS including operational problems such as distribution of goods, production scheduling, machine sequencing; planning problems such as capital budgeting, facility location, portfolio analysis; design problems such as communication and transportation networks design, VLSI circuit design. There are also many applications in combinatorics, graph theory and logic. An even broader model class are mathematical models amalgamated with some logical conditions.

But the MIP formulation of a problem is sometimes far from being trivial. Ingenious techniques and a lot of modeling experience is needed to formulate such problems. Often, a more natural formulation and representation for many problems in the mentioned domains is (a subset of) predicate logic. An extension of the LPL modeling language has been designed recently by introducing various logical operators to use more natural formulation techniques. This allows the modeler to formulate his problem in a subset of predicate logic. By default, the LPL compiler translates such representation into a mixed-integer linear mathematical formulation in order to apply a general MIP solver.

Several modeling techniques in integer programming are investigated in this paper. Formulation methods are given for different problems, which can be expressed as MIP problems.

It is well known [Williams 1977], that Boolean expressions can be translated into linear, mathematical constraints such that - loosely speaking - the solution space is the same. Suppose as an example, a mathematical model containing different linear restrictions is given and the modeler wants - among other constraints - to add the logical constraint '*X or Y*' where *X* and *Y* are two propositional statements. Adding '*X or Y*' means that the model is no longer a pure mathematical model, but a mixed model of mathematical and logical constraints. To mould the whole model into a pure mathematical form, the logical statement '*X or Y*' must be replaced by the linear constraint '$x+y \geq 1$', where *x* and *y* are 0-1 variables with the meaning that they are 1 if the corresponding proposition is true and 0 otherwise. It is not difficult to see that the constraint '$x+y \geq 1$' holds if and only '*X or Y*' is true.

Although there are different methods to translate logical statements into 0-1 constraints, an mechanical translation procedure is useful, since it will allow to apply a professional MIP-solver to solve such mixed problems. Furthermore, the translation step - coded by hand - would be very time consuming and prone to errors. Hence, an automated translation procedure is especially interesting for mixed models containing symbolic

and quantitative knowledge. But even pure logical models such as the SAT-problem (satisfiability problem) can be translated into an IP model. Since these problems are all at least NP-hard, it is advantageous to approach their solutions using different methods. Furthermore, an important subset of logical problems that can be formulated as Horn clauses is *not* NP-hard. Horn clauses translated into IP programs can be solved using an LP solver, since it is sufficient to solve the LP relaxation of the IP program. The integrality constraints are automatically fulfilled. This is an interesting aspect of many (Horn)-rule based knowledge bases: Instead of using inference and resolution techniques to solve such problems, one may translate the problem into a LP problem and solve the transformed problem; large LP problems can be solved quickly.

It is relatively new to integrate a mechanical translation procedure into a modeling system. McKinnon & Williams [1990] presented a procedure and its implementation in Prolog, which accepts logical statements and outputs the corresponding linear constraints. Lucas & Mitra & Moody [1992] also expose the specification of such a converter procedure which will be integrated into the CAMPS modeling system [Lucas & Mitra 1988]. The present paper exposes another translation procedure which is already included into the LPL language.

Although such a general converter is beneficial for many problems, one should not imply that every logical model should be translated into an IP model to solve it with a general IP solver. Logical models are habitually solved more efficiently by specialized and more appropriate solvers. One should here clearly separate the formulation and the solution process. LPL is a language which allows the modeler to *formulate* his model, but the language has no general mechanism to *solve* the problem. Since it is generally admitted that a general solver for mathematical and logical models will never show up, and since many specialized and efficient solvers exist for different subset of problems, the motivation is to have at least a unique modeling language framework with which all kind of models might be *formulated* and which is formalized enough to be processed automatically by a computer. If there is no hope for a unique universal solver, it might at least be possible to have a unique language of formulation.

2 LPL EXTENSION FOR LOGICAL MODELING

Mathematical models can be represented by the LPL language in a form close to algebraic, indexed notation.

Example: the constraint

$$a_i \le \sum_{j=1}^{n} b_{ij} + \sum_{l \in S} \sum_{k=2}^{m-1} c_{i,k-1,l} x_{kl} \quad \text{for every } i \in T$$

would be formulated using the LPL syntax as

```
EQUATION R(i|T) : a[i] <= SUM(j) b[i,j] + SUM(k,l|S and k>1 and k<n) c[i,k-1,l]*x[k,l]
```

The statement contains data tables such as c_{ikl} or a_i, an indexed variables x_{kl}, several indices ($i,j,...$), arithmetic and logical operators. Several logical operators have been incorporated into the older versions of the LPL language to evaluate Boolean expressions. But Boolean expressions were not allowed in expressions containing variables as operands. A Boolean expression, such as 'S and k>1 and k<n' in the last example, can be evaluated immediately, since the operands are all 'known' quantities.

But consider now the following expression, where x is an unknown quantity

$$((x \leq 2) \text{ or } (x \geq 5)) \text{ and } (x \geq 0)$$

This Boolean expression contains a variable x and can, therefore, not be evaluated, since the value of x is unknown. If x is between zero and two or greater than 5, the expression returns true otherwise it returns false (Figure 1).



Figure 1

The second Boolean expression must be approached differently than the first expression. Nevertheless, a modeling language should accept both expressions, independently of how they are processed. In LPL, Boolean expression such as $((x \leq 2) \text{ or } (x \geq 5)) \text{ and } (x \geq 0)$ can be written in a straightforward way and can be integrated into the model as model constraint such as

```
EQUATION MyConstraint: ((x<=2) or (x>=5)) and (x>=0);
```

To process such a constraint using a standard solver , it must be translated into a pure logical or into a pure mathematical statement. Since LPL interacts well with an LP/MIP-solver, the compiler translates them by default into linear constraints, in order to apply the LP/MIP-solver. But again, there might be a more efficient solver for the problem at hand, and in this case the modeling system should translate the formulation into the appropriate form.

Table 1 summarizes all logical operators which are defined in LPL and which can be used in the formulation of a constraint. Of course, all operator can also be used in Boolean expression which are evaluated immediately as in

```
VAR X(i,j: ATLEAST(3) (i) a[i,j]);
```

The declaration is perfectly correct. It means that a variable *X* is declared for every (i,j)-tuple, such that at least three of a row i in the (known) data matrix a_{ij} are different from zero.

Note also, that the operators AND, OR, XOR, NOR, and NAND can be used as binary operators as well as index-operators. As an example, *AND{i} a[i]* simply means *a[1] and a[2] and ... and a[n]*; furthermore, *x AND* y can also be written as *AND(x,y)*. For a detailed syntax see the Reference Manual [Hürlimann 1992b]. It should also be noted that the *AND{}* has the same meaning as the old *FORALL{}* and the *OR{}* is the same as *EXIST{}*.

EXACTLY, ATLEAST, and ATMOST are index-operators with a slightly different syntax. The reserved word is followed by an expression surrounded by parentheses. The expression

```
ATMOST (4) {i} a[i];
```

means that 'at most 4 out of all a[i] should be true (=non-zero)'. Several applications that use these operators will be shown in this paper.

Operator	Alternative formulation	Interpretation
	(x and y are any sub-expression containing variables)	
unary operators		
NOT x		x is false
binary operators		
x AND y	ATLEAST(2) (x,y)	both (x and y) are true
	AND(x,y)	
x OR y	ATLEAST(1) (x,y)	at least one of x or y is true
	OR(x,y)	
x XOR y	EXACTLY(1) (x,y)	exactly one is true (either ... or)
	XOR(x,y)	
	(x OR y) AND (NOT x OR NOT Y)	
x IMPL y	NOT x OR y	x implies y (implication)
x IFF y	(x IMPL y) AND (y IMPL y)	x if and only if y (equivalence)
	NOT (x XOR y)	
x NOR y	NOT (x OR y)	none of x and y is true
	NOT x AND NOT y	
	ATMOST(0) (x,y)	(at most none is true)
x NAND y	NOT (x AND y)	at most one is true
	NOT x OR NOT y	
	ATMOST(1) (x,y)	(at least one is false)
indexed operators		
FORALL(i) x[i]	AND(i) x[i]	all x[i] are true
AND(i) x[i]	ATLEAST(#i)(i) x[i]	all x[i] are true
EXIST(i) x[i]	OR(i) x[i]	at least one out of all x[i] is true
OR(i) x[i]	ATLEAST(1)(i) x[i]	at least one out of all x[i] is true
XOR(i) x[i]	EXACTLY(1)(i) x[i]	exactly one out of all x[i] is true
NOR(i) x[i]	ATMOST(0)(i) x[i]	none of all x[i] is true
NAND(i) x[i]	ATMOST(#i-1)(i) x[i]	at least one of x[i] is false
ATLEAST(k)(i) x[i]		at least k out of all x[i] are true
ATMOST(k)(i) x[i]		at most k out of all x[i] are true
EXACTLY(k)(i) x[i]		exactly k out of all x[i] are true

Table 1: logical operators in LPL

LPL allows also to introduce predicate variables. They are simply declared as variables of type LOGICAL such as

```
VAR MyPredicate{i} LOGICAL;
```

To link the predicate with the rest of the otherwise mathematical model, an expression can be attached separated by a assign operator. Suppose that a predicate P is introduced into the model with the meaning that it is true, if another (real) variable x is strictly between the lower (l) and upper (u) bounds. The following declaration introduces the predicate and the real variable and links the predicate to the variable.

```
VAR x [l,u];          "quantity x of product i produced with lower and upper bound"
VAR P LOGICAL : x;    "product i is manufactured (true or false!)"
```

Using this declaration, one can express the logical condition $P \leftrightarrow (x \geq l)$ and $(x \leq u)$, which means that P is true if and only if x is between the lower and upper bound. If the lower bound for x is not declared, the declaration of P expresses the condition $x > 0 \rightarrow P$ (which is the same as $\overline{P} \rightarrow x = 0$); if the upper bound of x is not declared, it expresses the condition $P \rightarrow x > 0$; and if no bound for x was declared, an error is generated by the LPL compiler.

It is also possible to link a predicate to any mathematical expression, such as

```
VAR x [lx,ux]; y[ly,uy];
VAR Q LOGICAL : (x>a) or (y<b);
```

The declaration of Q imposes the logical condition $Q \rightarrow ((x > a)$ or $(y < b))$. On the other hand, if the modeler wants to impose the condition $((x > a)$ or $(y < b)) \rightarrow Q$, the expression must be preceded by the symbol '<' as in

```
VAR Q LOGICAL :< (x>a) or (y<b);
```

More explicit examples will be shown in the paper.

REFERENCES

HÜRLIMANN T. [1992], Reference Manual for the LPL Modeling Language, Version 3.8, Institute for Automation and Operations Research, Working Paper No. 191, February 1992, Fribourg.

HÜRLIMANN T. [1992a], LPL: A modeling Language, Institute for Automation and Operations Research, Working Paper No. 177, August 1990, revised April 1992, Fribourg.

LUCAS C., MITRA G., [1988], Computer-assisted mathematical programming (modeling) system: CAMPS, The Computer Journal, Vol 31(4), pp 364-375.

LUCAS C., MITRA G., MOODY S., [1992], Tools for Reformulating Logical Forms into Zero-One Mixed Integer Programs (MIPS), Working Paper.

McKINNON K.I.M., WILLIAMS H.P. [1989], Constructing Integer Programming Models by the Predicate Calculus, Annals of Operations Research, Vol 21, p.227-246.

WILLIAMS H.P. [1977], Logical problems and integer programming, Bull. Inst. Math. Appl., 13 (1977), pp.18-20.

On Vaidya's Volumetric Center Method for Convex Programming

B. Jansen C. Roos T. Terlaky
Fac. of Technical Mathematics/Computer Science
Delft University of Technology
Mekelweg 4
2628 CD Delft
The Netherlands

Interior Cutting Plane Methods

The huge amount of research on interior point methods which has been performed in the last few years also contains a number of cutting plane methods for convex programming, nondifferentiable optimization, integer programming and semi-infinite programming based on interior point ideas. The basic idea of these "interior cutting plane methods" is easy: work with a linear approximation of the feasible set, which is iteratively updated. In each iteration a certain interior point of the linear approximation is computed; if this point is feasible for the original constraints then the algorithm stops (feasibility problems) or an "objective cut" is added (optimization problems); if, on the other hand, this point is infeasible then a cutting plane is added. The methods differ e.g. in the way the specific interior points are computed for which feasibility is checked; these points will be called *centers*.

Methods that fit in the above scheme include

author(s)	ref.	center
Elzinga, Moore	[2]	ball centers
Goffin, Vial et al.	e.g. [3]	analytic centers
Vaidya	[8]	volumetric centers
Den Hertog et al.	[4,5]	minimizers log. barrier
Atkinson, Vaidya	[1]	analytic centers

A great advantage of cutting plane methods as described above is that in each iteration only linear constraints matter. Moreover, successive approximations differ only slightly, which makes "warm starts" possible. Without an efficient strategy for dropping constraints the number of constraints in the linear approximation would however grow too much.

From the practical point of view Goffin, Vial et al. have obtained very good results with their implementations, since the algorithm appears to be very stable. Theoretically the best algorithm of this form is Vaidya's [8], which uses socalled volumetric centers. The method needs a number of iterations which is bounded by $O(mL)$ where m is the dimension of the space and L somehow measures the input length of the convex program. This complexity bound can be achieved by applying a very good and natural strategy for dropping constraints; the number of constraints is bounded by $O(m)$.

Vaidya's Volumetric Center Algorithm

Preliminaries

Consider the problem of finding $y \in C \subset \mathbb{R}^m$, where C is a convex

set. A major assumption made is that the set C is contained in a ball of radius 2^L centered at the origin and that if C is nonempty then it contains a ball of radius 2^{-L}. The set C is successively approximated by bounded full-dimensional polyhedra of the form $P = \{ y \mid A^T y + s = c,\ s \geq 0 \}$, that $C \subseteq P$. Conforming to 'standard interior point notation' we denote the dimension of the space by m, whereas the number of constraints in P is denoted by n (as opposed to [8]). We denote the $i-th$ column of A by a_i.

Many interior point methods make use of the analytic center of a polytope, introduced by Sonnevend [7], which is the maximizer of

$$\max_y \{ \phi(y) = \sum_{i=1}^{n} \ln(s_i) \mid A^T y + s = c \}.$$

The hessian of $\phi(y)$ is given by $H(y) = AS^{-2}A^T$, where S denotes the diagonal matrix with the components of the vector s on its diagonal, and e is an all-one vector of appropriate dimension. The volumetric center as introduced by Vaidya is now defined as the minimizer ω of the potential function

$$F(y) = \frac{1}{2} \ln \det(H(y)).$$

To measure the distance of a point to a certain constraint the following distance measure is introduced (this measure also plays a role in methods based on analytic centers, e.g. [4]):

$$\sigma_i(y) = \frac{\sqrt{a_i^T H(y)^{-1} a_i}}{c_i - a_i^T y}.$$

It can easily be shown that $\sigma_i(y) \leq 1$ for $y \in P$. The smaller $\sigma_i(y)$ the further is the constraint from y. Moreover, it holds that $\sum_{i=1}^{n} \sigma_i(y)^2 = m$, since the σ_i's are the diagonal elements of a projection matrix. This result is a key ingredient to establish the complexity bound.

Algorithm

Let the current polytope be P^k, with volumetric center ω^k. Moreover, we have a point y^k approximating ω^k. Let ϵ be a 'small constant'.

The algorithm performs iterations of the form:

Compute $\tau := \min_i \sigma_i(y^k)^2$. Depending on the value of τ, one of two cases apply. If $\tau > \epsilon$ then a cutting plane is added, which is computed by calling an appropriate oracle. The constraint is put in a 'nice position'. On the other hand, if $\tau < \epsilon$, the constraint which is furthest away from the current center is dropped. In both cases, a Newton-type procedure is performed to find an approximation to the new center. If the resulting point is in C then stop.

Analysis

In both cases of the algorithm $O(1)$ Newton-type steps suffice. The distance measure used to measure closeness to the exact center is in fact the length of the search direction. As in the analysis of barrier methods for convex programming (e.g. Nesterov and Nemirovskii [6]), it is this measure that plays an important role in proving polynomiality.

The function $F(y)$ is used as a potential function to measure the volume of the polytope. Denote by y^* the analytic center of P. Then since

$$P \subset \{\, y \mid (y - y^*)^T H(y^*)(y - y^*) \leq n^2 \,\}$$

it holds that

$$\begin{aligned} \text{volume}(P) \leq (2n)^m (\det(H(y^*)))^{-1/2} &\leq (2n)^m (\det(H(\omega)))^{-1/2} \\ &= (2n)^m \exp(-F(\omega)). \end{aligned}$$

Vaidya has proved that after k iterations

$$F^k(\omega^k) - F^0(\omega^0) \geq \frac{k\epsilon}{2}.$$

This is enough to show that if the algorithm does not stop with a feasible point, then after $O(mL)$ iterations the volume of P must fall below 2^{-mL}, hence C is empty.

The main part of the paper [8] is a very technical analysis of the effect of adding and dropping constraints on the potential function, the position of the center and the distance measure. Also, the analysis of the Newton steps requires careful analysis. In this talk we will discuss Vaidya's method, which has very good theoretical results but seems to get little attention in the field of interior point methods. We will try to give some insight in these topics, and stress the relationship with e.g. Nesterov and Nemirovskii's [6] analysis.

References

[1] D.S. Atkinson and P.M. Vaidya, "A Cutting Plane Algorithm that Uses Analytic Centers", Working Paper, Dep. of Mathematics, University of Illinois at Urbana-Champaign, 1992.

[2] J. Elzinga and T.G. Moore, "A Central Cutting Plane Algorithm for the Convex Programming Problem", *Mathematical Programming*, 8, (1975), 134-145.

[3] J.L. Goffin and J.Ph. Vial, "Cutting Planes and Column Generation Techniques with the Projective Algorithm", *Journal of Optimization Theory and Applications*, 65, (1990), 409-429.

[4] D. den Hertog, C. Roos, and T. Terlaky, "Adding and Deleting Constraints in the Path-Following Method for LP", SHELL Report AMER.92.003, Royal/Shell-Laboratories Amsterdam, The Netherlands, 1992.

[5] D. den Hertog, J. Kaliski, C. Roos and T. Terlaky, "A Logarithmic Barrier Cutting Plane Method for Convex Programming", Working Paper, Faculty of Technical Mathematics/Computer Science, Delft University of Technology, Delft, The Netherlands, 1992.

[6] Y.E. Nesterov and A.S. Nemirovsky, "Self-Concordant Functions and Polynomial Time Methods in Convex Programming," Moscow, 1989.

[7] G. Sonnevend, "An 'analytical centre' for polyhedrons and new classes of global algorithms for linear (smooth, convex)

programming," *in: System modelling and optimization: Proceedings of the 12th IFIP-conference, Budapest, Hungary, volume 84 of Lecture Notes in Control and Information Sciences, Springer Verlag, Berlin,* (1985), 866-876.

[8] P.M. Vaidya, "A New Algorithm for Minimizing Convex Functions over Convex Sets," Working Paper, AT&T Bell Laboratories, Murray Hill, New Jersey, 1989.

AN EFFICIENT LINE-SEARCH FOR LOGARITHMIC BARRIER FUNCTIONS

Florian Jarre*

APMOD 93

Abstract

We present a new strategy for performing a line-search on a logarithmic barrier function. The strategy exploits the analyticity of the barrier function and yields a simple and efficient method for finding the minimum of the barrier function along a given line. For our theoretical investigations we define the notion of *self-concordance of order two* for the restriction f of a logarithmic barrier function to the real line. Based on this notion we define a new search step for the line search and prove a bound on the size of the new search step which is slightly better than the optimal bound known for the case of a self-concordant function (of order one). We conclude with some numerical examples that illustrate the potentials of the new line search.

1. INTRODUCTION

Interior-point methods have found wide interest in many applications in the recent past. The development of an efficient subroutine for a line search is a very important detail of the implementation of interior-point methods for solving linear or nonlinear convex programs that found little attention so far. Even though the line-search is "only" a one-dimensional problem, it is far from trivial, and our analysis below shows the rich structure that can be exploited. We devise a cheap, cubically convergent procedure that is faster than a suitably damped version of Newton's method. The underlying work is motivated by two articles by Murray and Wright [3, 4] that investigate the same problem but are completely different in their approach.

The analysis of this article is applicable to a class of problems with non-linear constraints that includes as special cases

linear and quadratic programming problems

and

minimization problems over the cone of positive semidefinite matices.

For simplicity we present our analysis for the case of a linear program, and outline briefly the modifications that are necessary for extensions to other classes of problems. Let

$$\varphi(x,\mu) = \frac{c^T x}{\mu} - \sum_{i=1}^{m} \log(b_i - a_i^T x) \tag{1.1}$$

*Institut für Angewandte Mathematik, University of Würzburg, 8700 Würzburg, (West) Germany.

be a barrier function for a linear program. (The same analysis also holds for the barrier function $\varphi(x,\mu) := c^T x/\mu - \sum \log x_i$ with the additional constraint $Ax = b$.) Let x be some strictly feasible point for φ and h be some search direction for finding the minimum of φ. Without loss of generality we assume for the rest of this article that h is a *descent direction* for φ. Define the function

$$f(t) = \varphi(x + th). \tag{1.2}$$

The function f hence depends on the barrier function φ as well as on x and h. By the assumption on h: $f'(0) < 0$.

2. KNOWN RESULTS

As shown in [5] (for a more specialized but shorter analysis see also [1]), for any x and h as above, the function f is convex and satisfies the following self-concordance relation

$$f'''(0) \le 2f''(0)^{3/2}. \tag{2.1}$$

This relation is true for functions f generated by a large class of logarithmic barrier functions including the two special cases mentioned above. (A slightly more generalized case is to replace the constant 2 in (2.1) by some other positive constant.) In the following we will always assume that f satisfies (2.1).

It is our goal to perform a line-search, i.e. to find the minimum of f for some $t > 0$. The following statements are proved in [5] for functions f satisfying (2.1):

- If the Newton step

$$\Delta t = -\frac{f'(0)}{f''(0)} \tag{2.2}$$

has H-norm less than one, i.e. if

$$\|\Delta t\|_H := |\Delta t|\sqrt{f''(0)} < 1 \tag{2.3}$$

holds, then f has a minimum. (Moreover, also the barrier function φ generating f via (1.2) has a global minimum.) Here, the notation H-norm is derived from the higher dimensional case where an analogous result holds true and H stands for the Hessian of f (respectively of φ) and $\|v\|_H = \sqrt{v^T D^2\varphi(x,\mu)v}$. We maintain this notation even though Δt is only a scalar.

- Second, if

$$\|t\|_H < 1, \quad \text{i.e. if} \quad |t| < 1/\sqrt{f''(0)} \tag{2.4}$$

then t is strictly feasible for f.

These results already supply the foundation for a simple step length rule for a line search and are also true for the higher dimensional case. A possible rule for the search step s referred to as *reduced Newton step* and for that [5] proved global convergence is the following:

$$s = \frac{\Delta t}{1 + \|\Delta t\|_H}. \tag{2.5}$$

Lemma 1
A line search using the reduced Newton step is monotonously convergent, i.e. the reduced Newton step s is "a little too short".
Proof:
We only prove monotonicity of the line-search. Let $a = f'(0) < 0$, $b = f''(0) > 0$, and $\bar{t}$ be the zero of $g(t) := f'(t)$. We show that $s \leq \bar{t}$ holds true by applying the differential inequality (2.1) (self-concordance of f) to the function $g = f'$, i.e.

$$g(0) = a, \qquad g'(0) = b, \qquad g''(t) \leq 2g'(t)^{3/2}. \tag{2.6}$$

The extremal solution v that satisfies $v'' = 2v'^{3/2}$ is an upper bound for g. (Since g' satisfies a first order initial value inequality it follows $g' \leq v'$, and hence also $g \leq v$.) The function v is given by $v(t) = (b^{-1/2} - t)^{-1} - a - b^{1/2}$ and has the (unique) zero $s = ab^{-1}/(1 + ab^{-1/2})$. By construction of v, we may conclude that $s \leq \bar{t}$. ∎

In the following we will further exploit the properties of the logarithmic barrier function and obtain additional results that are unpractical in higher dimensions but appear to be useful for a line-search.

First we illustrate with a simple example the difficulties that arise when trying to devise a heuristic for a line-search.

3. A SIMPLE EXAMPLE

Consider the case that the interval $[0, 1]$ is given by m (identical) constraints of the form $t \leq 1$ ($m \gg 1$) and the constraint $t > 0$. Even though this appears to be a very artificial example the situation arising here is typical for the problems that are encountered during a line-search. The resulting barrier function is

$$\varphi(t) = -m \log(1 - t) - \log t. \tag{3.1}$$

Clearly, the minimum of this function is at $\bar{t} = 1/m$. If we choose an initial point for Newton's method for finding $\bar{t}$ that is fairly close to $\bar{t}$, e.g. $t_0 = 1/\sqrt{m}$, we find that the Newton step starting at t_0 has length $\approx 1/2$. This means, the Newton step is by a factor $\sqrt{m}$ too long, and the resulting point is outside of the feasible set. If we choose a starting point that is further away from $\bar{t}$, e.g. $t_0 = 1/2$, we find to our surprise that the Newton step is exact. And if we move further way from $\bar{t}$ it turns out that the Newton step becomes way too short. Summarizing we see that moving from $t_0 = 1$ to $t_0 = 0$, the Newton step is first too short, then too long (much too long, by a factor of $O(\sqrt{m})$) and then again too short. Aso, the information that the H-norm of the Newton step is large does not give any information whether the Newton step is too long or too short. Thus it is hard to identify the domain in which the Newton step contains some useful information, and the domain in which bisection (and backtracking) would be the best choice to minimize f. Note further, that a seemingly "good" Newton step starting at $t_0 = 0.45$ that is only by a factor 1.1 too long still yields an infeasible point and is therefore useless. In the next section we describe a property that may help us to further analyze f and to develop new results that improve the rate of convergence.

4. SELF-CONCORDANCE OF ORDER TWO

Similarly to the definition of self-concordance, we may observe a second property of the logarithmic barrier function, namely that

$$f''''(t) \leq 6f''(t)^2. \tag{4.1}$$

This property will be referred to as *self-concordance of order 2.* The proof that this estimate is valid for the logarithmic barrier function of linear constraint functions is straightforward: Clearly, if $f(t)$ satisfies property (4.1) then so does $f(rt+s)$ with r, s constants, $r \neq 0$, i.e. the property is invariant under affine transformations. Also, if f_1 and f_2 satisfy (4.1) then so does $f_1 + f_2$, (and for $r > 0$, rf satisfies a similar property with 6 replaced by $6/r$). Since the relation is trivially true for $f(t) = \log t$ this completes the proof. Similarly we obtain for the restriction to the real line of the logarithmic barrier function of a semidefiniteness constraint

$$f(t) = -\log\det(X+tY) = -\log\det X - \log\det(I + X^{-1}Y) \tag{4.2}$$

with a positive definite matrix X and a symmetric matrix Y that

$$f^{(k)}(t) = (-1)^k(k-1)!\ \operatorname{trace}(((X+tY)^{-1}Y)^k) \tag{4.3}$$

and that

$$6f''(0)^2 = 6\operatorname{trace}(B)^2 \geq 6\operatorname{trace}(B^2) = f''''(0) \tag{4.4}$$

where $B = (X^{-1/2}YX^{-1/2})^2$ is positive semidefinite. (Note that by $\operatorname{trace}(AB) = \operatorname{trace}(BA)$ we may commute a factor $(X+tY)^{-1/2}$ to the right for our convenience.)

5. ADDITIONAL RESULTS

This section is motivated by the fact that higher order derivatives are easily computable for functions f generated by the logarithmic barrier function of linear constraints and of the determinant. If f is generated by the function φ in (1.1) as in (1.2), i.e.

$$f(t) = c^T(x+th) - \sum_{i=1}^{m} \log(b_i - a_i^T(x+th)),$$

the derivatives of f are given by

$$f'(t) = c^Th + \sum_{i=1}^{m} \frac{a_i^Th}{b_i - a_i^T(x+th)} \tag{5.1}$$

and

$$f^{(k)}(t) = (k-1)!\sum_{i=1}^{m} \frac{(a_i^Th)^k}{(b_i - a_i^T(x+th))^k} \tag{5.2}$$

for $k \geq 2$. The "expensive" part ($O(mn)$) is to compute all the scalar products a_i^Tx and a_i^Th, but once these are known, it is comparatively cheap ($O(m)$) to compute

higher order derivatives[1]. We point out that exactly the same situation holds for the logarithmic barrier function f of (4.2) with

$$f^{(k)}(t) = (-1)^k(k-1)!\sum \lambda_i^k$$

where the λ_i are all the eigenvalues of the matrix $(X+tY)^{-1}Y$.

The cheap computation of f''' allows two modifications of the line search. Rather than taking the Newton step Δt (2.2) for the line-search, (with some suitable damping factor as above) we suggest to use the zero of the second order approximation to f'. More precisely, let

$$a = f'(0) < 0, \quad b = f''(0) > 0, \quad \text{and} \quad c = f'''(0). \tag{5.3}$$

Define the discriminant $D = \max\{0, b^2 - 2ac\}$ and let

$$dt = -2a/(b + \sqrt{D}) \tag{5.4}$$

be the *cubic search step*. (It is trivial to derive that for $b^2 - 2ac \geq 0$ this value of dt is one of the zeros of the second order approximation to f', i.e. it is a critical point of the cubic approximation to f.)

Clearly, the cubic search step will be locally cubically convergent, but here, our interest is rather on how to *extend* the domain of rapid convergence compared to damped Newton's method. This goal is addressed in our second suggestion below for modification of the line search.

Before we continue, we briefly recall some known results (taken from [6]) about the Weierstraß $\wp$-function that will be used below. The Weierstraß $\wp$-function is determined by two paramters g_2 and g_3. We are only interested in the case that $\wp$ is defined by the invariants $g_2 = 0$ and $g_3 = 1$ (this is referred to as the equianharmonic case in the literature) and in real arguments for $\wp$. For this case, the description of $\wp$ involves the following constants:

$$\omega_2 = \frac{\Gamma^3(\frac{1}{3})}{4\pi} \approx 1.529954037, \quad e_2 = \frac{1}{\sqrt[3]{4}}, \quad \eta_2 = \frac{\pi}{2\omega_2\sqrt{3}} \approx 0.59276269754. \tag{5.5}$$

The function $\wp$ is periodic with period $2\omega_2$, and has poles at $t = 2n\omega_2$, $n \in \mathbf{Z}$. Further, for $t \in (0,\ 2\omega_2)$: $\wp(t) \geq e_2$. Numerically, $\wp$ can easily be evaluated by its power series expansion. We list two possible expansions for $\wp$:

$$\wp(t) = \frac{1}{t^2} + \frac{1}{28}t^4 + \frac{1}{10192}t^{10} + \frac{1}{5422144}t^{16} + O(t^{22}) \tag{5.6}$$

for $t \in (0,1)$ e.g., and

$$\wp(t+\omega_2) = e_2 + 3e_2x[1 + x + x^2 + \frac{6}{7}x^3 + \frac{5}{7}x^4 + \frac{4}{7}x^5 + O(x^6)] \tag{5.7}$$

where $x = e_2t^2$. (Both $\wp(t)$ and $\wp(\omega_2+t)$ are even functions.) The above expansions also yield the expansions for $\wp'$ and for the Weierstraß ζ-function, with $\zeta' = -\wp$.

[1]For linear constraints, it is also cheap to evaluate the function f and its derivatives for other values of t, once f is valaluated for $t = 0$.

(The constant term of ζ in (5.6) is zero, in (5.7) it is η_2.) It is known, that $\wp$ satisfies the following differential equation

$$\wp'^2 = 4\wp^3 - 1, \qquad \wp'' = 6\wp^2.$$

Finally, also the inverse functions to $\wp$ and to ζ are known by their power series expansions. Let $\Im : [e_2, \infty) \to (0, \omega_2]$ be the inverse function of $\wp$, then

$$\Im(z) = z^{-1/2}[1 + \frac{u}{7} + \frac{3u^2}{26} + \frac{5u^3}{38} + \frac{7u^4}{40} + \frac{63u^5}{248} + \frac{231u^6}{592} + \frac{429u^7}{688} + O(u^8)],$$

where $u = z^{-3}/8$ and $z \geq 0.9$. For $z \in [e_2, 0.9)$ the following expansion may be used:

$$\Im(e_2 + z) = \omega_2 - \frac{1}{3e_2}\sqrt{3z - \frac{z^2}{e_2} + \frac{z^3}{3e_2^2} - \frac{2z^4}{21e_2^3} + O(z^5)}.$$

The inverse function $\Re : \mathbb{R} \to (0, 2\omega_2)$ to ζ has the power series

$$\Re(z) = z^{-1}[1 - \frac{\gamma}{7} + \frac{17\gamma^2}{143} - \frac{496\gamma^3}{3553} + O(\gamma^4)] \tag{5.8}$$

for $z \geq 0.75$ and $\gamma = 1/20z^6$, and

$$\Re(z) = \omega_2 - \frac{z - \eta_2}{e_2}[1 - y + \frac{12y^2}{5} - \frac{267y^3}{35} + \frac{139y^4}{5} - \frac{30192y^5}{275} + \frac{1634208y^6}{3575} + O(y^7)]$$

where $y = (z - \eta_2)^2/e_2$ and $|z - \eta_2|$ is small. Finally, it holds that $\zeta(z + 2\omega_2) = \zeta(z) + 2\eta_2$ ($\wp$ is periodic but ζ is not!) and that ζ is an odd function. From this we obtain the relation $\Re(z) = 2\omega_2 - \Re(2\eta_2 - z)$ which allows us to evaluate $\Re$ for $z \leq 0.45$ e.g. via (5.8).

We now return to the problem of increasing the step length for the line search. Let $v(t) := f''(t)$. Self-concordance of order two implies that $v''(t) \leq 6v(t)^2$. We are given $v(0) = b$ and $v'(0) = c$ and intend to determine the domain of the function v for positive t as well as a lower bound of the location of the minimum of v. As in the proof of Lemma 1, the structure of this differential inequality implies that the solution of the initial value problem

$$w''(t) = 6w(t)^2, \quad w(0) = b > 0, \quad w'(0) = c \tag{5.9}$$

is an upper bound for v. The general solution of (5.9) is the $\wp$-function with invariants $g_2 = 0$ and $g_3 \in \mathbb{R}$. In our case, the solution can be reduced to the equianharmonic case $g_2 = 0$, $g_3 = 1$[2] by the transformation

$$w(t) = \alpha^2\wp(\alpha t + x)$$

with

$$\alpha = \sqrt[6]{4b^3 - c^2} \quad \text{and} \quad x = \begin{cases} \Im(b/\alpha^2) & c \leq 0 \\ 2\omega_2 - \Im(b/\alpha^2) & c > 0 \end{cases}.$$

[2]The case that $g_2 = g_3 = 0$ yields $\wp(t) = 1/(1 - t)^2$.

It is easily verified that w indeed satisfies (5.9). Since $v \leq w$, the pole of v must lie behind the one of w which gives us the following Lemma:

Lemma 2
If α and x are given as above, $t \geq 0$ and

$$\|t\|_H = t\sqrt{b} < \frac{\sqrt{b}}{\alpha}(2\omega_2 - x)$$

then t is strictly feasible[3].

More important yet, we may also derive a lower bound on the minimum $\bar{t}$ of f: Let V be an antiderivative to v e.g. $V(t) = -\alpha\zeta(\alpha(t+x))$. The value $\bar{s}$ with $V(\bar{s}) - V(0) = \int_0^{\bar{s}} v(t)\,dt = -a$ is a lower bound for $\bar{t}$ that is better than the bound derived in Lemma 1 and can easily be computed using the function $\Re$ given above. We obtain

$$\bar{s} = \alpha^{-1}(\Re(\frac{a}{\alpha} + \zeta(x)) - x), \tag{5.10}$$

and will refer to $\bar{s}$ as the *reduced cubic search step*. The construction of $\bar{s}$ implies the following lemma.

Lemma 3
The reduced cubic search step $\bar{s}$, the reduced Newton step s, and the minimum $\bar{t}$ of f satisfy the relation

$$0 < s \leq \bar{s} \leq \bar{t}.$$

Moreover, the reduced cubic search step is the maximum possible step that is less than $\bar{t}$ for given values of f, f', f'' and f''' at $t = 0$.

Unfortunately, also the reduced cubic search step $\bar{s}$ (as well as the cubic search step dt) does not grant a satisfactory rate of convergence for all starting points t_0 (e.g. not for $t_0 > 0.5$ in the example of Section 3). We therefore suggest a simple *modified cubic search method* for the first step of the line search. Set the initial search step $s = \bar{s}$ to the reduced cubic search step. Then repeatedly quadruple s as long as s is feasible for f and the absolute value of $f'(s)$ decreases. In the next section we compare the different methods.

6. SOME NUMERICAL EXAMPLES

In Table 1 we compare the reduced Newton method, the reduced cubic method, and a combined cubic method that uses the modified cubic search step (outlined at the end of the previous section) in the first iteration (to come close to the minimum), and then switches to the reduced cubic method. Each row lists the starting point t_0 and the average number of function evaluations needed by each method for 20 random examples with 5000 constraints for the interval [0, 1]. (We assume that the derivatives are cheap once f is evaluated.) In these examples 100 constraints lie in the interval $[-10, 0]$ and 4900 in the interval [1, 2]. The minima of the barrier functions lie around $t \approx 0.0003$. As final accuracy we require $\|\Delta t\|_H \leq 0.01$. (This correponds to about 10 digits relative accuracy for these examples.)

[3] For linear or semidefinite constraints this bound is merely for illustration since the exact domain is known.

t_0	red. Newton	red. cubic	comb. cubic
10^{-8}	29.2	5.1	12.2
0.1	9	8	6.6
0.999	127.7	104.7	39.2

Table 1

Since we are interested in the *global* convergence behaviour we chose starting points that all have a poor accuracy (relative to the feasible domain [0, 1]). Clearly, the last row has a very bad starting point. Below, in Table 2 we list an example with 5000 constraints and a linear objective $\frac{1}{\mu}t$ There are 2500 constraints in $[-10, 0]$ and 2500 in $[1, 11]$. The value $\bar{t}$ is the approximate minimum, and is almost constant for the 20 random examples in each row.

μ	t_0	$\bar{t}$	red. Newton	red. cubic	comb. cubic
10^{-5}	0.1	10^{-5}	12.9	11.5	8.7
10^{-5}	10^{-8}	10^{-5}	21	3.6	8.2
10^{-10}	0.1	10^{-10}	13.4	11.9	9.4
10^{-10}	10^{-14}	10^{-10}	27	3	8

Table 2

References

[1] F. Jarre, "Interior-point methods for convex programming", Report SOL 90-16, Dept. of Operations Research, Stanford University, Stanford, CA (1990), to appear in *Applied Math. and Opt.*

[2] C. Moler, J. Little, S. Bangert, "PRO-MATLAB user's guide", The Math-Works, Inc., Sherborn, MA (1987).

[3] W. Murray, M. H. Wright, "Efficient linear search algorithms for the logarithmic barrier function" Report SOL 76-18, Dept. of Operations Research, Stanford University, Stanford, CA (1976).

[4] W. Murray, M. H. Wright, "Line search procedures for the logarithmic barrier function" AT&T report 92-01, AT&T Bell Laboratories, (1992)

[5] J.E. Nesterov, A.S. Nemirovsky, "Self-concordant functions and polynomial-time methods in convex programming", Report, Central Economical and Mathematical Institute, USSR Acad. Sci., Moscow, USSR (1989).

[6] T. H. Southard, "Weierstrass elliptic and related functions" in M. Abramowitz, I. A. Stegun eds. *Handbook of math. functions* Dover Publications Inc. New York (1965) 627–671.

A Projected Conjugate Gradient Method for Sparse Minimax Problems.

Kristján Jónasson and Kaj Madsen
Institute for Numerical Analysis
The Technical University of Denmark, 2800 Lyngby, Denmark

A new method for nonlinear minimax problems is presented. The method is of the trust region type and based on sequential linear programming. It is a first order method that only uses first derivatives and does not approximate Hessians. The new method is well suited for large sparse problems as it only requires that software for sparse linear programming and a sparse symmetric positive definite equation solver are available. On each iteration a special linear/quadratic model of the function is minimized, but contrary to the usual practice in trust region methods the quadratic model is only defined on a one dimensional path from the current iterate to the boundary of the trust region. Conjugate gradients are used to define this path. One iteration solves an LP subproblem and requires three function evaluations and one gradient evaluation. Promising numerical results obtained with the method are presented. In fact we find that the number of iterations required is comparable to that of state-of-the-art quasi-Newton codes.

1. Introduction

This short paper is based on the report [4], and we shall frequently refer to this for details. Our main purpose is to present a new method for solving sparse minimax problems of the type

$$\min_{\mathbf{x}} F(\mathbf{x}) \qquad \text{where } F(\mathbf{x}) = \max_{1\le i\le m} c_i(\mathbf{x}). \tag{1.1}$$

The functions $c_i: \mathbf{R}^n \to \mathbf{R}$ are smooth.

Minimax problems occur frequently in engineering and science, and they are often both large and sparse. Such problems arise in microwave circuit design, satellite antenna design, digital filtering and optimal truss design, to name a few applications. Another important use of a minimax method is to solve constrained nonlinear programming problems, and we return to this in section 3.

We are not aware of any methods specifically designed for large and

sparse minimax problems, but several methods for dense problems exist in the litterature. A list of many of these may be found in [4]. Some of the methods are first order methods, that are based on linearizing the functions c_i and the rest are second order methods, in the sense that they are based on the Hessian of a Lagrange function. Many of the methods approximate this Hessian using quasi-Newton updates but this has the serious disadvantage that the commonly used updates quickly produce a full matrix, even for sparse problems. The approach is therefore prohibitively expensive for large sparse problems. An alternative is to use the exact Hessian or a finite difference approximation to it, and some of the methods do that. This may however also be too expensive for large problems. An additional disadvantage of the second order methods is that most of them rely on solving quadratic programming (QP) subproblems, and methods for sparse QP are not readily available.

These drawbacks are avoided by the first order methods. The use of a Hessian is avoided completely and no QP problems are solved. It is well known that for certain problems, methods based on linearization will be quadratically convergent. These problems have a strongly unique solution, which means that all directions from the solution are strictly uphill. Strongly unique solutions are necessarily such that the maximum in (1.1) is attained for at least n+1 of the functions c_i. The shortcoming of these basic first order methods is, that if the solution sought is not strongly unique, then the convergence may be quite slow.

At any point x the functions c_i that attain the maximum in (1.1) are called *active*, and the *active set*, $\mathfrak{A}(x)$, is the corresponding set of indices i. In the neighbourhood of a non-degenerate solution x^* the condition $\mathfrak{A}(x) = \mathfrak{A}(x^*)$ determines an $(n+1-t^*)$-dimensional differential manifold M, and this will be called the *active manifold*. An important fact is that the restriction of F to the active manifold is a smooth function.

The application of a basic first order descent method quickly gives a point on or near the active manifold, but convergence inside the manifold is slow, unless the problem is specially simple. There are two reasons for this slow

convergence. Firstly such a method has problems with keeping within the manifold, because a search direction found using linearization will be tangent to the manifold. The second reason is only relevant if the manifold is at least 2-dimensional. We then find that even if the manifold is linear the search direction need not be in the direction of the minimum on the manifold and a steepest-descent-like zigzagging may occur. These reasons are discussed further in [4].

In [3] a first order method that avoids the first cause of slow convergence is given. That method is of the trust region type and solves the same LP subproblems as the method given in [5] for a basic step. If the basic step goes uphill and away from the active manifold, a special corrective step back toward the manifold is tried. The method uses one gradient evaluation per iteration, and on the test problems that we tried it used an average of 1.35 function evaluations per iteration.

In the present method we attempt to avoid both causes of slow convergence. The method is still purely first order and again uses only one gradient evaluation per iteration, but now three function evaluations are needed. It will however often be the case that gradient evaluations are significantly more expensive than function evaluations, so this should not be too much of a detriment. We have used conjugate gradients to avoid the steepest descent like zigzagging inside the manifold. In addition we use a simple linesearch along an active arc to obtain the next iterate.

2. The method

In this section we give a rather informal description of the algorithm. A precise description is given in [4]. In what follows, $\mathbf{x}$ is the current iterate and ρ is the current value of a trust region radius. The gradient of c_i at $\mathbf{x}$ is a row vector denoted by $c_i'(\mathbf{x})$. The method uses sequential LP and each iteration begins with solving the following *linear minimax subproblem*

$$\begin{aligned} &\min_{\mathbf{d}_{LP}} \max_{1\le i\le m} \left(c_i(\mathbf{x}) + c_i'(\mathbf{x})\,\mathbf{d}_{LP}\right) \\ &\text{s.t. } \|\mathbf{d}_{LP}\|_\infty \le \rho\,, \end{aligned} \tag{2.1}$$

which is equivalent to an LP problem. We need a current estimate of the active set $\mathfrak{A}(\mathbf{x}^*)$ and we take this to be the active set of (2.1) at the solution and denote it by $\mathfrak{A}$. On most iterations this is the only information that is used from (2.1), but occasionally it is advantageous to take the step $\mathbf{d}_{LP}$. The current estimate of the active manifold will then be

$$M = \{\mathbf{y} \mid c_i(\mathbf{y}) = c_j(\mathbf{y}),\ \forall i,j \in \mathfrak{A}\}$$

(or, perhaps more precisely, some connected subset of this).

In trust region methods based on sequential QP, it is common that an iteration involves minimizing a quadratic model of the objective function inside the trust region. What we do is somewhat similar. On each iteration we minimize a model function, but our model function is only defined on a one-dimensional path from x. The path consists of two parts:

(1) A line segment from x to a point $\mathbf{x}_P$ on or near M.
(2) An *active arc* γ that approximately follows M from $\mathbf{x}_P$ to the boundary of the trust region.

We first determine a *projective step* $\mathbf{p}_1$, that goes approximately the shortest distance from x to M. From the requirement $\mathbf{x} + \mathbf{p}_1 \in M$ we get $c_i(\mathbf{x}+\mathbf{p}_1) = \text{constant}$, $i \in \mathfrak{A}$. First order Taylor expansion gives

$$c_i(\mathbf{x}) + c_i'(\mathbf{x})\mathbf{p}_1 = \text{constant},\ i \in \mathfrak{A} \qquad (2.2)$$

and $\mathbf{p}_1$ is defined as the minimum norm solution to (2.2). The method is designed so that on most iterations x is close to M and $\mathbf{p}_1$ is (therefore) short. From $\mathbf{x}_P := \mathbf{x} + \mathbf{p}_1$ we find a direction s which is approximately tangent to the manifold M. On the first iteration, and on iterations when a *restart* is made, s is the steepest projected descent direction. On successive iterations s is found using *conjugate projected gradients*. The linear space $T = \{\mathbf{y} \mid c_i'(\mathbf{x})\mathbf{y} = c_j'(\mathbf{x})\mathbf{y}\ \ \forall i,j \in \mathfrak{A}\}$ is approximately the tangent space to M at x. Let P be a matrix that projects onto T. For $i \in \mathfrak{A}$ the projected gradient of c_i at x is $Pc_i'(\mathbf{x})$. This turns out to be independent of i and we

define the *projected gradient* to be $\mathbf{g} = Pc_i'(\mathbf{x})$, where $i \in \mathfrak{A}$ is arbitrary. The direction $\mathbf{s}$ is $-\mathbf{g}$ when a restart is made. Otherwise,

$$\mathbf{s} = -\mathbf{g} + \beta P\mathbf{s}_{last} \quad \text{where} \quad \beta = \frac{(\mathbf{g}-P\mathbf{g}_{last})^T\mathbf{g}}{(P\mathbf{g}_{last})^T(P\mathbf{g}_{last})} \tag{2.3}$$

and $\mathbf{s}_{last}$ and $\mathbf{g}_{last}$ are the values of $\mathbf{s}$ and $\mathbf{g}$ on the previous iteration. Here we have adapted the Polak-Ribiére formula. We restart with $\mathbf{s} = -\mathbf{g}$ if one of the following occurs:

(1) the active set $\mathfrak{A}$ has changed from the last iteration,
(2) the conjugate gradient direction turns out to be uphill, (2.4)
(3) no progress was made on the last iteration.

We define $\mathbf{d}$ to be a vector in the direction of $\mathbf{s}$ with length equal to some estimate of the optimal steplength. One may for instance use the length of the step taken on the previous iteration. From $\mathbf{x}_P + \mathbf{d}$ we find a new *projective step* $\mathbf{p}_2$, toward M, in exactly the same way as $\mathbf{p}_1$ was found. Details of the linear algebra needed to obtain $\mathbf{p}_1$, $\mathbf{s}$ and $\mathbf{p}_2$ are given in [3] and [4]. The active arc γ is now defined by $\gamma(t) = t\mathbf{d}+t^2\mathbf{p}_2$. We approximate F along $\mathbf{p}_1$ with a *linear function* L and along γ with a *parabola* Q so that $F(\mathbf{x}+t\mathbf{p}_1) \approx L(t)$, $F(\mathbf{x}_P+\gamma(t)) \approx Q(t)$ and $L(1) = Q(0)$. Finally the next iterate is found by minimizing the approximation given by L and Q. If the minimum is not at $\mathbf{x}$ it is on the arc γ. We therefore let t_{min} be the minimizer of Q and choose $\mathbf{x}_P + \gamma(t_{min})$ as the new iterate, if the objective function is reduced sufficiently. If it is not possible to find a new iterate by searching along the path defined by $\mathbf{p}_1$ and γ, the last attempt on this iteration is to try the point $\mathbf{x}+\mathbf{d}_{LP}$. If this also fails to give a sufficient reduction in F then $\mathbf{x}$ is unchanged on the next iteration.

The final task of the iteration is to update the trust region radius ρ. In common with other trust region methods this is based on the quotient

$$r = \frac{\text{actual reduction in } F}{\text{predicted reduction in } F} = \frac{F(\mathbf{x}) - F(\mathbf{x}_{new})}{L(0) - Q(t_{min})}. \tag{2.5}$$

If r is close to 1 there is good agreement between the model and F, and we

increase ρ. If r is small the agreement is bad, and we decrease ρ.

The modus operandi of an iteration is illustrated in fig. 2.1 for the case n=3, t=2. We now summarize the foregoing description in an algorithm. Several details are left out, and may be found in [4].

while not stop **do**
  solve the linear subproblem (2.1) for $\mathbf{d}_{LP}$ and find $\mathfrak{A}$ and M
  find the projection $\mathbf{p}_1$ using (2.2).
  find **d** using (2.3), (2.4) and the surrounding discussion
  find the projection $\mathbf{p}_2$ using (2.2) with x replaced by $\mathbf{x} + \mathbf{p}_1 + \mathbf{d}$
  minimize the approximation Q along γ and thereby determine t_{min}
  $\delta \leftarrow \mathbf{p}_1 + \gamma(t_{min})$
  if $F(x+\delta) < F(x)$ **then** $x \leftarrow x+\delta$
    else if $F(x+d_{LP}) < F(x)$ **then** $x \leftarrow x+d_{LP}$
  update ρ based on r from (2.5)
end while

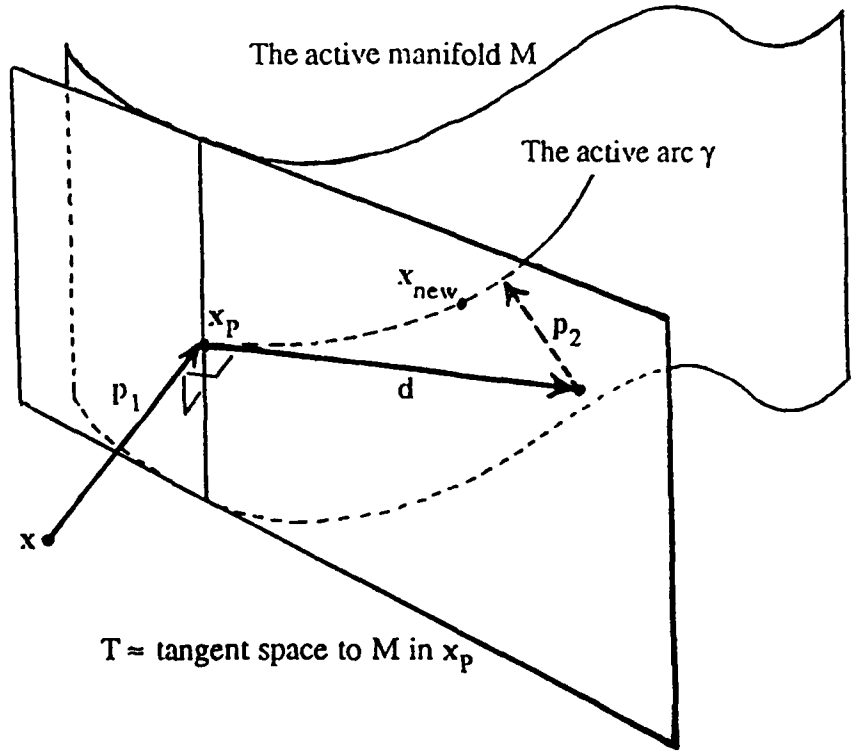


Fig. 2.1 The steps of an iteration

3. Numerical results

The new method has been applied to some standard test problems and the results are described here. For comparison some other methods have also been tried. So far we have not applied the method to very large problems. The reason is that we don't have a sparse code yet, and furthermore our philosophy is, that if the method should be able to solve large problems, it is necessary that standard small test problems are solved efficiently.

As we said in the introduction constrained optimization problems may be solved by a minimax algorithm. For $\sigma > 0$ small enough, the problem

$$\min_x \ f(x) \quad \text{s.t. } c_i(x) \le 0 \quad i=1,\ldots,m-1 \tag{3.1}$$

is equivalent to minimizing the exact penalty function $F(x) \equiv \max_{1\le i\le m}(\sigma f(x) + c_i(x))$, where $c_m(x) \equiv 0$. We believe that this is an important and often overlooked way of solving constrained optimization problems.

TEST PROBLEM	n	m	t*	PARA-METERS	FIRST ORDER METHODS: NEW METHOD	JONASSON & MADSEN	MADSEN '75	SECOND ORDER METHODS: HALD & MADSEN	WATCH-DOG	LANCE-LOT
HS100	7	5	3	σ=0.5	12	31	102	25	17	54
				σ=0.01	17	44	700	31		
				σ=0.001	30	57	7538	45		
HS113	10	9	8	σ=0.1	15	38	32	19	12	85
				σ=0.001	20	36	2166	51		
				σ=0.0001	27	55	>9999	71		
BARD	3	15	3	y=y'	7	11	5	5	8	34
	3	15	4	y=y''	7	23	5	5	8	47
KOW-OSB	4	11	4		7	9	9	14	13	
BROWN-DEN	4	20	3		12	42	37	13	11	717
ROSENB	2	2	4	w=10	8	13	28	15	5	
	2	2	4	w=100	23	17	344	15	8	
PARABOLA	2	2	2		6	21	17	8	11	26
COSH-FUN	9	3	3	n=9	8	31	36	16	12	50
	24	8	8	n=24	57	73	247	67	48	81
	42	14	14	n=42	>999	37	>9999	233	221	166
	60	20	20	n=60	>999	>999	7386	813	530	873
Function eval. pr. iteration					3.04	1.35	1	1	1.14	1
Gradient eval. pr. iteration					1	1	1	1	1.14	0.83

Table 3.1. The number of iterations to reach an accuracy of 10^{-8} in F

The test problems that we have used are described in [4]. The first two test problems are of the type (3.1) and the rest are of the type (1.1). The methods that we have used for comparison are listed below. More detailed references may be found in [4].

JÓNASSON & MADSEN The method described in [3].
MADSEN '75 The method of Madsen [5].
HALD & MADSEN The two phase method of [2].
WATCHDOG An SQP method of M.J.D. Powell (the routine MINCF of [6]).
LANCELOT The optimization package described in [1].

The results of the test runs are summarized in Table 3.1. Each entry in the table gives the number of iterations to reach an accuracy of 10^{-8} in the objective function (see [4] for details of the stopping criterion).

The test results show clearly that the new method is quite competitive with the more complicated second order methods, at least on this (small) sample of test problems. The second order methods all maintain a quasi-Newton approximation to the Hessian, and thus both storage requirements and work per iteration will be higher. We also see that the new method gives considerable improvement over the other first order methods.

References

[1] Conn, A.R., N.I.M. Gould and Ph.L. Toint, "LANCELOT: A Fortran Package for Large Scale Nonlinear Optimization (Release A)", Springer Verlag, 1992.

[2] Hald, J. and K. Madsen, *Combined LP and quasi-Newton methods for minimax optimization*, Math. Prog. **20**, 1981.

[3] Jónasson, K. and K. Madsen, *Corrected sequential linear programming for sparse minimax optimization*, Report NI-92-06, Inst. for Num. Anal., Technical University of Denmark, 1992.

[4] Jónasson, K., *A projected conjugate gradient method for sparse minimax problems*, Report NI-92-09, Inst. for Num. Anal., Technical University of Denmark, 1992.

[5] Madsen, K., *An algorithm for minimax solution of over-determined systems of non-linear equations*, J. Inst. Math. Applic. **16**, 1975, 321-328.

[6] Madsen, K., O. Tingleff, P.Chr. Hansen and W. Owczarz, *Robust subroutines for non-linear optimization*, Report NI-90-06, Inst. for Num. Anal., Technical University of Denmark, 1990.

Model Management for Stochastic Linear Programming

P. Kall and J. Mayer
IOR, University of Zurich
Moussonstrasse 15
CH-8044 Zurich

The purpose of this paper is to summarize the main model management aspects of stochastic linear programming and to outline some basic features of SLP-IOR, a model management system being under development at the Insitute for Operations Research of the University of Zurich. For model management systems in OR see Dolk [2], and for SLP-IOR see Kall and Mayer [4], [5], [6]. The discussion will be focused on single- and two-stage models these being the model-classes incorporated into the first version of SLP-IOR.

The models considered are as follows, for details see Kall [3].

Two-stage models

$$\begin{array}{lll} \min & \{c^{\mathrm{T}}x + E_\xi Q(x,\xi)\} & \\ \text{s.t.} & Ax & \propto b \\ & x & \in [l,u], \end{array} \tag{1}$$

with $\xi \in \Xi \subset \mathbb{R}^k$, $(\Xi, \mathcal{F}, P)$ is a given probability space; $\propto$ means that any one of the relations $\geq$, $\leq$ or $=$ is permitted, componentwise; and

$$\begin{array}{lllll} Q(x,\xi) = & \min & q^{\mathrm{T}}(\xi)y & & \\ & \text{s.t.} & W(\xi)y & = h(\xi) - T(\xi)x \\ & & y & \geq 0 \end{array} \tag{2}$$

where

$$\begin{cases} q(\xi) &= q^0 &+ \sum_{j=1}^{k} q^j \xi_j \\ h(\xi) &= h^0 &+ \sum_{j=1}^{k} h^j \xi_j \\ T(\xi) &= T^0 &+ \sum_{j=1}^{k} T^j \xi_j \\ W(\xi) &= W^0 &+ \sum_{j=1}^{k} W^j \xi_j \end{cases} \tag{3}$$

Problem (2) is called the second-stage (or recourse) problem; it can be interpreted as accounting for violations in the random equations $h(\xi)-T(\xi)\,x = 0$. The matrix $W(\xi)$ is called the recourse matrix of the model.

The set of equations (3) serve for modeling the way in which random variables influence the problem.

Chance-constrained (or probabilistic-constrained) models

a) Joint constraint

$$\begin{aligned} &\min Ec^{\mathrm{T}}(\xi)x \\ P(\{\xi \mid Tx \geq h(\xi)\}) &\geq \alpha \\ Ax &\propto b \\ x &\in [l,u] \end{aligned} \tag{4}$$

b) Separate constraints

$$\begin{aligned} &\min Ec^{\mathrm{T}}(\xi)x \\ P(\{\xi \mid t_i^{\mathrm{T}}(\xi)x \geq h_i(\xi)\}) &\geq \alpha_i, \ \forall i \\ Ax &\propto b \\ x &\in [l,u] \end{aligned} \tag{5}$$

In models (4) and (5) α, α_i, $\forall i$ are the prescribed probability (reliability) levels for the fulfillment of the random inequalities.

From the model management point of view the following main constituents can be identified.

Underlying algebraic structure

Neglecting randomness in the models above (e.g. by replacing all random variables by their expected value) results in deterministic LP-models which are considered as the underlying algebraic structure in the linear case.

Underlying random-variable structure

This stucture corresponds to the random variables ξ_j, j=1,...,k, and is determined by the dependency-structure and probability distributions.

Underlying regression structure

The connection of the two previous structures is established via the linear affine relations (3). The terms in the affine sums determine the way in which randomness appears in the model.

The main model-management features of stochastic linear programming can be summarized as follows.

(1) Deterministic LP is a special case of stochastic linear programming (SLP). This implies that a model management system for SLP should include or at least facilitate the use of the powerful model management tools and systems available for LP.

(2) SLP-models are hard to solve numerically: Their solution involves numerically difficult NLP-problems (including multivariate integrals) or large scale LP or mixed-integer problems. The various solution approaches substantially depend on the type of model and on the random-variable and

regression-structures. Selecting an appropriate solver is a key issue here: Solver performances for a given numerical model may differ by several orders of magnitude, including also virtually infinite solution time.

(3) Most of the solvers are designed for a determinisitic equivalent of the original SLP-model. This means that developing a model-solver interface is not merely a question of data-format, a model-transformation is also involved. This transformation depends on the model-type and on the underlying structures as well as on the specific solver.

When considering a DSS containing also an SLP-component, features (2) and (3) imply that a reliable implementation would most likely involve a full-scale MMS component for handling the difficulties outlined above.

(4) Analysis features (needed also for selecting a solver) have to account for the algebraic structure but should also include special features for SLP, e.g. to determine whether a recourse matrix is of the complete-recourse type, or whether a covariance matrix is positive definite. For this kind of model manipulation operators see also Wallace and Wets [7].

(5) The system architecture should facilitate extensions into the direction of nonlinear or multi-stage stochastic programming models.

Main features of SLP-IOR

The basic idea is to build SLP-IOR around an existing algebraic modeling system; GAMS has been chosen for this purpose, see Brooke et al. [1]. This approach provides us with an excellent tool for handling the underlying algebraic structure and also facilitates extensions for nonlinear models. GAMS serves as a uniform interface facility for solvers and supports the important issue of model documentation. Powerful general-purpose solvers are also available with GAMS. They serve for comparing solver performance.

The model manipulation operators are implemented on three levels:

- The elementary entites in SLP-IOR are the various arrays and random variables appearing in the models. Typical manipulation operators are load,

store, edit, show.

- The next level consists of the underlying algebraic-, random variable- and regression-structures. Besides the operators for elementary items some further operators like inject into a model, extract from a model, sample or discretize (for random variables) are also available.

- On the highest level are the models themselves. High level operators are e.g. solve, randomly generate, perturb, transform into another type, export/import.

Elementary items and the underlying structural elements serve as building blocks for models.

The model library contains a collection of test problems from the literature. We plan to endow SLP-IOR with a wide variety of solvers, for details see Kall and Mayer [4].

The user interactions are performed via an interactive menu-driven interface. A different approach is proposed by Gassmann and Ireland for scenario-based SLP models. They define an extension of an algebraic modeling language to account also for these SLP models.

To facilitate extension by new model-types or solvers the system is built in an object-oriented style. The model management activities , the models, the various array-types, the distributions as well as the solvers are implemented in class-hierarchies. Rules concerning the models, or the model-solver connection are implemented as polymorphic Boolean functions.

The present version of SLP-IOR is implemented for IBM/PC AT computers, endowed with an arithmetic coprocessor and having at least 8MB storage. Programming language is Turbo Pascal 6.0. To improve portability a C++ version will also be developed.

References

[1] BROOKE, A., KENDRICK, D., MEERAUS, A.: *GAMS. A User's Guide*, The Scientific Press (1988)

[2] DOLK, D.R.: *Model management systems for operations research: A prospectus*, in Mitra, G. (ed.) Mathematical Methods for Decision Support, Springer (1988) pp. 347-373

[3] KALL, P.: *Stochastic linear programming*, Springer (1976)

[4] KALL, P., MAYER, J.: *SLP-IOR: A model management system for stochastic linear programming - Sytem design*, in Beulens, A.J.M., Sebastian, H.-J. (eds.) Optimization-Based Computer-Aided Modelling and Design, Springer (1992) pp. 139-157.

[5] KALL, P., MAYER, J.: *A model management system for stochastic linear programming*, in Kall, P. (ed.) System Modelling and Optimization, Springer (1992) pp. 580-587

[6] KALL, P., MAYER, J.: *SLP-IOR: On the design of a workbench for testing SLP codes*, Manuscript, IOR, University of Zurich (1992)

[7] WALLACE, S.W., WETS, R.J-B.: *Preprocessing in stochastic programming: The case of linear programs*, ORSA Journal on Computing 4 (1992) pp. 45-59

"Two algorithms for solving linear programs with logical constraints."

Dr. W. DE KEYSER

Vrije Universiteit Brussel
Centrum voor Statistiek en Operationeel Onderzoek
Pleinlaan 2
B-1050 Brussel
BELGIUM

Extended Abstract :

Several problems such as blending problems, bundle pricing problems, logic design and circuits problems, etc. can be formulated as linear programs with logical constraints.

A logical constraint can be viewed as a constraint (or a set of constraints) that is connected with a kind of logical function that gives and indicates for which solutions this constraint (or set of constraints) must be fulfilled and for which solutions not. Two basic **logical constraints** can be distinguished :

1. Either-Or constraints : all the constraints of q (with $1 \leq q < p$) out of p subsets of constraints must be satisfied, but it is not necessary to satisfy all constraints of all p subsets. If all constraints are linear, an Either-Or constraint can be formulated as :

$$\begin{matrix}\text{EITHER-OR} \\ q^h \text{ out } p^h\end{matrix} \left(\sum_{j=1} a_{ij}^{hl} x_j \leq b_i^{hl} \quad i=1,..,m^{hl} \mid l=1,..,p^h \right)$$

2. Conditional constraints : a set of constraints must not necessarily be met by all feasible solutions; the constraints of this set must only be met by those feasible solutions that fulfil a specified condition. If all constraints involved in the condition and in the set are linear, a Conditional constraint can be formulated as :

$$\text{IF } \overset{k}{\underset{g=1}{\text{MAX}}}_{OR} \left(\overset{k}{\underset{i=1}{\text{MIN}}}_{AND} \left(\left[\sum_{j=1}^{n} \alpha_{ij}^{k} x_j \le \beta_i^k\right]\right)\right) = 1$$

$$\text{THEN } \sum_{j=1}^{n} a_{ij}^{k} x_j \le b_i^k \quad i = 1,..,m^k$$

Where :

$$\left[\sum_{i=1}^{n} \alpha_{ij}^{k} x_j \le \beta_i^k\right] = 1 \quad \text{if } \sum_{i=1}^{n} \alpha_{ij}^{k} x_j \le \beta_i^k$$

$$= 0 \quad \text{otherwise}$$

A linear programming model with **m constraints**, r **Either-Or constraints** and o **Conditional constraints** (each Conditional constraint contains m^k **constraints** and r^k **Either-Or constraints**) can be formulated as :

$$\text{Min } \sum_{j=1}^{n} c_j x_j$$

$$\sum_{j=1}^{n} a_{ij} x_j \le b_i \qquad i = 1,..,m$$

$$\text{EITHER-OR } \left(\sum_{j=1}^{n} a_{ij}^{hl} x_j \le b_i^{hl} \quad i=1,.,m^{hl} \mid l=1,.,p^h\right) \mid h=1,.,r$$
$$q^h \text{ out } p^h$$

$$\text{IF } \overset{k}{\underset{g=1}{\text{MAX}}}_{OR} \left(\overset{k}{\underset{i=1}{\text{MIN}}}_{AND} \left(\left[\sum_{j=1}^{n} \alpha_{ij}^{k} x_j \le \beta_i^k\right]\right)\right) = 1 \qquad (1)$$

$$\text{THEN } \sum_{j=1}^{n} a_{ij}^{k} x_j \le b_i^k \quad i = 1,..,m^k \qquad \Big| \; k=1,.,o$$

$$\text{EITHER-OR } \left(\sum_{j=1}^{n} a_{ij}^{khl} x_j \le b_i^{khl} \; i=1,.,m^{khl} \mid l=1,..,p^{kh}\right) \mid h=1,..,r^k$$
$$q^{kh} \text{ out } p^{kh}$$

$$x_j \ge 0 \qquad j = 1,..,n$$

Each instance of the above model for linear programming with logical constraints can be reformulated as a mixed integer program by using certain basic modelling tools. This reformulation proves the solvability of the proposed model and provides a first algorithm.

A new algorithm for solving directly the instances of the model is constructed based on following observations :

1. The region of feasible solutions consists of several separated (or adjoining) convex **subregions**.

2. The region of feasible solutions of the relaxed problem -this is the linear program without the logical constraints- is a closed **convex cover** of the subregions mentioned in observation 1 if at least one of these subregions is not empty; otherwise it is empty.

3. At least one of the optimal solutions, if there are solutions, is among the basic solutions of the convex subregions. In other words, at least one of the optimal solutions is among the basic solutions formed by <u>all</u> the linear inequalities in (1).

4. The SIMPLEX-algorithm moves from one basic solution to another neighbour basic solution.

The concept of this new algorithm is that of the cutting plane algorithms; where the objective function is used to obtain the cut-off-constraint. The algorithm can be described by the following steps :

Step 0 : initial solution.

Take as initial basic solution(s) X_l^* and as initial value of the objective function z^*, the optimal basic solution(s) and corresponding value of the objective function of the relaxed problem :

$$\text{Min} \sum_{j=1}^{n} c_j x_j$$

$$\sum_{j=1}^{n} a_{ij} x_j \leq b_i \qquad i = 1,..,m$$

$$x_j \geq 0 \qquad j = 1,..,n$$

If there is a solution for the relaxed problem then go to step 1; else stop.

Step 1 : Check feasibility.

Check if one of the X_l^* is feasible; in other words check if one of the X_l^* fulfils all the logical constraints.

If so, then X_l^* is the optimal solution and the algorithm stops.

IF not, go to step 2.

Step 2 : Reduce convex cover

Determine for each basic solution X_l^* the value of the objective function in its neighbour basic solutions that are within the feasible region of the relaxed problem. (Those basic solutions

are formed by all the linear inequalities of the LP-CCEO).
Select as new z^* the value with the smallest strict positive increase.
Add the following constraint to the relaxed problem :

$$\sum_{j=1}^{n} c_j x_j \geq z^*$$

Reoptimize and take as solution(s) X_i^* the new optimal solution(s) -if they exist- and go to step 1; if there are no solution then stop.

A first comparison between the two algorithmic solution-methods, on the criteria "Required memory-space" and "Observed computational time", will be made.

Bibliography

[1] E. Balas
"Disjunctive Programming"
Annals of Discrete Mathematics n° 5, p 3 - p 51, 1979

[2] W. De Keyser
"Linear Programming with Logical Constraints"
V.U.B. 1992

[3] W. Hanson and R.K. Martin
"Optimal Bundle Pricing"
Management Science vol. 36 n° 2, p 155 - 174, 1990

[4] R.G. Jeroslow and J.K. Lowe
"Modelling with Integer Variables"
Mathematical Programming Study 22, p 167 - 184, 1984

[5] R.G. Jeroslow
"Representability in Mixed Integer Programming, I : Characterization Results"
Discrete Applied Mathematics 17, p 223 - 243, 1987

[6] R.G. Jeroslow
"Logic-Based Decision Support. Mixed Integer Model Formulation"
Annals of Discrete Mathematics 40
North-Holland Amsterdam, 1989

[7] S. Muroga
"Logic Design and Switching Theory"
Wiley New York, 1979

[8] G.L. Nemhauser and L.A. Wolsey
"Integer and Combinatorial Optimization"
Wiley New York, 1988

[9] D.R. Plane and C. Jr. McMillan
"Discrete optimization : integer programming and network analysis for management decisions"
Prentice-Hall, 1971

[10] G.S. Thomas, J.C. Jennings and P. Abbott
"A Blending Problem using Integer Programming On-Line"
Mathematical Programming Study n° 9, p 30 - 42, 1978

[11] H.P. Williams
"Logical Problems and Integer Programming"
Bulletin of the Institute of Mathematics and its Applications, vol. 13,
p 18 - 20, 1977

[12] H.P. Williams
"Model Building in Mathemetical Programming"
A Wiley-Interscience Publication, John Wiley & Sons, Third Edition 1990

[13] W.I. Zangwill
"Nonlinear Programming, A Unified Approach"
Prentice-Hall, 1969

THE DYNAMIC THEORY OF THE INVESTING POLLUTING FIRM AND POLLUTION INSURANCE

PETER M. KORT* AND CHARLES S. TAPIERO**

* Department of Economics, Tilburg University, P.O. Box 90153, 5000 LE Tilburg, The Netherlands

** ESSEC, Department of Decision Sciences and Logistics, 95021 Cergy Pontoise, France

Extended Abstract

Consider a firm that owns a stock of capital goods K through which it can produce $Q(K)$ and it holds that $Q(0) = 0$, $Q' > 0$, $Q'' < 0$. Q can be sold on the market against a fixed price p. It is assumed that within this production process pollution is hardly controllable. Rather it is subject to uncertainties of various sorts which cannot technologically be controlled easily or would require extremely large investments. This occurs for example in situations where there is a small probability of polluting.

We assume that pollution occurs following a compound Poisson process and approximate this process by a diffusion (Wiener) process. Namely, let λ be the rate at which pollution occurs. Thus, the probability of pollution occurring in a time interval dt is λdt. Given that pollution occurs let $F(\cdot)$ be the density of the pollution size. It seems reasonable to assume that mean and variance of this size increase with production and therefore we suppose that the pollution size has mean $\alpha Q(K)$ ($\alpha > 0$ and constant), where α is the expected emissions-output ratio (cf. Dasgupta (1982), p. 20), and variance $\sigma^2\alpha^2Q^2(K)$ ($\sigma > 0$ and constant). A mean-variance diffusion normal approximation to this process is well known (see Tapiero (1984), Tapiero and Zuckermann (1982)) and is given by $\lambda\alpha Q(K)dt$ and $\lambda\sigma^2\alpha^2Q^2(K)dt$.

Therefore, if we normalize the cost per unit of pollution to one and dz is the (stochastic) pollution damage in dollars in the time interval dt, its diffusion approximation is

$$dz = \lambda\alpha Q(K)dt + \sqrt{\lambda}\sigma\alpha Q(K)db, \tag{1}$$

where db is a standard Wiener process.

The firm can partly insure itself against the uncertainty of the emission cost, but a drawback is that the premium rate includes a loading factor, which is often used by insurance firms to account for the dollar margin paid for risk protection (see Tapiero (1985)). If the part is insured is denoted by θ and δ stands for the loading factor, then the premium rate equals

$$(1+\delta)\theta\lambda\alpha Q(K) \tag{2}$$

and the emission cost arising from the part that is not insured is given by

$$(1-\theta)\lambda\alpha Q(K)dt + \sqrt{\lambda}(1-\theta)\sigma\alpha Q(K)db \tag{3}$$

In the model we suppose that θ is exogeneously determined and we will try to establish how the solution is influenced when θ increases or decreases.

Capital stock is of the non-depreciating type and can be increased by investment:

$$dK = Idt \tag{4}$$

In our quest to obtain analytical results we leave financing possibilities like borrowing and issuing new shares aside. If we let the cash process of the firm include all transactions (such as dividend distribution, investments, returns, insurance premiums, emission costs) we obtain the following state equation for the cash balance:

$$\begin{aligned} dM &= \{pQ(K) - (1-\theta)\lambda\alpha Q(K) - (1+\delta)\theta\lambda\alpha Q(K) - I - D\}dt + \\ &\quad \sqrt{\lambda}\sigma\alpha(1-\theta)Q(K)db. \end{aligned} \tag{5}$$

The firm behaves as if it maximizes the shareholders' value of the firm which can be expressed as the mathematical expectation of the discounted dividend stream over the planning period. Hence, the objective function becomes:

$$\text{maximize } E_0\left[\int_0^T D\exp(-it)dt\right] \tag{6}$$

The horizon date T is determined such, that it equals bankruptcy time. We assume that the firm is bankrupt as soon as the cash balance becomes negative, which is expressed in the following equation for the horizon date:

$$T = \inf\{t | M(t) \leq 0\} \tag{7}$$

The assumption of irreversibility of investment and the nonnegativity of the dividend rate are captured by the following inequalities:

$$D \geq 0 \tag{8}$$

$$I \geq 0 \tag{9}$$

It is assumed that the firm does not spend more on investment and dividend than the revenue net from expected pollution expenses:

$$D + I \leq pQ(K) - (1-\theta)\lambda\alpha Q(K) - (1+\delta)\theta\alpha\lambda Q(K) \tag{10}$$

To summarize: the model contains two state variables K and M, two control variables I and D, and can be expressed as follows:

$$\underset{I,D}{\text{maximize}}\ E_0\left[\int_0^T D\exp(-it)dt\right] \tag{11}$$

s.t.

$$dK = Idt, K(0) = K_0 \tag{12}$$

$$\begin{aligned} dM \;=\; & [\{p-(1-\theta)\alpha\lambda-(1+\delta)\lambda\theta\alpha\}Q(K) - I - D]\,dt + \\ & \sqrt{\lambda}\sigma\alpha(1-\theta)Q(K)db, M(0) = M_0 \end{aligned} \tag{13}$$

$$D \geq 0 \tag{14}$$

$$I \geq 0 \tag{15}$$

$$D + I \leq \{p - (1-\theta)\lambda\alpha - (1+\delta)\lambda\theta\alpha\}Q(K) \tag{16}$$

For this model to be well defined it is necessary that the right-hand side of (16) is positive. In order to make sure that this is the case for all $\theta \in [0,1]$ we introduce the following additional assumption:

$$p - \lambda\alpha(1+\delta) > 0 \tag{17}$$

(17) implies that the revenue from selling one product exceeds the premium per product to be paid when the firm is fully insured against the uncertainty of the emission cost.

In order to facilitate the notation we introduce the following symbols:

$$v = p - (1-\theta)\alpha\lambda - (1+\delta)\theta\alpha\lambda \tag{18}$$

$$\hat{\sigma} = \sigma\alpha(1-\theta)\sqrt{\lambda} \tag{19}$$

If $v = 1$ and $\hat{\sigma} = \sigma$ our model reduces to the original Bensoussan and Lesourne model with irreversible investment (see Bensoussan and Lesourne (1980)). Like in that model, also here we have three candidate policies for optimality:
Investment policy: $I = vQ(K), D = 0$;
Cash Policy: $I = D = 0$;
Dividend policy: $I = 0, D = vQ(K)$.
Given the parameter values, it completely depends on the values of the state variables M and K which of the three policies is optimal for the firm to carry out (due to the fact that the horizon time is state dependent the optimal solution does not depend on time). Therefore we divide the $M-K$ plane in three regions: investment-region (I), cash-region (M) and dividend-region (D). From the analysis of Bensoussan and Lesourne (1980) (see also Van Hilten, Kort and Van Loon (1992), Chapter 11) we obtain the most realistic solution which is depicted in Figure 1.

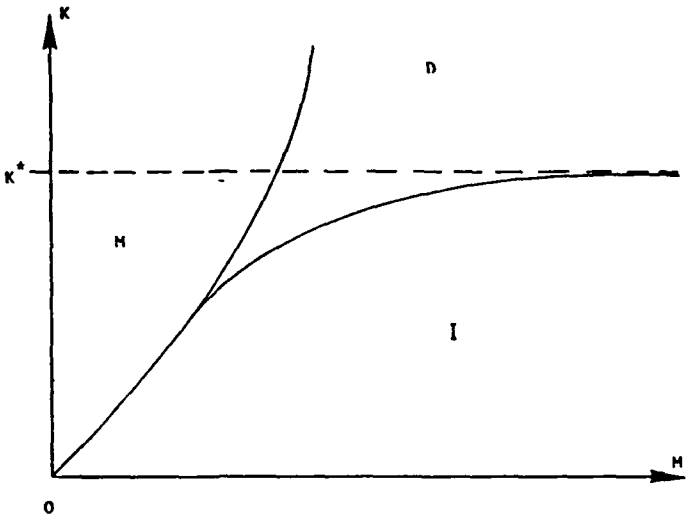


Figure 1. The most realistic solution.

The solution of Figure 1 is only optimal when a certain condition on the parameter values is met. In the paper we establish in what way this condition and the configuration of this solution is influenced when we change the values of the insurance part θ and the pollution occurrence probability λ.

References

Bensoussan, A. and J. Lesourne, 1980, Optimal growth of a self-financing firm in an uncertain environment, in: A. Bensoussan, P.R. Kleindorfer and C.S. Tapiero, eds., Applied stochastic control in econometrics and management science (North-Holland, Amsterdam), 235-269.

Dasgupta, P., 1982, The control of resources (Basil Blackwell, Oxford).

Hilten, O. van, P.M. Kort and P.J.J.M. van Loon, 1992, Dynamic policies of the firm: an optimal control approach (Springer, Berlin).

Tapiero, C.S., 1984, A mutual insurance diffusion stochastic control problem, Journal of Economic Dynamics and Control 7, 241-260.

Tapiero, C.S., 1985, A dynamic insurance firm model and dividend optimization, Large Scale Systems 9, 19-33.

Tapiero, C.S. and D. Zuckerman, 1982, Optimum excess-loss reinsurance: a dynamic framework, Stochastic Processes and their Applications 12, 85-96.

A Comparative Study of Modelling and Problem-Solving by Mathematical Programming and Logic Programming

László Béla Kovács
Department of Computer Science, University of Copenhagen
Universitetsparken 1, DK-2100 Copenhagen Ø, Denmark

Presented at Symposium on Applied Mathematical Programming, Budapest, 6-8 Januar 1993

Extended abstract

1. INTRODUCTION

In the last fifty years mathematical programming has been playing a vital role in studying the behaviour of and solving problems related to complex systems consisting of numerous interrelated objects and activities. New disciplines like operations research and management science as well as new branches of established ones had emerged to cope with the rather difficult development of general and field specific adaptations of models, methods and algorithms.

By the end of 1970s and more characteristically during the 1980s the mathematical programming based modelling and problem-solving had to face new challenges. This was due to increasing demand raised by computerization, automation and industrial development. Advances in computers and computer science provided better hardware and software tools on the one hand, but created concurrent disciplines, paradigms and system architectures for supporting human problem solving and decisions on the other. Object oriented, functional and logic programming; artificial intelligence; decision support systems, deductive databases, knowledge based and expert systems to name but a few.

Many of the mathematical programming models and algorithms are very powerful and efficient indeed, but a retrospective inspection might reveal apparent difficulties of their present applications, some of which has been observed and discussed by practitioners in many years. Two of them should serve as an illustration here. Perhaps the most severe symptom is the long development time of the basic model and reasonably efficient algorithm variant, even if standard elements are built into it. All substantial modifications and extensions require rather long time. Another inherent difficulty is the handling of approximate or missing data and an incomplete model, in the sense that certain aspects, properties, relations, types of constraints not yet known, but realized later during problem solving.

The present paper covers alternative approaches to modelling and problem solving by mathematical programming and by logic programming. The logic programming paradigm is shortly reviewed together with the principles nondeterminism and unification. Problem solving is considered as a knowledge based search in a search tree which is unfolding through the problem solving process in the extent needed to determine all the solutions. The conceptual discussion on differences and similarities in the process of development in the two disciplines will be illustrated by parallel examples. Advantages and difficulties of both approaches are discused. An attempt is made to combine the two approaches in several different ways.

2. TERMINOLOGY

The main subject of the present paper is the support of human problem solving and decisions by computer models and automatic or interactive problem solving systems. The central focus of attention is the comparison of methodologies developed around paradigms of mathematical programming and logic programming respectively. In order to get compact discussions and clear views a terminology is introduced reflecting different roles and elements, that are important in the process of building up such support systems. For example the word 'problem' has tree basically different meanings in our context. It can be the initial problem of a human decision maker, one of the mathematical problems within the computer model and a difficulty arising at any stage. These will be labelled as 'decision problem', 'problem' and 'difficulty'. In the rest of the paper the following terminology will be used:

Expert is the human decision maker who has knowledge and expertise on his field and capable of producing answers to questions, solutions to problems arising within his area, making high quality decisions of different kind. He is the one who's decisions are to be supported the way he needs it. Naturally, the word 'expert' might stand for a male/female person, a group of persons or perhaps even an entire organization. In different fields or context he might be called decision maker, manager, leader, engineer, designer, scientist etc.

Decision problem is a problem the expert is facing and wants to solve according to his knowledge and the expectations of his closer and wider community. His result or solution to the decision problem will be called *decision,* even if it is a plan, an estimate or a diagnose. Accordingly, all problem solving activities of the expert is called *decision process.*

Supporter is the person or group who carries on discussions with the expert, makes a *model* being a reflection of the real world around the decision problems which is captured in a certain extent by *problem* described within the model and builds up a *problem solver* or shortly a *solver*, capable of producing a *solution* to the problem. The process of finding the solution(s) is the *problem solving process* regardless of automatic or interactive. The solver together with a computer environment for communication, interaction, input/output, report writing, etc.will be called *support system.*

When mathematical programming is used for decision support it is combined with a certain environment, which is often models and methodology of operations research or management science. Such modelling environments for logic programming are often knowledge systems, expert systems or artificial intelligence. These categories differ from each other both in focus and in methods and techniques, but are closely related. In the further discussion, *operations research* and *knowledge systems* are chosen respectively as typical environments.

3. PROBLEM SOLVING VIA MATHEMATICAL PROGRAMMING

3.1 What is the problem?

In the first phase of modelling it is vital to understand what kind of decision problems the expert is facing and most importantly, what are the actual difficulties of the expert, influencing the quality of the decisions he is coming up with and the resources required through the decision process.

The supporter with a mathematical programming/operations research background is looking for decision problems or subproblems, that can be formulated as optimization problems perhaps through an operations research model, preferably with a good efficient algorithm. The variables, the

constraints and an objective function are identified, the latter by selecting the most important measure of quality of the decision. Other alternative measures are kept for secondary optimization or as constraints with parametric level of satisfaction. If there are measures almost equally important, then usually the weighted sum of objectives is used as an objective function. The expert is supposed to give suggestions for the weights. The expert is to be convinced, that enormous perspectives are opened into the structural examination of the problem he never have hoped for.

3.2 Modification of the model and the algorithm

The requirements, constraints, conditions and variables of the decision problem(s), that are identifiable within the model framework, but cannot be handled by the existing efficient algorithms, are reconsidered for inclusion in a modified optimization model. Linear approximation, aggregation, decomposition, simulation, relaxation are examples of such techniques. In fact, these are attempts to get a reasonably efficient algorithm for the extended problem, but in this phase algorithmic and model changes are almost inseparable.

3.3 Using real data

In the next phase the modified and usually extended algorithm should be tested against real data, using combinations of possibilities built into the support system in the course of basic development. Very often the standard source of data (e.g. regular database) is insufficient and special data collection is needed. Perhaps the expert could not do as extensive data manipulations as done by many of the support systems easily. ('The human is not a machine.') But the opposite can also be true. Some human mental activities might be difficult to follow by the machine, thus a large amount of date and very complicated algorithms may be necessary to cope with tasks easy for humans. ('The machine is not a human being.') Additionally, the algorithm may behave very differently on almost random data, than on real data. Field - or problem specific modifications of the algorithm may become advisable or even necessary to get an acceptable performance.

3.4 Testing the model and algorithm for real decision problems

There is almost never enough time for off-line testing of the support system on real decision problems, i.e. simulating the entire decision environment and full decision process for the purpose of testing the system. Usually only a few quite artificial or at least isolated test problems are created and used over and over again. Many of the requirements and demands become obvious only when the system is in regular use, leading to serious difficulties and even to rejection of the system, if it reaches this stage at all.

3.5 Forming and standardizing the support system

In the final phase the standardization of the input and output of the models and algorithms is done. The user guide describes how to use the system, including the parameters, their standard settings, other possible values and their modifying effects on the algorithms (step sizes, precisions, frequencies of numeric corrections, selection of heuristics and other strategies, etc.)

4. REFLECTIONS ON THE MATHEMATICAL PROGRAMMING APPROACH

Advantages of optimization/operations research approach

- It is supported by a vast amount theoretical research results.
- More then forty years of experience in modelling and problem solving including implementations of algorithms and code optimization.
- Many professional operations research group, management science and other groups develop and reuse models for a large variety of applications.
- There are many well understood and efficient algorithms for standard optimization problems, available also in software packages.

Difficulties with the optimization based decision process support

Dispite of all these facts, several symptoms indicate serious difficulties with introducing optimization based support systems into the real practice in cases when the application field is in a dynamically changing environment or the types of decision problems cannot be described in advance or some of the data required by the models is not available. These symptoms include unreasonably long modelling and systems development time, strong difficulties with modifying the model or the view of the problem solving. There are often claims about the flexibility of the decision support the system can provide considering starting points, priorities, special conditions, amount and kind of data requested by the support system or the variety of solutions obtained at the end of a session. Interactive decision support and problem solving seems to be particularly difficult. The decision problems should be specified in terms of the mathematical problems of the support system. Interfacing the two different way of thinking is difficult and rigid.

5. THE THREE PILLARS OF LOGIC PROGRAMMING

5.1 What is logic programming?

The basic idea of logic programming is to describe properties and relations of the objects and concepts of the area of our present interest ('the universe of discourse') in terms of predicate logic and getting answers to our questions and solutions to our problems by the help of a logic reasoning system. The questions or problems, posed to the logic program, are called goals. In other words the reasoning system, called inference engine, is supposed to find the condition under which our goal is a consequence of the logic program. Then this condition is a solution of our goal (problem). If there are several solutions, the inference engine is supposed to find all of them. The strength of logic programming is provided by the three main components called unification, non-determinism and inference engine, which are informally summarized below.

5.2 Unification

Suppose, it is given a goal or subgoal G which is to be solved. From a high level viewpoint, unification has three main functions. Firstly, to find one or more knowledge description K, that is applicable for solving G. Secondly, if the knowledge K is too general for goal G, then it should be specialized appropriately. If the knowledge K is not sufficiently general to handle/solve the goal G, then the goal should be appropriately specialized. Usually specialization of knowledge K and goal G

is done simultaneously by the unification process. It should be noted, that specializing a subproblem by a condition implies, that the rest of the problem must be specialized the same way. If the application of knowledge K to solve goal G is not successful or alternative solutions are wanted, then other applicable knowledge descriptions are used, according to the paradigm of non_determinism.

5.3 Non-determinism

Informally, non_determinism means, that alternative definitions, properties and relations of the objects as well as alternative methods, heuristics, strategies and tactics may be used in the logic program. As long as at least one combination of these alternatives can positively support the reasoning system (inference engine) finding all solutions of the stated goal is guaranteed. In the opposite case the inference engine should be able to prove, that on the basis of the knowledge provided by the logic program, there is no solution to the goal.

5.4 Inference engine

The main task of the inference engine is to show whether a stated goal is a consequence of a given logic program, representing the present knowledge about the field under consideration. It is an essential part of the task to find gradually the conditions under which the goal is provable from the logic program, because this set of conditions is considered the solution of the stated goal/problem. All alternative solutions should be determined.

5.5 Prolog as a logic programming based computer language

In sections 5.2-54 some of the basic principles of logic programming were indicated. The Prolog language inherits the declarative semantics of logic programming. Prolog has a procedural semantics as well, which property and the inference engine makes it executable on a computer. This does not mean however, that the Prolog language is procedural, though - being a general purpose computer language - one can, and occasionally does write procedures in Prolog. A more detailed explanation is beyond the scope of the present paper, but one can say that the vast majority of a good Prolog program is declarative in nature. This means, that it is not a series of instructions to the computer, but a set of definitions of objects, their properties and relationship. It is 'not doing anything' until the goal/subgoal is stated. After this, the logic is interpreted in such a way, that the goal is proven/solved by the inference engine. That activity might require very different kinds of reasoning, depending on what the goal is. Shortly, from the point of view of the run-time behaviour of the logic program, it can be considered as a condensed description of many potential algorithms, not necessarily known before the goal/problem is stated. Even the same goal/subgoal might be reasoned about in different ways, often resulting in several conceptually different solutions.

6. PROBLEM SOLVING VIA LOGIC PROGRAMMING

The logic programming methodology makes it natural to think in terms of field of interest, initial problems with gradually discovered additional properties and relations of its objects, and finally, knowledge needed to interpret the kind of problems we want to understand and solve, rather then defining a particular mathematical problem and an algorithm to solve it. Incomplete problem descriptions and incomplete information is very common in (real life) decision processes. The main

point is to advance the problem solving and the problem definition in parallel. It is usually difficult or even practically impossible to foresee all the requirements, conditions, criteria and relations of elements in the as yet unknown 'solution' to the decision situation, which are often highly dependent upon the kind of solutions we are trying to impose.

7. REFLECTIONS ON THE LOGIC PROGRAMMING APPROACH

Advantages of the logic programming approach

High problem solving power including automatic or interactive ways of

- Finding applicable knowledge within the knowledge base
- Knowledge and problem specialization
- Problem decomposition
- Using and combining alternative definitions, relations and methods
- Applying problem solving strategies and collected experience
- Conflict resolution

Problem formulation matters

- Problems need not be specified at development time
- Requirements are not restricted to have a specific mathematical form
- Incomplete problems or incomplete information can be handled
- A concept level language can be developed for problem description

Generality, flexibility and extendability of the support system is ensured by

- Declarative programming for human beings and computer interpretation
- High modularity and flexible communication between sentences, the basic program units
- Self standing sentences accepting many input/output patterns

Rapid prototyping and incremental systems development supported by

- Automatic knowledge inclusion, search and specialization
- Alternative definitions of objects, their properties and relations
- Meta-programming facilities

Concept development support, including

- Natural language interpretation facilities
- Defining new concepts in terms of relations of existing concepts
- Representing different ways of thinking and reasoning
- Flexible representation of knowledge and meta-knowledge
- Finding contradictions and resolving conflicticts

Difficulties with the logic programming approach

The traditional software development technology is often inadequate

- The analysis-planning-coding-testing cycle
- Open or hidden procedurality, including program control elements
- A concept description serving present purpose(s) only
- Precise program specifications may exclude hidden intentions, usual testing is insufficient

Keeping a high level of generality/flexibility with improving efficiency

- The ever improving efficiency should come mainly from more knowledge
- Logic level efficiency is also important, but should be kept clear and understandable
- Implementation level efficiency should be kept separated and hidden
- Limiting the potentially vast search space by meta-knowledge
- Preventing combinatorial explosion by appropriate techniques

Knowledge engineering is difficult

- Knowledge acquisition requires experience, courage and understanding
- Knowledge representation should be general and flexible
- Run-time addition of knowledge requires strong knowlegde management

Distance from traditional support systems

- Existing models, problems, algorithms cannot be directly 'plugged-in'
- Experts using more traditional approaches need time/understanding
- It is quite difficult to accept conceptual support from a computer

8. COMBINING THE TWO APPROACHES

Extensions of optimization problems and algorithms by logic programming

The mathematical programming approach might benefit from a logic programming extention and a knowledge base in the following areas:

- Natural language interface between the expert and the support system
- Man - machine interactions, 'collective' problem solving
- Problem transformations
- Pre- and postprocessing of data, requirements, criteria and solutions
- Reasoning outside the scope of the mathematical model
- Supporting formulation of problems from intentions
- Selecting and combining existing algorithms for well-defined problems
- Finding contradictions and resolving conflicts
- Using a knowledge base to improve algorithm performances
- Development and use of generic algorithms
- Generation and experimentation with various heuristics

Extensions of logic programming by mathematical programming techniques

A logic programming based decision support system might benefit from optimization techniques and models in several different ways

- Traditional models and algorithms as internal utilities
- Relaxations and estimates for pruning the search tree
- Grouping and ordering the set of solutions e.g. by lexicographic methods
- Repetition free generation of all solutions
- Sampling techniques in case of very many solutions
- Selection from alternative heuristics
- Reorganizing the search space
- Partial ordering techniques

9. CONCLUSION

The mathematical programming and the logic programming based approaches to modelling and problem solving in order to support decision processes, are quite different. Serious efforts were made for deeper integration of mathematical programming into logic programming. One of them is the definition and implementation of Prolog III, in which linear programming with a flexible application of the simplex method, is part of the language and is readily available as part of the declarative programming. Other constraint satisfaction approaches are also being used for logic programming. In constraint satisfaction the actual numerical problem solving is delayed if numeric data are not yet available or the selection is not unique. In such a way the advantages of declarative programming can be preserved. The suggestions indicated in section 8 indicate,that there are many more reasons for combining the two approaches, the work has only begun.

Modelling Tools for Reformulating Logical Forms into Zero-One Mixed Integer Programs

Bjarni Kristjansson*,Cormac Lucas, Gautam Mitra* and Shirley Moody***

* Maximal Software Ltd., Klapparas 11, 15-110 Reykjavik. Iceland

** Department of Mathematics and Statistics, Brunel University, Uxbridge, Middx. UB8 3PH, UK

Introduction

Computer based languages for constructing and analysing Mathematical Programming models have been investigated over the last two decades. There are many experimental and commercial systems currently available which provide modelling support. Most modern modelling systems enable the modeller to specify models in a declarative algebraic language. A set of algebraic statements in a modelling language both specifies and documents a model, whereas the generation of a constraint matrix takes place in the background.

Although some modelling systems have been extended to incorporate non-linearities and to help with a greater variety of discrete optimisation problems, very little attention has been given to the modelling of discrete programming extensions of LP problems. Many Mathematical Programming problems involve logical restrictions which may be expressed relatively easily using propositional calculus, but the reformulation of such statements into Mixed Integer Programs (MIPs) is conceptually difficult. This reformulation may be carried out systematically, but as yet there is no computer support for this task within a Mathematical Programming modelling system.

We present a reformulation procedure for transforming statements in propositional logic into integer or mixed integer programs; this procedure makes novel use of the Reverse Polish notation and the resulting expression tree. We define a new syntax involving logical propositions and operators whereby the structure of an LP modelling language is extended. This method is particularly suitable as a modelling technique which allows one to automate the reformulation process to construct equivalent IP or MIP models. The final goal is to integrate this modelling function into an "intelligent" mathematical programming modelling support system.

Logic Forms Represented by 0-1 Variables

The main task of reformulation is to transform a compound proposition into a system of linear constraints so that the logical equivalence of the transformed expressions is maintained.

In order to explain the reformulation process and the underlying principles more clearly, two cases are distinguished namely, connecting logical variables and logically relating linear form constraints.

Let P_j denote the $_j$th logical variable which takes values TRUE or FALSE and represents an atomic proposition describing an action, option or decision. Associate an integer variable with each type of action (or option). This variable, known as the binary **decision variable**, is denoted by "δ_j" and can take only the values 1 and 0 (binary). The connection of these variables to the propositions are defined by the following relations:

$\delta_j = 1$ iff proposition P_j is TRUE

$\delta_j = 0$ iff proposition P_j is FALSE

Imposition of logical conditions linking the different actions in a model is

achieved by expressing these conditions in the form of linear constraints connecting the associated decision variables.

Using Propositional Calculus, a list of standard form "variable transformations" are defined. These transformations are applied to compound propositions involving one or more atomic propositions P_j, whereby the compound propositions are restated in linear algebraic forms involving decision variables.

Bound Analysis/Logically Relating Linear Form Constraints

Consider the linear form restriction

$$LF_k : \sum_{j=1}^{n} a_{kj} x_j \{ \rho \} b_k$$

where ρ defines the type of mathematical relation, $\rho \in \{ \leq , \geq , = \}$. Let L_k, U_k, denote the lower and upper bounds, respectively, on the corresponding linear form, that is

$$L_k \leq \sum_{j=1}^{n} a_{kj} x_j - b_k \leq U_k .$$

Finite bounds L_k and U_k are used in the reformulation procedure. These bounds may be given or, alternatively, can be computed for finite ranges of x_j. For example, if $\ell_j \leq x_j \leq u_j$ $(j = 1,...,n)$ then

$$L_k = \sum_{j \in P_k} a_{kj}\ell_j + \sum_{j \in N_k} a_{kj}u_j - b_k \text{ and } U_k = \sum_{j \in P_k} a_{kj}u_j + \sum_{j \in N_k} a_{kj}\ell_j - b_k$$

where $P_k = \{j : a_{kj} > 0\}$ and $N_k = \{j : a_{kj} < 0\}$.

A "Logical Constraint in the Implication Form" (LCIF) is a logical combination of simple constraints and is defined as

If **antecedent** then **consequent**

where the antecedent is a logical variable and the consequent is a linear form constraint.

A "logical constraint in the general form" can be always reduced to an LCIF using standard transformations. To model the LCIF, a 0-1 indicator variable is linked to the antecedent. Whether the linear form constraint LF_k applies or otherwise is indicated by a 0-1 variable δ'_k.

$\delta'_k = 1$ iff the kth linear restriction applies

$= 0$ iff the kth linear restriction does not apply

A set of constraint transformations are defined which illustrate how this binary variable, namely the indicator variable of the antecedent, using the bound value relates to the linear form restriction, that is the consequent.

Polish Notation and Expression Trees

Using the normal precedence operators and the conventional evaluation of expressions the following logical form

$$P \vee Q \vee \sim R \wedge S$$

would be written as

$$((P \vee Q) \vee ((\sim R) \wedge S)).$$

Not using brackets as above but simply placing the operator symbols at the nodes, one can build up a tree representation using Polish notation. Choice of the directions in which the variables and symbols are scanned leads to two well known variations, namely, forward (right to left scan) or reverse (left to right scan) Polish notation. The Polish notation for an expression is not unique and within forward Polish, for instance, early-operator form or late-operator form leads to two different notations and corresponds to inserting Church's brackets from the left or from the right

respectively. The given expression can be written as

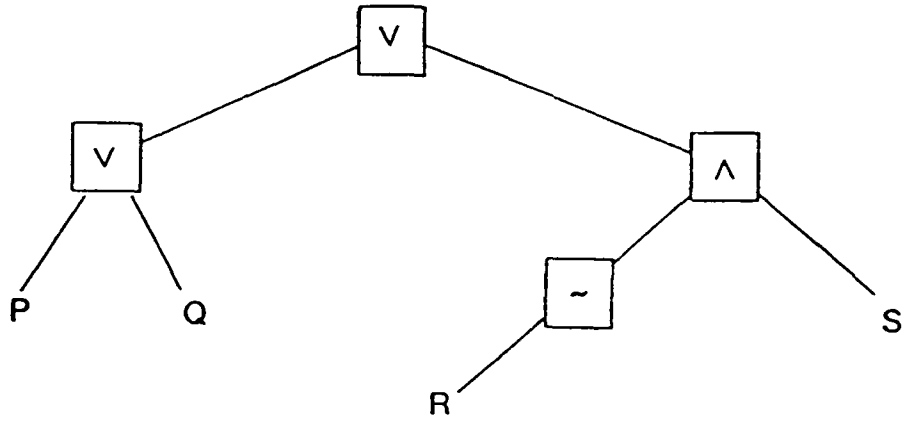
$$((P \vee Q) \vee ((\sim R) \wedge S)).$$

or

$$(P \vee (Q \vee (\sim R \wedge S))).$$

The tree representation for the first of these expression is shown below.

((P ∨ Q) ∨ (~ R ∧ S))



The Algorithm

The reformulation of a logical statement into inequalities is not unique; in fact as a result of the many, but equivalent, forms any logical statement can take, there are often different ways of generating the same or equivalent mathematical reformulations.

One possible way would be to convert the desired expression into a normal form such as the conjunction of disjunctive terms into the corresponding clauses. Each clause is then transformed into a linear constraint so that the resulting conjunctive normal form can be represented by a system of constraints which have to be satisfied invoking the logical "and" operation.

In the absence of a systematic approach, the above process appears to be unduly complicated. This has motivated us to propose a systematic

procedure to reformulate a logical condition imposed on a model into a set of integer linear constraints. Our approach, in essence, involves identifying a precise compound statement of the problem and then processing this statement. This compound statement (S) is represented as an extended expression tree by the Polish notation and two working stack mechanisms, namely VSTACK for variables and CSTACK for constraints are created. The expression tree is traversed, that is, the expression is analysed and constraints are created in CSTACK using variables which are introduced in VSTACK. The steps of the procedure which fully processes and resolves the tree are described in our presentation.

Outline of a Prototype System

The reformulation procedure thus described is illustrated by means of an example. Consider the following problem:

In order to satisfy a country's energy demands, it is possible to import coal, gas and nuclear fuel from three neighbouring countries. There are three grades of coal and gas (low, medium and high) and one grade of nuclear fuel which may be imported.

The import costs for each fuel (in £s per gigajoule of energy obtained) are provided together with upper and lower limits on the fuel supplied by each country. The problem is to decide what quantities of each fuel should be imported from each country so that the total import cost is minimized and the country's energy requirements are met.

In addition, there are the following logical conditions which must also be satisfied:

(i) Each country can supply **either** up to three non-nuclear, low or medium grade fuels **or** nuclear fuel and one high grade fuel.

(ii) Environmental regulations require that nuclear fuel can be used only if medium and low grades of gas and coal are excluded.

(iii) If gas is imported then either the amount of gas energy imported must lie between 40 - 50% and the amount of coal energy must be between 20 - 30% of the total energy imported or the quantity of gas energy must lie between 50 - 60% and coal is not imported.

The model specification for this problem including the reformulation of the logical conditions is detailed in our presentation. An extended syntax is also described which illustrates how reformulation support for logical statements may be incorporated into the modelling system MPL (Maximal Software).

Concluding Remarks

The zero-one mixed integer programming formulation of logical conditions presented as propositional calculus statements is an important research topic in discrete modelling. In our work the established syntax of representing LP models in an algebraic form is extended to incorporate logical restrictions set out as propositional calculus statements. The methods described do not necessarily achieve the most computationally efficient model after reformulation. Our main aim has been to reduce the chore for an experienced analyst, and also to provide support for a problem owner who is capable of describing his problem but may not be experienced in reformulation techniques. A system constructed in this way not only provides discrete modelling support but can also be used as a teaching aid to new modellers in MIP reformulation techniques.

References

Brearley, A. L., Mitra, G., Williams, H. P. (1975), "Analysis of mathematical programming problems prior to applying the simplex algorithm", *Mathematical Programming*, 8, 54-83.

Hadjiconstantinou, E., Lucas, C., Mitra, G. and Moody, S. (1992), "Tools for Reformulating Logical Forms into Zero-One Mixed Integer Programs", To appear in Software Tools for Modelling, *European Journal of Operational Research*

Maximal Software (1991), *MPL Modelling System Release 2 User Manual*, Maximal Software.

Williams, H. P. and McKinnon, K.I.M. (1989), "Constructing integer programming models by the predicate calculus", *Annals of Operations Research*, Vol.21, p.227-246

A B S T R A C T

Madrid and Ramsay Models for Slovak Economy.

Adam Laščiak, University of Economics,
Bratislava, Odbojárov 10.

The Madrid's Model (3) the basis of which is Pontriagin's approach on the basis of Keynesian economy type was adapted and solved for the Slovak economy with the help of NLMOS software.

Ramsay's model (4) is the mathematical and economic application for optimal repartition ofinvestment and consumption as a framework of national income growth in Slovakia. GAMS software presents special possibility to solve this variational problem with MINOS adaptation.

We have used the following adaptation of Madrid's model (proposed by Mc Fadden in 1969, see (5)) for Slovak economy.

General accounting identities.	Adaptation for Slovak economy.
$Y(k) = C(k) + S(k) + T(k)$	GNPB = -267389.3 + 0.876 GNPA + T + C
$X(k) = C(k)+I(k)+K(k)+G(k)$	X = -5738.8 + 0.497 HND + G + C + 12408.9 RIN
$B(k) = E - M(k) - K(k)$	B = 45739.5 - 0.286 GNPB + E - 1171.8 REM -12408.9 RIN
$R(k) = Y(k) - X(k)$	R = GNPB - X
$T(k) = Y(k) - C(k) - S(k)$	

Behavioral equations.

$S(k) = -a_0 + a_1 Y(k)$, $a_1 > 0$		Aggregate saving
$M(k) = -b_0 + b_1 Y(k)$, $b_1 > 0$		Imports of goods and services
$I(k) = d_0 - d_1 r(k)$, $d_1 > 0$		Domestic gross investment
$K(k) = c_0 - c_1 r(k)$, $c_1 > 0$		Net Capital outflows
$C(k) = m_0 + m_1 Y(k)$, $m_1 > 0$		Aggregate private consumption
$\bar{E}$		Exports (exogenous)

Control variables.

$$u(k) = \begin{vmatrix} dr(k) \\ dD(k) \end{vmatrix}, \text{ where } \begin{matrix} dr(k) = r(k+1) - r(k) \\ dD(k) = G(k) - T(k) \end{matrix}$$

Symbols and parameters.

$r(k)$, r_f	..... Domestic nominal interest rate or foreign interest rate
$Y(k)$ = GNPB	..... Domestic production
$X(k)$ = X	..... Aggregate expenditure
$G(k)$	..... Aggregate public consumption
$T(k)$	..... Taxes
GNPA = $Y(k) + S(k)$	..... Domestic production and saving
HND	..... Gross national income
REM	..... Change rate for USD
RIN	..... Nominal domestic interest rate
$B(k)$ = B	..... Net surplus in Payments Balance
$R(k)$ = R	..... Balance between Ressources and Expenditure

The quarterly statistical indicators were quoted from the period 198901 - 199104 and the solution of the model variables are given for the period of 199201 - 199304.

The model was controlled by taxes, public consumption and nominal interest rate. The goal of the model was maximalization of the GNPB.

The solution of the model indicates that the tax volume for Slovakia was at the limits of possibility Slovak economy, the interest rate was not significant as a parameter of control.The relation for Net surplus in balance of payments can be solved only as an approximate goal.

All statistical indicators, derivated parameters, given relations of model were tested on the basis of the SORITEC software. The model was calculated also for the given periods 198901-199104. In this case it is possible to use the Optimal Control Theory results to calculate the control that minimizes the deviations from the desired control trajectory and to make an analysis of the past period.

The basic core of the model may be described in the following form:

$$\begin{pmatrix} x(k+1) \\ y(k+1) \end{pmatrix} = \begin{pmatrix} I & 0 \\ 0 & I \end{pmatrix} \begin{pmatrix} x(k) \\ y(k) \end{pmatrix} + \begin{pmatrix} B \\ 0 \end{pmatrix} u(k)$$

$$y(k) = \begin{pmatrix} 0 & I \end{pmatrix} \begin{pmatrix} x(k) \\ y(k) \end{pmatrix}$$

$u(k) = \begin{pmatrix} dr(k) \\ dD(k) \end{pmatrix}$ is the control vector, where $dr(k) = r(k+1) - r(k)$ is a monetary measure and $dD(k) = D(k+1) - D(k)$,

D(k) being the domestic deficit and D(k) = G(k) - T(k) is a fiscal measure.

$$x(k) = \begin{vmatrix} B(k) \\ y(k) \end{vmatrix}$$ is the state vector.

$$B = \begin{vmatrix} c_1 + n\, b_1 d_1 - n\, b_1 & \\ -n\, d_1 & n \end{vmatrix}$$ is input distribution matrix with $n = 1/(a_1 + b_2)$

If we are interesteds in the open loop control of the system (see detail in (3)):

$$\begin{vmatrix} B(k+1) \\ y(k+1) \end{vmatrix} = I \begin{vmatrix} B(k) \\ y(k) \end{vmatrix} + B \begin{vmatrix} dr_f(k) \\ dD(k) \end{vmatrix}$$ and if

$$x_0(k) = \begin{vmatrix} B(0)(1 - e_B)^k \\ y(0)(1 + e_Y)^k \end{vmatrix}$$ is the desired trajectory and $(0 \leq e_B < 1)$ $(0 \leq e_Y)$,

the open loop control should have the form:

$$u_0(k) = B^{-1} \begin{vmatrix} -B(0)\, e_B (1 - e_B)^k & 0 \\ 0 & y(0)\, e_Y (1 + e_Y)^k \end{vmatrix}$$

and transforming $x_0(k)$ into dynamic equilibrium path. But for the Slovak economy case it is possible only with some approximation and deviation from the goals. The dynamic equilibrium path for Slovak economy in the next future is very hard to satisfy.

The Ramsay model ((4),(6)) modification propsed for Slovak economy has the following form set:

$$\text{MAX} \quad U = \sum_{t=t_1}^{t_r} (b_t \log C_t)$$

$$A(t)K_t = C_t + I_t \quad - \text{ National receive}$$

$$I_t \leq A(1 + a_c)^t \quad - \text{ Gross investment}$$

$$A = Y_0/(K_0)^e \quad - \text{ is the initial condition}$$

$$A(t) = A(1 + w)^{(1-e)t}$$

$$Y_t = C_t + I_t \quad - \text{ "National income"}$$

$$K_{t+1} = K_t + I_t \quad - \text{ Capital for the producing sphere}$$

$$w K_t \leq I_t \quad - \text{" w" is labour growth rate}$$

$$T \qquad T$$

and the initial conditions: $C_t \geq C_0$, $I_t \geq I_0$, $K_t \geq K_0$.

The another symbols: U - Utility function

C_t - Aggregate consumption (for all inhabitans)

b_t - Discount factor (necessary to make the parametrization in the steps of the optimalisation)

e - Elasticy

w - Labour growth rate

a_c - The mesure for labour capacity absorbtion: 1/(K/L), L - labour capacity.

$Y(t) = F(\exp(r))(dk^{e}L^{(1-e)})$ is the production function as a checking function for National receive (inhabitans and enterprises) when the problem of taxes is outside the Ramsay's model. (For the Taxes problem see the Madrid model.)

All parameters and indicators were derivated by SORITEC software. The adaptation of GAMS software with MINOS algorithm was successful.

Optimal solution of repartition of "National income" for Slovak economy was the following:

(Mld Kcs)	NI	Consumption	Investment	Capital stock
1990	163.6	107.0	56.6	1163.6
1995	169.8	112.9	56.8	1446.6
2000	175.8	119.2	56.8	1741.2
(i)				
1990	185.9	123.1	62.8	1163.6
1995	193.3	130.5	62.8	1477.9
2000	200.5	137.7	62.8	1804.9

(i) The first table has the statistical classification according the old calculatiom for Material product. The second - in accordance with the Gross Domestic Product.

The Madrid and Ramsay models are only one part of the system models which are described as succesive approch to market mechanism in Slovak economy. The key model for the period of trasition is MODELEP - the model on the basis of linear and nonlinear econometric equations (set of 75 equations) which is outside of this paper.

The possibility of economic growth is described in the pe-

riod 1989Q1 - 1991Q4 also by means of extensive and intensive factors which have been developed into a model through dynamization of different production functions on the Hamilton's principle of the least change of the economic state. For the mentioned period we notice a decrease of the total economical growth dynamics as well as the influence share of the intensive factors.

These mentioned mostly long-term tendencies follow also from the results of using other analytical tools as the IS - LM model. But with application of the IS - LM model were some problems because at present we are in a period of transition from central to market economy. Moreover it is very difficult to obtain and to precise the indicators of money - market now. For these reasons it is very difficult to achieve one equilibrium among the goods and sevices and the money - market.

For the keynesian or classical monetary approach in our modelling the way to market mechamism there are following reasons: The difference lies in the assumption of classical macroeconomics that, prices and wages are flexible - trussting that after an economic shock, the price flexibilty can restore full employment very fast. That's not the case for eastern countries, neither it is Slovakia. In fact, the keynesian revolution combined two elements. First, it is the concept of agragate demand, in which agragate spending would be driven by the consumption function and by investment decisions. Second, the price flexibility in the transition period is only in a one-way one and wages are under consensus of discrete inflexibility or sticky. (See Paul Samuleson - William Nordhaus in (7).

References:(1) A. Laščiak and oth."Dynamic Models", ALFA, 1985, Bratislava.

(2) J. Sojka:"Economic Equilibrium and Growth",1970, EPOCHA, Bratislava.

(3) B. Rosa, P. Gomez. J. Patricio: "State Space Control of Dynamic Linear Economic System".Madrid, UNIVERSIDAT, 1988.

(4) "Investment,Equilibrium,Optimal Growth".PRAVDA, Bratislava, 1970.

(5) Aoki, M.: "Optimal Control and Systems Theory in Dynamic Economic Analysis", McGraw-Hill Co.,N.Y. 1980.

(6) Ramsay, Frank P.:"A Mathematical Model of Saving." Economic Journal, Dec. 1928, pp. 543-59.

(7) Samuelson, Paul A. and Nordhaus, William D.: "ECONOMICS", XIII edition, McGraw-Hill Co. N,Y. 1989, p. 161.

Experimental Investigations in combining IPM and Simplex Based LP Solvers

Ron Levkovitz
Gautam Mitra
Brunel, the university of west London, UK
Mehrdad Tamiz
University of Portsmouth UK

Abstract

The use of interior point method (IPM) based optimizer as a robust linear programming (LP) solver is now well established. The default parameter settings of new generation of IPM solvers are sufficient to process a wide class of LP problems whereas traditional sparse simplex (SSX) based solvers may require a considerable adaptation and tuning from one model class to another. The progress of IPM iterations is not hindered by the degeneracy or the stalling problem of SSX, indeed it reaches the 'near optimum' solution very quickly. The SSX algorithm, in contrast, is not affected by the boundary conditions which slow down the convergence of IPM.

The extreme point solution is the corner stone of the SSX algorithm and use of the corresponding optimal LP basis as a starting point for solving integer programming problems or post optimal analysis amongst others is well known. The IPM algorithms usually converge to a point in the interior of the optimal face (non-extreme point) and their poor behaviour near the boundary make it difficult to apply IPM in the same way as the SSX.

To take advantage of the attractive properties of IPM and SSX, we have designed a hybrid framework whereby cross over from IPM to SSX can take place at any stage of the IPM optimisation run. The cross over to SSX, at a non optimal solution, involves the prediction of the optimal face. Our prediction incorporates several methods suggested by us and other researchers in this area.

We review a number of cross over methods and test their suitability for optimal and intermediate cross over. Some computational results on a set of degenerate and non degenerate test problems are given.

1. Introduction

The use of IPM for the solution of linear programs provides a number of benefits which are summarized below. For large or highly degenerate LPs IPM is usually faster than the SSX solver. Whereas SSX based algorithms require considerable adaptation and control parameter tuning from one model class to another, default settings of IPM are sufficient to process a wide class of LPs. IPM is not only robust in this way, its progress is not hindered by the degeneracy or the stalling problem of the SSX; indeed it reaches the "near optimal" solution very quickly. SSX algorithms, in contrast, are not affected by the boundary conditions which slow down the convergence of IPM.

There are three well known and well exercised extensions of traditional LP namely, successive linear programming, integer programming, and post-optimal analysis. In all these cases optimum solutions for a family of problems need to be computed which in turn involves reoptimization from the last computed primal and dual optimum basis (extreme point solution). The extreme solution is the corner stone of SSX algorithms and the use of the corresponding basis as starting point is naturally applied in this context to solve efficiently a family of similar problems. IPM algorithms, on the other hand, usually converge to a point in the interior of the optimal face. This property and the behaviour of the IPM algorithm near the boundary (Megiddo 89) make it difficult to apply IPM in the same way for a family of similar LP's. Some researchers have attempted to develop new theory and methods that do not depend on the extreme point representation (Adler 89) (Guler 92). Alternatively, the fast initial convergence of IPM to a near optimal solution can be followed up by the superior near optimal to optimal convergence of SSX algorithms. This strategy is not only computationally attractive it also provides the (currently) desirable extreme point solution and the corresponding primal and dual optimum basis. Most researchers (see for example (Megiddo 88,91), (Bixby 92), (Mitra 88)) consider this latter approach to be a promising computational scenario. This hybrid approach, however, requires a substantial performance superiority of IPM and an efficient IPM-SSX integration to make it worthwhile.

The research issues reported in this paper are mainly concerned with extending IPM whereby it can be brought into the mainstream of solving large LP's. After this introduction, in section 2 we review and extend methods for predicting the variables that are active in the IPM solution before optimality is reached. In section 3 we describe methods for converting

the IPM optimal or predicted solution to an extreme point solution. In Section 4 we present experiments in early and optimal cross over from IPM to SSX using alternative prediction criteria. Our conclusions are summarized in section 5.

2. Prediction of variables in an optimal solution

Consider the following primal and dual LP problems:

$$\begin{aligned} &(\textit{Primal})\ \textit{Min}\ c^T x \quad \textit{S.T.}\ Ax=b,\ x\geq 0 \\ &(\textit{Dual}) \quad \textit{Max}\ b^T y \quad \textit{S.T.}\ A^T y+z=c,\ z\geq 0 \\ &\qquad A\in\mathbf{R}^{m\times n},\ x,z,c\in\mathbf{R}^n,\ y,b\in\mathbf{R}^m \end{aligned} \tag{2.1}$$

An interior point algorithm applied to these problems generates a sequence of strictly interior points and theoretically converges to an optimal solution which is a boundary point of the feasible polyhedron. The actual termination of the algorithm, however, is not on the boundary of the polyhedron but in the interior (Levkovitz 92) close to an optimal solution. At the optimal solution, strict complementarity is enforced, that is :

$$c^T x - b^T y = x^T z = 0 \quad \textit{and} \quad XZe=0 \tag{2.2}$$

We define the *indicator sets* of *active* (positive) and *dormant* (non positive) indices $\sigma(v)$, $\bar{\sigma}(v)$ of a non negative vector v as:

$$v\in\mathbf{R}^n,\ v\geq 0\ ,\ \sigma(v)=\{i,v_i>0\},\ \ \bar{\sigma}(v)=\{1,..,n\}-\sigma(v) \tag{2.3}$$

Of all the primal optimal solutions and the dual optimal solutions of (2.1) there exist at least one solution pair (x^*), (y^*,z^*) where the strict complementarity of (2.2) applies, or in terms of the indicator set σ defined in (2.3):

$$(i)\ \sigma(x^*)\cap\sigma(z^*)=\phi \ \textit{and}\ (ii)\ \sigma(x^*)\cup\sigma(z^*)=\{1,..,n\}$$

(2.4)

We call this solution the strict complementary solution. We note that the second condition of (2.4) does not hold if in both the primal and the dual problems are degenerate. This case is considered in section 3.

Guler and Ye (91) show that a class of interior point methods which also contains the primal dual predictor corrector algorithm (Mehrotra 90) generates a sequence of feasible pairs

the IPM optimal or predicted solution to an extreme point solution. In Section 4 we present experiments in early and optimal cross over from IPM to SSX using alternative prediction criteria. Our conclusions are summarized in section 5.

2. Prediction of variables in an optimal solution

Consider the following primal and dual LP problems:

$$\begin{aligned} &(Primal)\ Min\ c^T x \quad S.T.\ Ax=b,\ x\geq 0 \\ &(Dual)\quad Max\ b^T y \quad S.T.\ A^T y+z=c,\ z\geq 0 \\ &\qquad A\in\mathbf{R}^{m\times n},\ x,z,c\in\mathbf{R}^n,\ y,b\in\mathbf{R}^m \end{aligned} \tag{2.1}$$

An interior point algorithm applied to these problems generates a sequence of strictly interior points and theoretically converges to an optimal solution which is a boundary point of the feasible polyhedron. The actual termination of the algorithm, however, is not on the boundary of the polyhedron but in the interior (Levkovitz 92) close to an optimal solution. At the optimal solution, strict complementarity is enforced, that is :

$$c^T x - b^T y = x^T z = 0 \ \text{ and } \ XZe=0 \tag{2.2}$$

We define the *indicator sets* of *active* (positive) and *dormant* (non positive) indices $\sigma(v)$, $\bar{\sigma}(v)$ of a non negative vector v as:

$$v\in\mathbf{R}^n,\ v\geq 0\ ,\ \sigma(v)=\{i,v_i>0\},\ \ \bar{\sigma}(v)=\{1,..,n\}-\sigma(v) \tag{2.3}$$

Of all the primal optimal solutions and the dual optimal solutions of (2.1) there exist at least one solution pair (x^*), (y^*,z^*) where the strict complementarity of (2.2) applies, or in terms of the indicator set σ defined in (2.3):

$$(i)\ \sigma(x^*)\cap\sigma(z^*)=\phi \ \text{ and } \ (ii)\ \sigma(x^*)\cup\sigma(z^*)=\{1,..,n\} \tag{2.4}$$

We call this solution the strict complementary solution. We note that the second condition of (2.4) does not hold if in both the primal and the dual problems are degenerate. This case is considered in section 3.

Guler and Ye (91) show that a class of interior point methods which also contains the primal dual predictor corrector algorithm (Mehrotra 90) generates a sequence of feasible pairs

(x^k, z^k) such that :

$$\frac{\min(X^k Z^k e)}{x^{k^T} z^k} > \Omega(\frac{1}{n}) \tag{2.5}$$

For this class of algorithms, they also proved the following theorem:

Theorem 2.1: At iteration k, let $\sigma^k=\{j:x_j^k \geq z_j^k\}$ *and assume the LP data is rational. If L is the input length of the problem then for all algorithms that satisfy (2.5) if* $(x^k)^T z^k \leq 2^{-3L}$, $\sigma^k=\sigma^*$.

Theorem 2.1 shows that when an interior point is close enough to the optimal solution, the set σ^* of active variable indices can be identified. The dormant variables can be set to zero and a smaller problem can be solved to retrieve the 'exact' optimal solution on a boundary of the polyhedron.

Although theorem 2.1 gives a theoretical stopping criterion for the primal dual IPM, in practice, a more realistic criterion is needed. Mehrotra (91-2) proves that whenever the set σ^* can be identified, the algorithm can terminate the normal execution and perform the following computation to retrieve the primal and dual optimal solutions:

$$\begin{aligned} &(\textit{Primal})\ A_{\sigma^*}.\Delta x_{\sigma^*} = b - A_{\sigma^*} x_{\sigma^*}^k \\ &(\textit{Dual})\ A_{\sigma^*}^T.\Delta y = c_{\sigma^*} - A_{\sigma^*}^T y^k = z_{\sigma^*}^k \end{aligned} \tag{2.6}$$

These equations can be solved in a single 'IPM-like' iteration.

There are some obvious advantages in terminating IPM before reaching the close neighbourhood of the optimal solution and by applying a procedure similar to (2.6). Some of these advantages are listed below:

(i) at an early stage of the IPM search procedure, namely 50% of the total number of iterations, IPM finds a near optimum solution which is often within 80-90% of the final solution.

(ii) the numerical stability of the algorithm deteriorates when the algorithm gets close to the

boundary, the computation of the trajectory in the k^{th} iteration of IPMABC includes the computation of the diagonal matrix $D^k=X^k(Z^k)^{-1}$. The diagonal of this matrix can have some very high value entries for variables that participate in the solution and some near zero entries for variables that converge to their lower bound. In such a case, the matrix AD^kA^T can become ill conditioned and cause numerical errors.

(iii) As active or dormant variables are identified, the size of the problem can be reduced by removing them. In particular, the removal of dormant variables maintains primal feasibility and reduces the computation work of every iteration. Also, the removal of primal variables that converge to zero increases the numerical stability of the calculation.

As a result, there is considerable interest in identifying the active and dormant variables by heuristic procedures at an intermediate stage of the IPM search. Gay (88), Mehrotra (91-2), Zhang et al. (91), El-Barki et al. (91), Levkovitz (91) and others have put forward heuristics and report encouraging results. These heuristics are based on indicator functions which are calculated at every iteration. El-Barki et al. in their survey of indicator functions classifies them in several types namely variables that are used as indicators, primal dual indicators and Tapia's indicators which are also related to the primal dual indicators. The key result in this survey is the proof that the convergence rate of the indicator sets to their optimal value is faster than the convergence rate of the solution. Thus, the solution set can be identified before the actual solution is reached. From a practical point of view, the need is to find some criteria that give a sharp separation between the sets of active and dormant variables. It is clear that the sets defined in Theorem 2.1 do not give such a separation. El-Barki's survey and our investigations indicate that using the variable values as indicators suffers from the same disadvantage: in the NETLIB problem Pilot, for example, some variables that participate in the solution have very low values; even at a relatively late stage of the optimization it is difficult to distinguish between them and the variables that are going to their bounds. In primal only or dual only algorithms, however, the indicators are usually based on such a criterion (Gay 88).

In primal dual IPM algorithms, both primal and dual solution vectors are computed at each iteration. This provides an opportunity to compute indicators based on the primal and dual behaviour throughout the iterative process. Mehrotra (91-2), Tapia, El-Barki (91) and

others have proposed the following indicator function :

$$\delta_M(x_j)^k = \frac{|x_j^{k+1} - x_j^k|}{x_j^k}$$
$$\delta_M(z_j)^k = \frac{|z_j^{k+1} - z_j^k|}{z_j^k} \qquad (2.7)$$

These indicator functions are applied to the primal and dual variables to create the indicator set in the $(k+1)^{th}$ iteration of IPM:

$$\sigma^k = \{j : \delta_M(x)_j^k \le \delta_M(z)_j^k\} \qquad (2.8)$$

Mehrotra uses this set σ^k of (2.8) to define the active variables of (2.6).

The primal dual indicators used by Gay, Lustig and others (Gay 88) are stated as follows:

$$\delta(x_j z_j)^k = \frac{x_j^k}{z_j^k} \qquad (2.9)$$

These indicators are based on the simple observation that when the sequence of solution points converges to the optimal solution, the indicator for the active variables converges to infinity while the indicator of a dormant variable converges to zero. This property provides a sharp separation between the two sets but only in the last few iterations of the algorithm.

For our investigations, we developed a different kind of primal dual indicators. These indicators can be seen as a combination of those used by Tapia and those used by Lustig. The indicator function is stated below:

$$\delta_j^k = \frac{\delta(x_j)^k}{\delta(z_j)^k} \qquad (2.10)$$

Where

$$\delta(x_j)^k = \frac{x_j^{k+1}}{x_j^k + 1}, \quad \delta(z_j)^k \frac{z_j^{k+1}}{z_j^k + 1} \qquad (2.11)$$

These indicator functions are used in a prediction heuristic incorporated into our primal dual

predictor corrector solver. The predicted sets of active and dormant variables are used in our cross-over procedure whereby the predicted solution variable set is used to fix the initial basis.

3. Cross over to SSX and basis recovery

In non degenerate LP problems, the optimal solution point set is restricted to a single extreme point of both primal and dual feasible polyhedrons. In most practical cases, however, the LP problem is primal or dual degenerate (or both) and the optimal solution point set describes a face of the feasible polyhedrons. It is well known that the optimal solution generated by the primal dual IPM algorithm converges to an interior point of this set (Megiddo 89). In many cases, however, an optimal extreme point solution is required. This optimal extreme point solution and the corresponding basis provide a powerful representation which arises in LP and its duality theory. Consider the LP problem of (2.1) and assume that the constraint matrix A is of full rank. For our purposes it is sufficient to state that the optimal basis of this LP problem is a submatrix B, $B\in\mathbb{R}^{m\times m}$ such that $(x^*)^T=[(B^{-1}b)^T,0], x^*\in\mathbb{R}^n$ is an optimal solution of the primal problem and $y^*=(B^T)^{-1}c_B$ is the optimal solution for the dual problem. A primal (non degenerate) basic solution requires exactly m primal variables to be active and their corresponding dual slack variables to be 0. Compared to optimal SSX solution for a given LP, the IPM solution has more active variables if the problem is dual degenerate or less if it is primal degenerate. In the integration of IPM and SSX a basis recovery procedure has to be constructed which applies to both these cases.

There are essentially three related problems which lie at the heart of the basis recovery procedure which may be stated as:

(i) given a primal **feasible** interior point solution $\hat{x}$, find a superior basic feasible solution x_B, such that $c_B^T x_B \le c^T \hat{x}$,

(ii) given a set of primal or dual **optimal** solution values, find B_{opt}, such that:

$x^*_{B_{opt}} = B_{opt}^{-1} b \ge 0$ *and* $[c^T_{B_{opt}} B_{opt}^{-1} A]^T \ge c$

(iii) given a set of primal and dual **optimal** solution values, find B_{opt}^{-1} as in (ii).

We note that if IPM stops at a primal feasible but suboptimal interior point we have to apply procedure (i); on the other hand, if the IPM terminates near enough to the optimal solution then solution procedures for (ii) or (iii) may be applied.

Basis recovery using a primal quantitative approach

The recovery of a superior or an optimum basic solution from a primal feasible non-extreme point solution is well understood in the context of the SSX algorithm. For instance the BASIC procedure within MPSX finds a superior extreme point solution and the corresponding basis. Mitra and Tamiz (88), Marsten (89), and others set out comparable pivotal schemes for a primal only approach to this problem. The steps of this algorithm are summarized below:

Let x^* be the primal feasible (or optimal) non-extreme point solution found by the interior point method. We partition the A matrix into three parts $[B, N, S]$ and the corresponding variable or column indices into three sets $[I_B, I_N, I_S]$. These correspond to basic, nonbasic variables at their bounds and superbasic variables. Let

$x^*=\left[x_B^*, x_N^*, x_S^*\right]^T$ then we can express the original LP system of equation as :

$$Ax^*=b : Bx_B^*+Nx_N^*+Sx_S^*=b \tag{3.1}$$

we also set

$$\forall j\in I_N: \ \textit{if} \ x_j<\epsilon \ \textit{then} \ x_j=0 \tag{3.2}$$

and re-express the system (3.2) as:

$$x_B^*=B^{-1}(b-Nx_N^*)-B^{-1}Sx_S^* \tag{3.3}$$

Note that since no upper bound is defined $Nx_N^*=0$.

To this equation representation of the problem we apply a "reverse simplex" algorithm in which the number of variables which are nonzero are reduced from $|I_B|+|I_S|$ to $|I_B|$ in $|I_S|$ pivotal steps and the corresponding x_S^* variables moved to their bound depending on their reduced cost coefficients; for a detailed explanation of this procedure see (Mitra 88). This method was also implemented within the first versions of OB1 (Marsten 89-2) and OSL (Forrest 90). The experimental results, however, were rather disappointing. On average the basis recovery steps took 15-30% of IPM time and sometimes it exceeded the IPM time. The difficulty experienced in the basis recovery (SSX) steps can be ascribed to the cases where the optimal solution is primal or dual degenerate.

Recently, Bixby and Saltzman (92) analyzed the above approach and suggested several extensions. They attribute the slow convergence of primal cross-over algorithm to one of the two following reasons:

(i) Let $x^*=(x_B^*,x_N^*,x_S^*)$ be the partition of the IPM solution to basic, superbasic and non basic variables. After the initial basis is constructed, a basic solution $\bar{x}_B=B^{-1}(b-Nx_N^*-Sx_S^*)=B^{-1}\bar{b}$ is computed. There is no guarantee that this solution is numerically stable and thus the residual $\bar{b}-B\bar{x}_B$ can be unacceptably large. Even if this residual is small, the computed $\bar{x}_B$ can be a bad approximation to the original x_B.

(ii) The variables that are superbasic but were not included in the basic solution are fixed to their bounds. The computed $\bar{x}_B$ may be highly sensitive to these perturbations.

To over come these difficulties, Bixby and Saltzman propose the following procedures:

(i) Construction of the initial basis

The variables x_j *such that* $x_j>0$ are sorted in a decreasing order, a variable is considered to be 0 if $x_j<10^{-5}$. The initial basis is constructed from the first m variables in the list, the rest

of the variables that are not set to zero are considered as superbasic.

(ii) Singularity tolerance

The LP is scaled such that $\max_{ij}|a_{ij}| \leq 1$, the standard singularity tolerance $\tau = 10^{-8}$ which gives the minimum of an acceptable pivot entry in the SSX algorithm is set to $\tau = 10^{-3}$. Columns with no acceptable pivot value are rejected. If this or the original selection results in an incomplete basis then non basic slack variables that correspond to uncovered rows are included in the basic.

(iii) Feasibility check

After the initial basis is constructed using steps (i) and (ii) $\bar{x}_B$ is computed. If the scaled sum of infeasibilities is more then 1.0, the basis is rejected and step (ii) is repeated with $\tau = 0.1$.

The basis constructed by this heuristic procedure is used in an algorithm similar to the one used by Marsten et al. and Mitra et al.

The results reported by Bixby et al. indicate that their heuristic improves the primal basis recovery algorithm dramatically. In average, the reported cross-over times are 5% of the total solution time.

Basis recovery using a primal dual quantitative approach

Megiddo (91) proved that from theoretical point of view, a cross-over algorithm which uses both primal and dual information is preferable. This idea is encapsulated in the following theorems:

Theorem 3.1: If there exists a strongly polynomial time algorithm that finds an optimal basis, given an optimal solution for either the primal or the dual, then there exists a strongly polynomial algorithm for the general LP problem.

Theorem 3.2: There exists a strongly polynomial time algorithm that finds an optimal basis, given optimal solution for both the primal and the dual.

Megiddo gives a constructive proof of Theorem 3.2 and shows that there is no known strongly polynomial algorithm to retrieve the optimal basis if we take into consideration only the primal or the dual optimal interior solution. This explains the importance of problem (iii) stated above. Megiddo's procedure which takes into account both primal and dual optimal solutions values is given in algorithm 3.1.

Algorithm 3.1: recovering an optimal primal dual basis

Let (x^*, y^*, z^*) *be the IPM optimal solution of the problem stated in (2.1) and assume that A is of full rank (the rows of the matrix A are linearly independent). Our aim is to construct an optimal basis solution from the optimal interior point solution.*

Let A_{X^*} *be the part of the constraint matrix which corresponds to the variables that are active at the solutions :* $A_{X^*}=\{a_j \mid a_j$ *is a column of A and* $x_j^*>0\}$. *Similarly, let* A_{Y^*} *be the part of the constraint matrix consisting of columns of A which correspond to dual slacks that are 0 in the solution:* $A_{Y^*}=\{a_j \mid a_j$ *is a column of A and* $z_j^*=0\}$ *(these sets are identical if the problem is non degenerate).*

From duality theory and the complementarity slackness conditions:

$z_{X^*}^*=0, (y^*)^T A_{X^*}=c_{X^*}^T$

1. Construct a minimal size primal solution

Reduce the size of the submatrix A_{X^*} *to create a linearly independent matrix by repeating the following procedure until no reduction is possible:*

If A_{X^*} ***is linearly dependent then there exists a vector*** $\eta \neq 0$ ***such that*** $A_{X^*}\eta=0 \Rightarrow c_{X^*}^T\eta=0$

thus for some scalar t we construct the vector x' ***defined by the relation:***

$$x': x_j'=\begin{cases} x_j+t\eta_j & a_j \in A_{X^*} \\ 0 & \text{otherwise} \end{cases} \quad \forall j=1,..,n \quad \text{such that } \exists j: \; x_j+t\eta_j=0$$

The vector x' ***is still an optimal solution but with a smaller set of positive indices.***

The result of this procedure is the submatrix $A_{X'}=\{a_j:x'_j>0\}$ *whose columns are all linearly independent.*

If $|A_{X'}|=m$ *then set* $B=A_{X'}$, *GOTO step 4*

2. Add variables according to the dual problem:

While (there is a column in $A_{Y\bullet}$ *which is independent of the columns of* $A_{X'}$ *) do*

add the column to $A_{X'}$

(the linear independence of the column can be easily checked by using Gaussian elimination)

end while

If $|A_{X'}|=m$ *then set* $B=A_{X'}$, *GOTO step 4*

3. Add more variables by dual range check

Since A is of full rank and $|A_{X'}|<m$ *then there must exist at least one column which is independent of the columns of* $A_{X'}$. *Let* a_j *be such a column then* a_j *cannot be in the original* $A_{X\bullet}$ *or* $A_{Y\bullet}$, *hence* $a_j:(y^*)^TA_j<c_j$. *For such a column* a_j *we solve the following system of equations:*

$$\xi^TA_{X'}=0$$

$$\xi^Ta_j=1$$

A solution for such a system exists because the system is linearly independent.

Since every column of $A_{Y\bullet}$ *is now a linear combination of columns of* $A_{X'}$, *then for every scalar t we have:* $(y^*-t\xi)^TA_{Y\bullet}=c^T_{Y\bullet}$.

We fix:

$$t_0=\min\left\{\frac{c_k-(y^*)^Ta_k}{\xi^Ta_k} \mid \xi^Ta_k>0\right\}$$

$$y'=y-t_0\xi,\quad A_{Y'}=\{a_k|y'^Ta_k=c_k\}$$

The submatrix $A_{Y'}$ contains at least one column which is linearly independent of the columns of $A_{X'}$. These columns are added to $A_{X'}$ one at a time in the same way as in step 2. This process is repeated until $|A_{X'}|=m$, then set $B=A_{X'}$

4. B is the desired primal and dual basis

Throughout the process described in algorithm 3.1 the primal and dual optimality conditions of the intermediate solutions x',y',z' are maintained such that $\forall a_k \notin A_{X'}$ $x'_k=0$. Variants of Algorithm 3.1 were implemented by Forrest (91) in the IBM OSL library and by Lustig (91-1) in OB1. These implementations proved that, from the practical point of view, this primal dual procedure is more efficient than the primal only basis recovery procedure.

Basis recovery using a primal qualitative approach

We now describe a qualitative method that we have developed for the crossover procedure. This procedure was especially developed to utilize the prediction of the optimal solution using active and dormant sets as described in the previous section. Instead of predicting the optimal solution and performing another IPM iteration to verify it, we use the prediction inside the SSX algorithm.

Given the original LP problem P we create a related problem $\bar{P}$ in the following way. In addition to I_B, I_N, I_S column index sets defined in the primal retrieval procedure we define R_E, R_P row index sets corresponding to zero and positive logical respectively. The relationship of the indicator sets to these sets are given as $\sigma^*(x)=I_B \cup I_S$, $\bar{\sigma}^*(z)=R_E$.

We construct the related problem $\bar{P}$ as shown in Figure 3.1 by fixing variables in the set I_N to their respective upper or lower bounds and making the rows in the set R_P free rows. We note that if the IPM prediction is correct then the given optimum solution to P is a feasible solution to $\bar{P}$. The basis recovery procedure is stated in algorithm 3.2:

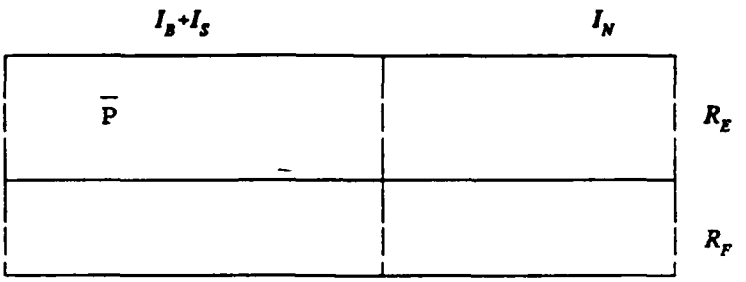


Figure 3.1 : construction of the restricted problem

Algorithm 3.2

1. Create a starting basis for $\bar{P}$ using a the simplex CRASH procedure (Forrest 90, NAG 91).

2. Solve $\bar{P}$ by a SSX algorithm, save the basis $B^{-1}(\bar{P})$

3. Reinstate the original problem P

start with $B^{-1}(\bar{P})$ and apply SSX to optimality.

If the prediction of the optimal basis is slightly wrong (as can happen if IPM is terminated before an optimal solution is reached) then step 2 is terminated with a 'no feasible solution' status. The resulting basis, however, is usually near optimal and thus requires a low number of iterations in step 3 to reach optimality.

4. Results and analysis

Figures 4.1, 4.2 and 4.3 illustrate the behaviour of our primal-dual indicators. Figures 4.1 and 4.2 show the behaviour of primal and dual Tapia's indicators for two dormant variables (var 3 and var 4) and two active ones (var 1 and var 2) on the problem Stair. As expected, in Figure 4.1 the indicators which represent the active variables converge to 1 while those which represent the dormant ones converge to 0. Figure 4.2 is the dual mirror image of figure 4.1. In it, the dual slacks of the active primal variables converge to 0 while the dual slack variables of the dormant primal demonstrate consistent growth. The convergence is monotonous only in the last stages of the algorithm and for some problems a sharp separation between the sets is not always easily derived (especially if some upper bound variables are

also present). In comparison, the behaviour of the indicators in Figure 4.3 that are based on equation (4.10) show an earlier as well as stronger separation. Our observations show that soon after feasibility is reached, many of the active primal variables show a rapid growth for several iterations. This growth is closely linked to the rapid arrival of IPM to a near-optimal solution; then, the growth of the primal variables is followed by a similar reduction in the value of the dual slack variables and the indicator shows an exponential growth from iteration to iteration. This growth is reduced when the solution reach optimality.

The behaviour of the primal, dual and our primal dual indicators are tested on four variables of the problem Stair as illustrated in Figures 4.1, 4.2 and 4.3

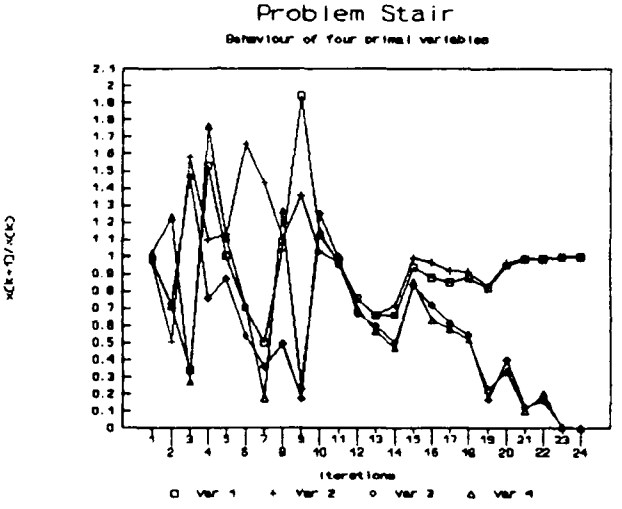


Figure 4.1 Primal indicators the problem Stair

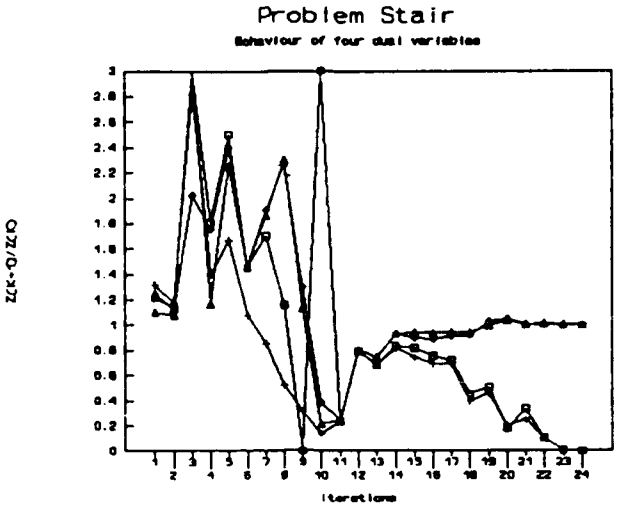


Figure 4.2 Dual indicators in the problem Stair

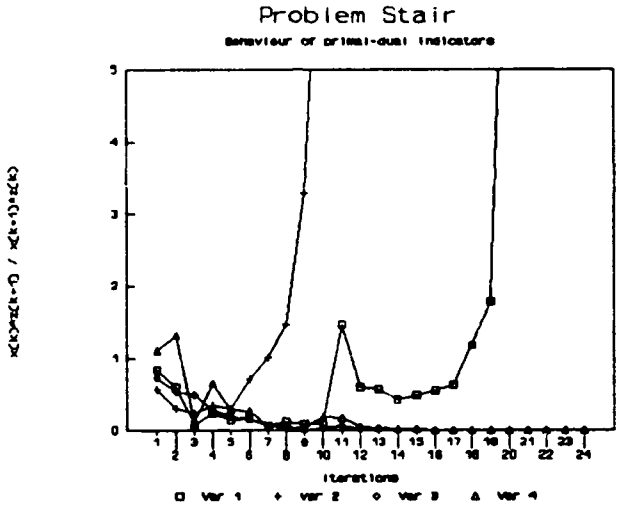


Figure 4.3 Primal-dual indicators in the problem Stair

In our identification heuristic, we use a combination of the prima dual indicators and those of (2.10) to identify the active and dormant sets. A special array *predict(ncol)* holds an integer value that is proportional to growth in the indicator value. If the value is increased by a sufficient amount for more than one iteration, the corresponding variable is marked as active. If the value is decreased by the same amount, the corresponding variable is marked as dormant.

In Table 4.1 and Table 4.2 we present the results of our identification heuristic in intermediate stages of the IPMC algorithms for two NETLIB problems. The table shows the number of active and dormant variables which are identified by the heuristic (the column marked Forc.) and compare them to those found at optimality. The number of hits and misses in every iteration is also given.

Iterations from optimality	Active variables			Dormant variables			Unknown
	Forc.	Hit	Miss	Forc.	Hit	Miss	
-10	195	170	25	264	236	28	10
-9	192	180	12	269	247	22	8
-8	199	192	7	268	262	6	2
-7	198	194	5	270	266	4	1
-6	199	197	2	269	269	0	1
-5	198	197	1	272	272	0	0
-4	197	197	0	272	272	0	0
-3	197	197	0	272	272	0	0
-2	197	197	0	272	272	0	0
-1	197	197	0	272	272	0	0
Optimality	197	197	0	272	272	0	0

Table 4.1 Prediction of the optimal solution set in the problem E226.

Iterations from optimality	Active variables			Dormant variables			Unknown
	Forc.	Hit	Miss	Forc.	Hit	Miss	
-10	1040	741	299	738	734	4	71
-9	998	743	255	825	825	0	46
-8	880	742	138	932	931	1	37
-7	905	743	162	902	902	0	42
-6	889	742	147	919	918	1	41
-5	818	742	76	999	998	1	32
-4	771	743	28	1071	1071	0	7
-3	746	743	3	1101	1101	0	2
-2	743	743	0	1106	1106	0	0
-1	743	743	0	1106	1106	0	0
Optimality	743	743	0	1106	1106	0	0

Table 4.2 Prediction of the optimal solution set in the problem 25FV47

The tables indicate that most dormant variables and some active variables can be recognized almost as soon as feasibility is reached. This property of the algorithm is later used for basis recovery and for reducing the model size dynamically by removing these dormant variables.

In Table 4.3 we demonstrate the qualitative approach on the problem OR3 (algorithm 3.2). Apart from the only SSX run (first row of the table) and cross over from optimality (last row of the table) the solution set is determined by the indicator heuristic.

The "IPM Iterations" column gives the iteration number at which cross over was made. The column marked "Basis recovery Pass I" gives the number of SSX iterations on the restricted problem, the column marked "Basis recovery Pass II" gives the number of SSX iterations needed to prove the optimality of the solution for the complete problem. We note that the predictions of the 14 and 16 iterations did not find the exact optimal set but as expected, they where close enough to the primal basis to cause a considerable reduction in the number SSX iterations. In Table 4.3 we give the qualitative basis recovery results for some more NETLIB problems.

Model	IPM Iterations	Time (sec)	Basis recovery		Time	Total Time
			Pass I	Pass II		
OR3	0	0		1765	354	354
OR3	14	10.5	200[1]	1480	365	365
OR3	16	12	40[1]	680	216	227
OR3	18	13.5	100	340	135	147
OR3	20	15	180	71	91	105
OR3	22	16.5	240	6	89	104
OR3	24	18	220	13	78	95

[1] Optimal solution not found in the restricted problem

Table 4.3 quantitative crossover results

Model	Iteration	Variable	Time	Pass I	Pass II	Time	Total
Ganges	26	1223	87	336	379	33.8	120.80
Ganges	30	1223	101.1	405	187	28.6	129.70
25fv47	39	773	214.61	267	206	25	239.61
25fv47	44	770	242	331	124	23.8	265.80
Ship12l	19	736	29	39	159	23	52.00
Ship12l	26	726	39.78	38	67	18.5	58.28
Cre_a	45	977	185	80	2423	224	409.00

Table 4.4 Basis recovery results

5. References

(Adler 89) Ilan Adler, Renato D.C. Monteiro, A geometric view of parametric linear programming, Industrial engineering and operations research department, University of California at Berkeley, Berkeley CA 94720 USA, February 1989.

(Bixby 91-1) R.E. Bixby, J.W. Gregory, I.J. Lustig, R.E. Marsten, D.F. Shanno, A very large scale linear programming: a case study in combining interior point and simplex methods. TR-91-11, Department of mathematical sciences, Rice University, Houston, TX 77251, USA, May 1991.

(Bixby 91-2) R.E. Bixby, The Simplex method- It keeps getting better,Notes from a lecture given to the MP symposium, Amsterdam, August 1991.

(Bixby 92) R.E. Bixby, M.J. Saltzman, Recovering an optimal basis from an interior point solution, Technical report #607, Department of mathematical sciences, Clemenson University, Clemenson, South Carolina, USA March 1992.

(El-Barky 91) A.S. El-Barky, R.A. Tapia, Y. Zhang, A study of indicators for identifying zero variables in interior point methods, presented to ICIAM 91, washington D.C. july, 1991.

(Forrest 90) J. Forrest, J.A. Tomlin, OSL Optimization subroutine library version 2.0, user guide and reference, IBM, 1990.

(Gay 89) David M. Gay, Stopping tests that compute optimal solutions for interior point linear programming algorithms, Numerical analysis manuscript 89-11, AT&T Bell laboratories, Murray Hill N.J., USA 07974, December 1989.

(Guler 91) O. Guler, Y. Ye, Convergence behaviour of some interior point algorithms, Working paper 91-4, The college of business administration, The University of Iowa, Iowa city, IA, 1991.

(Levkovitz 91-1) R. Levkovitz, G Mitra, M. Tamiz, Integration of the interior point method within simplex: experiments in feasible basis recovery, Present in APMOD91 symposium, Brunel University, U.K., January 1991.

(Levkovitz 92) R. Levkovitz, Interior point methods for large scale linear programs, theory and computational algorithms,

(Marsten 89-1) Roy Marsten, David Shanno, Interior point method for LP: ready for production use, ORSA/TIMS workshop, New York, USA, 15 October 1989.

(Marsten 89-2) Roy Marsten, OB1 user manual, primal dual interior point method for linear programming, XMP software inc. P.O.Box 58119, Tucson, AZ 85732, USA. 1989.

(Megiddo 88) N. Megiddo, Switching from a primal dual newton algorithm to a primal dual (interior) simplex algorithm, RJ 6327 (61996), Computer Science/Mathematics, IBM Almaden research center, 650 Harry road, San Jose, California 95120-6099, USA, 7 November 1988.

(Megiddo 89) N. Megiddo, M. Shub, Bondary behaviour of interior point algorithms in linear programming, Mathematics of operations research vol 14. no. 1., February 1989.

(Megiddo 91) N. Megiddo, On finding primal and dual optimal bases, ORSA journal on computing vol. 3 no. 1, Winter 1991.

(Mehrotra 90) Sanjay Mehrotra, High order methods and their performance, Technical report 90-16R1, Department of industrial engineering and management sciences, Northwestern University, Evanston, IL, 60208-3119, U.S.A, July 1990.

(Mehrotra 91-1) Sanjay Mehrotra, Handling free variables in interior methods, Technical report 91-06R1, Department of industrial engineering and management sciences, Northwestern University, Evanston, IL, 60208-3119, U.S.A, March 1991.

(Mehrotra 91-2) Sanjay Mehrotra and Yinyu Ye, On finding the optimal facet of linear programming, Technical report 91-10, Department of industrial engineering and management sciences, Northwestern University, Evanston, IL, 60208-3119, U.S.A, June 1991.

(Mitra 88) G. Mitra, M. Tamiz, J. Yadegar, Experimental investigation of an interior search method within a simplex framework, CommUnications of ACM vol. 21 no. 12, December 1988.

(Zhang 91) Yin Zhang, Richard A. Tapia, On the convergence of interior point methods to the centre of the solution set in linear programming, Department of mathematics and statistics, University of Maryland, Baltimore county campus, Baltimore, Maryland 21228 USA, September 1991.

An Adaptive Stochastic Global Optimization Algorithm for One-dimensional functions*

Marco Locatelli Fabio Schoen†
Department of Computer Sciences - University of Milano
via Comelico 39/41 - I-20135 Milano (Italy)
e-mail:schoen@ghost.dsi.unimi.it

keywords: *global optimization, Wiener process, sequential stopping rules, one-step look-ahead*

Abstract

In this paper a new algorithm is proposed, based upon the idea of modeling the objective function of a global optimization problem as a sample path from a Wiener process. Unlike previous work in this field, in the proposed model the parameter of the Wiener process is considered as a random variable whose conditional (posterior) distribution function is updated on-line. Unfortunately, using a natural conjugate prior distribution on such a parameter, consistency of the Bayesian algorithm is lost. The authors propose a modified prior distribution which overcomes this difficulty. Furthermore, stopping criteria for Bayesian algorithms are discussed.

Introduction

Let us consider the global optimization problem defined as the problem of finding f^* in such a way that

$$f^* = \max_{x \in K} f(x) \tag{1}$$

where $K \subseteq \mathbb{R}^n$ is a compact set and f is a continuous, real-valued function defined over K. Many algorithms have been proposed in the literature for dealing with the rather frequent situation in which f possesses many local optima. For a general survey the intersted reader is referred to (Törn & Žilinskas, 1989). Among the algorithms which deserve special attention are, in the authors' opinion, those based on stochastic elements: research in this area may be roughly

*This research has been partially supported by "Progetto MURST 40% Metodi di Ottimizzazione per le Decisioni".

†to whom all correspondance should be sent

categorized in two main streams, namely research based on the idea of modeling the objective function f as a sample path of a stochastic process and research based on the introduction of a random element in the algorithm itself. Algorithms belonging to the former stream have been recently surveyed in (Betrò, 1991), while those belonging to the latter are surveyed in (Schoen, 1991). A very good account of classical as well as recent results in both areas can be found in (Zhigljavsky, 1991).

In this paper attention will be restricted to the class of objective functions defined over a closed interval of the real axis. Such a restriction is, from the *theoretical* point of view, admissible in view of the fact that multidimensional global optimization problems can be transformed into one-dimensional ones e.g. by means of Peano-mappings (Strongin, 1992). The authors are obviously aware of the *computational* difficulties inherent to such transformation; however the analysis of one-dimensional global optimization problems is already sufficiently complex to be worthwhile, and, in the authors' opinion, it can shed light into the challenging problem (1).

One of the most well-known and best performing method for one-dimensional global optimization, known as the Bayesian approach, was introduced in (Mockus, 1975) and extensively reviewed in (Mockus, 1989).

The main idea of the Bayesian approach is that of considering the objective function f as a sample path of a stochastic process. Following the traditional scheme of Bayesian analysis a loss function is introduced and the "decision" to be taken at each iteration of the algorithm, namely the choice of a point $x \in K$ where to evaluate f, is made according to the minimization of the expected loss, or, in Bayesian terminology, risk.

1 Sequential Bayesian optimization

Let the objective function f be considered as a sample path of a stochastic process $F(x;\omega)$; i.e., we assume that there exists an $\bar{\omega} \in \Omega$ such that $f(x) = F(x;\bar{\omega})\,\forall\, x$; a suitable probability space over the sample space Ω is assumed to be defined. The idea of Bayesian algorithms for problem (1) is that the "decision-maker" should choose an "action" a, which, in the present context, corresponds to choosing a point where to evaluate f, trying to minimize expected value of a suitably defined "loss function". The reader is referred, for example, to (Berger, 1985) for a detailed account of Bayesian analysis and terminology. Let us assume that n observations of f have been already performed in correspondance with the points $x_1 < \cdots < x_n$ and let $f_n^* = \max_{i=1,n} f(x_i)$. A natural form of the loss function for problem (1) is given by

$$\mathscr{L}(\omega, a) = \max_y F(y;\omega) - \max\{F(a;\omega), f_n^*\}. \tag{2}$$

The above loss function introduces a measure of the loss incurred when an action consisting in the evaluation of the objective function at $a \in K$ is taken

when the objective function $f(\cdot)$ is $F(\cdot;\omega)$; it is assumed that f has already been evaluated in n points.

In most of the published papers on this subject it is assumed that a fixed number of observations, $N > 0$, can be performed. However it seems more sensible to assume that the disadvantage of having to evaluate the (presumably) expensive function f is directly included in the definition of the loss function. We propose thus the use of the following loss:

$$\mathscr{L}(\omega, a, n) = \max_y F(y;\omega) - \max\{F(a;\omega), f_n^*\} + (n+1)c \tag{3}$$

where $c > 0$ measures the cost of each function evaluation; here, as before, action $a \in K$ is chosen: having already evaluated f in n points, a further observation at a gives a total of $n+1$ observations, each at cost c.

Let

$$z_k = (x_1, f(x_1), \ldots, x_k, f(x_k))$$

be the information available after the k-th observation, with $z_0 = \emptyset$. A sequential decision is thus defined as a function

$$d = (\tau, \delta)$$

where

$$\begin{aligned} \tau &= \tau_0, \tau_1(z_1), \ldots, \tau_k(z_k), \ldots \\ \delta &= \delta_0, \delta_1(z_1), \ldots, \delta_k(z_k), \ldots \end{aligned}$$

Functions τ_k map the available information at step k into one of the actions "stop" and "don't stop"; functions δ_k give the decision on where to choose the next observation point.

Altough theoretically it is possible to define an optimal sequential decision rule, it is practically impossible to identify a closed form expression or even a computationally manageable optimal rule. It is thus very common practice to implement a so-called rolling-horizon or k-step look-ahead rule. Given the practical difficulty of even these decision strategies, one is usually left to the use of 1-step look-ahead rules, i.e. rules which are optimal in the subset of decision rules which prescribe stopping not later than the next observation. The "rolling-horizon" feature of this method arises as, in the implementation, if the rule calls for stopping then the algorithm terminates, whereas if the rule calls for one more observation, that observation is taken and a new 1-step look-ahead rule is implemented.

In the present context the bayes risk of a decision (τ_n, δ_n) is given by

$$r(\tau_n, \delta_n) = \begin{cases} E(\max_y F(y;\omega) - f_n^* + nc \mid vz_n) \text{ if } \tau_n = \text{"stop"} \\ E(\max_y F(y;\omega) - \max\{F(\delta_n;\omega), f_n^*\} + (n+1)c \mid z_n) \text{ otherwise} \end{cases} \tag{4}$$

where E stands for expectation.

A one-step look-ahead rule $(\tau_n^\star, \delta_n^\star)$ has the form:

$$\delta_n^\star = \arg\min_{x \in K} E\left(\max_y F(y;\omega) - \max\{F(x;\omega), f_n^\star\} + (n+1)c \mid z_n\right) \quad (5)$$

and $\tau_n^\star$ = "stop" if and only if the current loss

$$E\left(\max_y F(y;\omega) - f_n^\star + nc \mid z_n\right)$$

is less than or equal to the predicted one:

$$E\left(\max_y F(y;\omega) - \max\{F(\delta_n^\star;\omega), f_n^\star\} + (n+1)c \mid z_n\right). \quad (6)$$

It is easy to see that the previous optimal decision problem decomposes into the problem of optimally placing the next observation, which is equivalent to finding

$$\delta_n^\star = \arg\max_{x \in K} E\left(\max\{0, F(x;\omega) - f_n^\star\} \mid z_n\right) \quad (7)$$

and then deciding to stop computation (and not evaluate f at $x_{n+1} = \delta_n^\star$) if and only if

$$E\left(\max\{0, F(\delta_n^\star;\omega) - f_n^\star\} \mid z_n\right) \leq c \quad (8)$$

Stopping rule (8) gives further insight into the meaning of c which can be considered as a treshold for the gain which is expected in performing one more observation: if this expected improvement falls below c it is considered not wortwhile continuing the sample.

In order to specify a practical algorithm an appropriate probability measure has to be defined; this is the subject of the next section.

2 Wiener process with unknown parameter

A standard probabilistic model for the objective function f when K is an interval on the real line is the Wiener process. In the following we shall assume that $K = [0, 1]$. Although this process possess an undesirable feature, namely almost everywhere non differentiability, computational tractability makes it almost the only feasible choice. We say that a stochastic process $F(x;\omega)$ is a Wiener process if

- $F(0;\omega) = f_0$, a constant;
- $F(x;\cdot)$ is a random variable with distribution $\mathcal{N}(f_0, \sigma^2 x)$, where σ is a parameter of the process.
- $F(x;\cdot)$ has independent, stationary increments.

In the literature the parameter σ is assumed to be known or, in some cases, estimated by means of an initial sample from f. In this paper the Bayesian paradigm will be assumed and σ will be considered as a random variable which has to be estimated on-line.

As a consequence of the normality assumption for $F(x;\cdot)$, it is only too natural to assume as a prior distribution on σ a "natural conjugate prior": definitions and examples of conjugate classes of distributions can be found in any text on Bayesian statistics. As the observations of a Wiener process are gaussian random variable whose variance is proportional to the unknown parameter σ^2, a natural choice for the prior distribution comes from the observation that, in an independent normal process $\{X_i\}$ with known mean μ and variance σ^2, the random variable

$$\frac{S_n}{\sigma^2} = \frac{\sum_{i=1}^{n}(X_i - \mu)^2}{\sigma^2}$$

is distributed as a χ^2 random variable with n degrees of freedom. Inverting the distribution it is thus possible to define a family of prior probability distribution functions for σ^2 which depend on two parameters, a_0 representing the initial degrees of freedom and S_0 (the particular distribution goes under the name of inverted-gamma function), given by

$$p(\sigma^2; a_0, S_0) = g(\sigma^2; a_0, S_0) \propto \exp(-S_0/(2\sigma^2))(\sigma^2)^{(a_0+1)/2}. \tag{9}$$

It is then possible to show (the detailed proofs of all the original results in this paper will appear elsewhere) that, given a sample z_n from the Wiener process F the *posterior distribution* of σ^2 is given by

$$p(\sigma^2 \mid z_n; a_0, S_0) = g(\sigma^2; a_n, S_n) \tag{10}$$

where

$$\begin{aligned}
a_n &= a_{n-1} + 1 \\
S_n &= S_{n-1} + \left(\frac{f(x_n) - \mu(x_n \mid z_{n-1})}{\sigma(x_n \mid z_{n-1})}\right)^2 \\
\mu(x \mid z_{n-1}) &= E(F(x;\omega) \mid z_{n-1}) \\
&= \begin{cases} f_1 & \text{if } x \le x_1 \\ f_i \frac{x - x_{i-1}}{x_i - x_{i-1}} + f_{i-1} \frac{x_i - x}{x_i - x_{i-1}} & \text{if } x_{i-1} \le x \le x_i \\ f_{n-1} & \text{if } x \ge x_{n-1} \end{cases} \\
\sigma^2(x \mid z_{n-1}) &= \begin{cases} x_1 - x & \text{if } x \le x_1 \\ \frac{(x - x_{i-1})(x_i - x)}{x_i - x_{i-1}} & \text{if } x_{i-1} \le x \le x_i \\ x - x_{n-1} & \text{if } x > x_{n-1} \end{cases}
\end{aligned}$$

Thus the posterior distribution of the parameter of the Wiener process can be updated sequentially by means of the above formulae.

3 One-step look-ahead algorithm

The choice of the next observation point is thus reduced to the maximization of the expected gain

$$T_x(f_n^\star) = E\left(\max\{0, F(x;\omega) - f_n^\star\} \mid z_n\right) \quad (11)$$

$$= E\left(E\left(\max\{0, F(x;\omega) - f_n^\star\} \mid z_n, \sigma^2\right) \mid z_n\right) \quad (12)$$

where the outermost expectation is with respect to the conditional distribution of σ^2 given the sample. It is possible to show that, at least when a_n is even, $T_x(f_n^\star)$ can be given in explicit form. Obviously, as this case generalizes the usual one with σ^2 a priori known, explicit maximization of the expected gain T_x is impossible and one has to resort to numerical optimization; however it has been shown in (Locatelli, 1992) that several criteria can be given for excluding subsets of $[0,1]$ from the search of an extremum of T_x, and, in many cases, conditions have been given by which such exclusion is permanent; in other words it is possible to show that if the expected gain in a certain sub-interval of $[0,1]$ falls below a treshold at step n, the Bayesian algorithm will not place any new observation in that interval before stopping.

For what concerns stopping of the algorithm, the one-step look-ahead stopping rule, once the choice of the new observation point x_{n+1} has been performed, is trivially obtained as that rule which calls for stopping as soon as the computed expected gain falls below c. Thus the proposed modification of the Bayesian approach permits at no cost to regulate the sample size dinamically on the bases of the observations.

The following result has also been proven:

Theorem. *If f is a continuous function on $[0,1]$ satisfying*

$$|f(x) - f(y)| < L\sqrt{|x-y|} \quad \forall\, x, y \in [0,1] \quad (13)$$

where $L > 0$, then the one-step look-ahead Bayesian algorithm will almost surely stop after a finite number of steps. The stopping time is $O(L^2/c^2)$.

As a final remark, notice that only existence of L is required for finite stopping; knowledge of the constant L is not necessary.

4 Consistency

Although it has been proven that finite stopping always occur, the question naturally arises of the accuracy of the proposed algorithm. It should be clear that in looking for finite stopping rules one has to trade accuracy with computational effort. Nonetheless it seems worthwhile to analyze the asymptotic behaviour of the algorithm (as $c \to 0$) in order to judge its consistency, i.e. the convergence of the estimated optimum to the true one.

Unfortunately it is possible to exhibit counter-examples showing that, for particular choices of f, the Bayesian algorithm will not produce a dense set of observation points, thus missing some subset of $[0,1]$ with positive measure. This situation is typical for any Bayesian optimization algorithm relying upon an estimate of the variance parameter σ^2. In practice it is possible to exhibit functions which are constant in $[\epsilon, 1]$ with $\epsilon > 0$ chosen in such a way that no observation point will ever be placed in $(0, \epsilon)$; the estimate of σ^2 (*any* consistent estimate, not just the one proposed in this paper) will rapidly converge to 0 and the Bayesian optimizer will become more and more convinced that his/her function is everywhere constant.

It is however possible, albeit with some technical difficulty, to restore consistency by changing the prior distribution on σ^2 in a way which forbids the estimate to go to 0. In practice one chooses a small treshold $\epsilon > 0$ and redefines the prior in such a way as to give positive mass only to $\sigma^2 \in [\epsilon, +\infty)$. It is then again possible to find update formulae similar to those in (10) and to prove results on finite stopping as well as consistency. The details of this modified algorithm, as well as the proofs, will appear elsewhere.

Conclusions

It has been shown that it is possible, with only little increase of computational effort with respect to the traditional Bayesian algorithm, to achieve the following goals:

- eliminate the necessity of pre-specifying the value of the parameter σ^2;
- introduce a finite stopping criterion by which the sample size has not to be specified in advance;
- guarantee consistency.

References

Berger, J. O. (1985). *Statistical Decision Theory and Bayesian Analysis* (second edition). Springer Series in Statistics. Springer-Verlag, New York.

Betrò, B. (1991). Bayesian methods in global optimization. *Journal of Global Optimization*, *1*, 1–14.

Locatelli, M. (1992). Algoritmi bayesiani per la ottimizzazione globale. Unpublished laurea thesis.

Mockus, J. (1975). On bayesian methos of optimization. In Dixon, L. C. W & Szegö, G. P. (Eds.), *Towards Global Optimization* Amsterdam. North-Holland.

Mockus, J. (1989). *Bayesian Approach to Global Optimization.* Mathematics and Its Applications (Soviet Series). Kluwer Academic Publishers, Dordrecht.

Schoen, F. (1991). Stochastic techniques for global optimization: a survey of recent advances. *Journal of Global Optimization*, *1*, 207–228.

Strongin, R. G. (1992). Algorithms for multi-extremal mathematical programming problems employing the set of joint space-filling curves. *Journal of Global Optimization*, *2*, 357–378.

Törn, A. & Žilinskas, A. (1989). *Global Optimization.* Lecture Notes in Computer Sciences. Springer-Verlag.

Zhigljavsky, A. A. (1991). *Theory of Global Random Search.* Mathematics and Its Applications (Soviet Series). Kluwer Academic Publishers, Dordrecht.

SCALE SENSITIVITY AND RANK PRESERVATION IN THE MULTIPLICATIVE AHP AND SMART (SUMMARY)

F.A. Lootsma
Faculty of Technical Mathematics and Informatics
Delft University of Technology
Mekelweg 4, 2628 CD Delft, The Netherlands

Abstract

We consider first a variant of the Analytic Hierarchy Process (AHP) with a one-parametric class of geometric scales to quantify human comparative judgement, and with a multiplicative structure: logarithmic regression to calculate the impact scores of the alternatives at the first evaluation level, and a geometric-mean aggregation rule to calculate the final scores at the second level. We demonstrate that the rank order of the impact scores and the final scores is scale-independent. Moreover, we show that the multiplicative AHP is an exponential version of the Simple Multi-Attribute Rating Technique (SMART). In fact, the multiplicative AHP is concerned with ratios of intervals on the dimension of desirability, whereas SMART analyzes differences of the corresponding orders of magnitude.

1 Introduction

We are concerned with two well-known methods for multi-criteria decision analysis: the Analytic Hierarchy Process (AHP) and the Simple Multi-Attribute Rating Technique (SMART). They have primarily been designed to evaluate a finite number of decision alternatives $A_1, \ldots, A_n$ under a finite number of conflicting performance criteria $C_1, \ldots, C_m$, by a single decision maker or by a decision-making body. The AHP (Saaty (1980), see also Zahedi (1986) and French (1988)) is based upon pairwise comparisons of the alternatives and the criteria, so that the decision maker's judgement is rather fragmented. SMART (see von Winterfeldt and Edwards (1985)), a popular off-spring of Multi-Attribute Utility Theory (MAUT), proposes a direct rating procedure enabling the decision maker to keep a more holistic view on the decision alternatives.

Throughout the paper we illustrate the relationship between the multiplicative AHP and SMART. We demonstrate that the multiplicative AHP is concerned with ratios of intervals on the axis of desirability, whereas SMART uses the differences of the corresponding orders of magnitude. The idea was suggested by the familiar mode of operation in psycho-physics, where ratios of light and sound intensities are expressed, not necessarily in their original magnitudes, but in orders of magnitude on the deci-Bell scale.

2 Criticism on the original AHP

Saaty's original version of the AHP has been criticized for various reasons: (a) for the fundamental scale to quantify human judgement, (b) because it estimates the impact

scores of the alternatives by the Perron-Frobenius eigenvector, and (c) because it calculates the final scores of the alternatives via the arithmetic-mean aggregation rule. The controversial issues, to be treated in the presentation, are not new. Only a few years ago, Zahedi (1986) signalized that the criticism on the AHP concentrated on the estimation of the impact scores, but that no major controversy existed concerning the aggregation step. Criticism on Saaty's fundamental scale was not mentioned in Zahedi's survey paper, but Belton (1986) brought forward several arguments against the scale and the aggregation rule.

More recently, Barzilai et al. (1987,1991) observed that the AHP, since it is essentially based upon ratio information, would benefit from a conversion into a variant with a multiplicative structure. With the geometric row means of the reciprocal pairwise-comparison matrices to calculate the impact scores, and with a geometric-mean aggregation rule to calculate the final scores of the alternatives, one could aggregate in two different ways without affecting the final scores: either by combining first the pairwise-comparison matrices into one matrix from which one obtains the final scores, or by combining the impact scores under the respective criteria into a vector of final scores. By these multiplicative operations one avoids rank reversal when copies of alternatives are added to or deleted from a consistently assessed set of alternatives (a deficiency of the original AHP).

Barzilai et al. (1987, 1991) restricted their analysis to the case where one has exactly one estimate for each pair of alternatives, under every criterion. Similarly, they did not address the question of how to scale the decision maker's verbal judgement. These issues have been our concern since the early eighties. We proposed logarithmic regression in order to handle missing as well as multiple estimates (the author (1982, 1987, 1990); note that the regression problem reduces to the calculation of geometric row means when there is exactly one estimate for each pair of alternatives), and we introduced a one-parametric class of geometric scales in order to quantify the judgemental statements expressing the opinions of the decision makers. We could demonstrate that the rank order of the impact scores is scale-independent. For a detailed analysis we refer the reader to the author's report (1991). In the presentation, we explain the choice of values for the scale parameter via psycho-physical arguments, and we show, via a new definition of the criterion weights, that the rank order of the final scores is also scale-independent in a multiplicative structure.

3 The multiplicative AHP

In summary, we propose a multiplicative version of the AHP which operates as follows. In the basic experiment at the first evaluation level, where two alternatives A_j and A_k are compared under the criterion C_i, we collect the preference information (indifference, weak, strict, strong, or very strong preference for one of the two), and we convert the verbal statement of the decision maker (the selected gradation of his comparative judgement) into a numerical value on a geometric scale, that is, on a discrete scale with echelons constituting a series with geometric progression. Next, we use logarithmic regression to calculate the single-criterion impact scores $\overline{v}_i(A_j), j = 1, \ldots, n$, approximating the subjective values of the alternatives under criterion C_i. The impact scores are not unique.

They have a multiplicative degree of freedom, and they can accordingly be normalized in such a way that they sum up to unity.

The basic experiment at the second evaluation level, where two criteria are mutually compared, is somewhat more complicated. We suggest the decision maker to consider two real or imaginary alternatives with the property that his preference for one of them under the first criterion equals his preference for the other alternative under the second criterion. Next, we ask him to state whether he is indifferent between the two alternatives under the two criteria simultaneously, or whether one of the two criteria gives a decisive (weak, strict, strong, or very strong) preference for one of the two alternatives. Thereafter, the judgemental statements are converted into numerical values on a particular geometric scale. Logarithmic regression yields normalized weights $\overline{w}(C_i), i = 1, \ldots, m$, for the respective criteria. Finally, there is an aggregation step generating the final, multi-criteria scores $\overline{f}(A_j)$ via the geometric-mean aggregation rule

$$\overline{f}(A_j) = \alpha \prod_{i=1}^{m} (\overline{v}_i(A_j))^{\overline{c}_i},$$

where $\overline{c}_i$ simply denotes the weight $\overline{w}(C_i)$, and α the normalization factor to guarantee that the final scores sum up to unity. By these quantities, the alternatives are unambiguously ranked in a subjective order of preference when we operate with geometric scales. Moreover, the ratio of any two final scores does not depend on the physical or monetary units whereby the performance of the alternatives under the respective criteria is originally measured.

In the multiplicative AHP, the gradations of comparative judgement are put on a scale with geometric progression. Let us briefly explain the underlying reasons. We assume that the subjective weighing of the alternatives under a particular criterion is carried out in a given context represented by an interval on the corresponding axis or dimension. This interval is partitioned into subintervals which are felt to be of the same order of magnitude. Hence, the echelons demarcating the subintervals constitute a sequence with geometric progression; the property is well-known in psycho-physics, see Stevens (1957), Marks (1974), Michon, Eykman and de Klerk (1978), Roberts (1979), and Zwicker (1982). Finally, we take ratios of echelons to represent ratios of subjective values and we let them correspond with the gradations of comparative judgement. This enables us to assign numerical values to the gradations. Thus, we set

$$r_{jk} = \exp(\gamma \delta_{jk})$$

where δ_{jk} is an integer-valued index designating the gradation of the decision maker's judgement as follows:

-8 very strong preference for A_k versus A_j,
-6 strong preference for A_k versus A_j,
-4 strict (definite) preference for A_k versus A_j,
-2 weak (mild, moderate) preference for A_k versus A_j,
0 indifference between A_j and A_k,
+2 weak (mild, moderate) preference for A_j versus A_k,
+4 strict (definite) preference for A_j versus A_k,
+6 strong preference for A_j versus A_k,
+8 very strong preference for A_j versus A_k.

Intermediate, integer values of δ_{jk} designate hesitations between two adjacent gradations. The positive parameter γ is the scale parameter which characterizes the scale, and $\exp(\gamma)$ is the progression factor.

4 Relationship with SMART

In psycho-physical measurement, the ratios of audible sound and visible light intensities are usually recorded as differences on the deci-Bell scale. This means that not the ratio magnitudes themselves are considered, but their orders of magnitude. The observation suggested us to assume that a difference of grades in SMART represents the order of magnitude of a ratio of subjective values in the multiplicative AHP. In doing so, we obtain a simple straightforward relationship between the two multi-criteria methods, enabling us to carry out a cross-validation of the results. Both methods are now incorporated in the REMBRANDT system of L. Rog (Delft University of Technology) for Ratio Estimation in Magnitudes or deci-Bells to Rate Alternatives which are Non-DominaTed. Considering two alternatives A_j and A_k under a given criterion, with the respective grades g_j and g_k assigned to them, we take the quantity

$$r_{jk} = \exp(\gamma(g_j - g_k))$$

to represent the gradation of comparative judgement (normally, we use the grades 4,6,8,10 to designate poor, fair, good, and excellent performance, the symbol γ stands for the scale parameter). Hence, the grades assigned to the alternatives under the respective criteria can immediately be employed in the multiplicative AHP. The user can even work with the multiplicative AHP under some of the criteria, and with SMART under the remaining ones.

In the psycho-physical literature, the issue of how human beings judge the relationship between two stimuli was brought up a few decades ago. Torgerson (1961) observed that human beings estimate differences of subjective values when they are requested to express their judgement on a category scale with arithmetic progression, and they estimate ratios of subjective values when the proposed scale is geometric. Thus, they estimate the relationship as it is required in the experiment. Which of the two interpretations is correct, cannot empirically be decided because they are alternative ways of saying the same thing.

Torgerson's observation is easy to understand if we assume that the subjective stimulus values are not identically used in the two types of experiments. In the ratio experiment

with a geometric scale, human beings judge the ratio of two stimulus values. In the difference experiment with an arithmetic scale, they do not judge the ratio itself but its order of magnitude, which is essentially a logarithm of the ratio. Thus, ratio judgement is exponentially related to difference judgement (this was confirmed by psycho-physical research in the seventies and eighties, see Veit (1978) and Birnbaum (1982)). Moreover, the multiplicative AHP and SMART do the same thing albeit in alternative ways, and they are exponentially related.

5 Choice of scale-parameter values

We sketch human behaviour in various areas in order to explain the numerical values assigned to the scale parameter and henceforth to verbal statements such as weak, strict, strong, or very strong preference for A_j with respect to A_k. To our knowledge, this approach to explain the numerical values is new. First, we provide a heuristic introduction to illustrate the transition from car prices to the subjective judgements whereby cars are referred to as "cheap", "somewhat more expensive", "more expensive", or "much more expensive". In fact, we subdivide a given price range into a number of price categories (intervals) which are felt to be of the same order of magnitude, and we use the corresponding grid points (levels) to establish ratios of price increments (echelons) expressing what we mean by "somewhat more", "more", and "much more". Next, we show that human judgement leads in many unrelated areas (progression of historical periods and planning horizons, classification of nations according to size, perception of light and sound intensities) to the same categorization of intervals: there are roughly four major categories, the echelons constitute a sequence with geometric progression, and the progression factor is roughly 4.

We use these results in the REMBRANDT system to introduce a natural geometric scale for the quantification of verbal, comparative judgement: a scale with major as well as threshold echelons, and the progression factor 2. The scale parameter γ is accordingly set to the value $ln\ 2$. Sensitivity analysis with a short and a long geometric scale in the neighbourhood of the natural scale usually shows that the impact scores are rather stable. This is illustrated by the numerical example at the end of the presentation.

Lastly, we show that the relative importance of the criteria can also be established via the pairwise comparison of two alternatives. By this new approach, we found only one geometric scale to quantify the relative importance: a scale with major and threshold echelons, and with progression factor $\sqrt{2}$, so that the scale parameter γ is set to $ln\sqrt{2}$. Pairwise comparisons at the first and the second evaluation level will accordingly be rather similar, despite the conceptual differences. Moreover, by the uniqueness of the scale at the second level we can show that the rank order of the final scores does not depend on the geometric scales employed at the first level.

6 Epilogue

The AHP was intended to structure a decision process by the introduction of a hierarchy of evaluation levels, much higher than the two-level model considered in the present paper. Given the difficulties encountered in the aggregation step, a hierarchical structure with more than two levels should be thoroughly studied before it is launched in a practical environment. So far, we only formalized the concept of the relative importance of the criteria, via a model which is based on the pairwise comparison of alternatives. In a hierarchy of evaluation levels, we would run up against the relative importance of sub-criteria, sub-sub-criteria etc., concepts which are still undefined. The original version of the AHP disregards these questions, and constructs multi-level hierarchies as audaciously as it carries out the subsequent analysis. Such a top-down approach (see als Keeney and Raiffa (1976)) is in sharp contrast with the cautious bottom-up approach followed by the French school in multi-criteria analysis (Roy (1985), Schärlig (1985), Vincke (1989)). Cross-fertilization between the French and the American school in multi-criteria analysis has been meagre, however. Well-known Anglo-Saxon textbooks (von Winterfeldt and Edwards (1986), French (1988)) ignore the French school. None of the five textbooks just mentioned gives a thorough description of the AHP, by its world-wide popularity a prominent method in the American school. The relationship between the AHP and SMART, established by the author (1991), may enhance the power of the two methods, provided that they are jointly embedded in a flexible decision-support system.

References

[1] J. Barzilai, W.D. Cook, and B. Golani, Consistent Weights for Judgement Matrices of the Relative Importance of Alternatives. Operations Research Letters 6, 131-134, 1987.

[2] J. Barzilai and B. Golani, An Axiomatic Framework for Aggregating Weights and Weight-Ratio Matrices. Proceedings of the Second Int. Symp. on the AHP, Pittsburgh, Penn., 1991, pp. 59-70.

[3] V. Belton, A Comparison of the Analytic Hierarchy Process and a simple Multi Attribute Value Function. European Journal of Operational Research 26, 7-21, 1986.

[4] M.H. Birnbaum, Controversies in Psychological Measurement. In B. Wegener (ed.), Social Attitudes and Psycho-physical Measurement. Hillsdale, N.J., 1982, pp. 401-485.

[5] S. French, Decision Theory, an Introduction to the Mathematics of Rationality. Ellis Horwood, Chichester, 1988.

[6] R. Keeney and H. Raiffa, Decisions with Multiple Objectives: Preferences and Value Tradeoffs. Wiley, New York, 1976.

[7] F.A. Lootsma, Performance Evaluation of Nonlinear Optimization Methods via Multi-Criteria Analysis and via Linear Model Analysis. In M.J.D. Powell (ed.), Nonlinear Optimization 1981. Academic Press, London, 1982, pp. 419-453.

[8] F.A. Lootsma, Modélisation du Jugement Humain dans l'Analyse Multicritère au moyen de Comparaisons par Paires. R.A.I.R.O./Recherche Opérationnelle 21, 241-257, 1987.

[9] F.A. Lootsma, The French and the American School in Multi-Criteria Decision Analysis. R.A.I.R.O./Recherche Opérationnelle 24, 263-285, 1990. A short version appeared in A. Goicoechea, L. Duckstein, and S. Zionts (eds.), Multiple Criteria Decision Making, Springer, New York, 1992, pp. 252-268.

[10] F.A. Lootsma, Scale Sensitivity and Rank Preservation in a Multiplicative Variant of the AHP and SMART. Report 91-67, Faculty of Maths. and Infs., Delft University of Technology, Delft, The Netherlands, 1991.

[11] L.E. Marks, Sensory Processes, the New Psychophysics. Academic Press, New York, 1974.

[12] J.A. Michon, E.G.J. Eijkman, and L.F.W. de Klerk (eds.), Handboek der Psychonomie (in Dutch). Van Loghum Slaterus, Deventer, 1976.

[13] F.S. Roberts, Measurement Theory. Addison-Wesley, Reading, Mass., 1979.

[14] B. Roy, Méthodologie Multicritère d'Aide à la Décision. Economica, Collection Gestion, Paris, 1985.

[15] Th.L. Saaty, The Analytic Hierarchy Process, Planning, Priority Setting, Resource Allocation. McGraw-Hill, New York, 1980.

[16] A. Schärlig, Décider sur Plusieurs Critères. Presses Polytechniques Romandes, Lausannes, 1985.

[17] S.S. Stevens, On the Psychophysical Law. Psychological Review 64, 153-181, 1957.

[18] W.S. Torgerson, Distances and Ratios in Psycho-physical Scaling. Acta Psychologica XIX, 201-205, 1961.

[19] C.T. Veit, Ratio and Subtractive Processes in Psycho-physical Judgement. Journal of Experimental Psychology: General 107, 81-107, 1978.

[20] Ph. Vincke, L'Aide Multicritère à la Décision. Editions de l'Université de Bruxelles, Belgique, 1989.

[21] D. von Winterfeldt and W. Edwards, Decision Analysis and Behavioral Research. Cambridge University Press, Cambridge, UK, 1986.

[22] F. Zahedi, The Analytic Hierarchy Process - a Survey of the Method and its Applications. Interfaces 16, Nr. 4, 96-108, 1986.

[23] E. Zwicker, Psychoakustik. Springer, Berlin, 1982.

LAGRANGEAN RELAXATION AND OPTIMAL SOLUTIONS TO TIME WINDOW CONSTRAINED VEHICLE ROUTING PROBLEMS

Oli B.G. Madsen

IMSOR

The Institute of Mathematical Statistics
and Operations research
The Technical University of Denmark
DK 2800 Lyngby, Denmark
Phone: +45 42881433
Fax: +45 42881397
E-mail: ogm@imsor.dth.dk

INTRODUCTION

Vehicle routing problems with time windows have interested researchers and practitioners for some years. Time window constraints occur in for example newspaper delivery, delivery of fresh and frozen food, dial-a-ride services, and school bus systems. In those problems one or several time windows are connected to each customer imposing earliest and latest allowable start for delivery at the customer. Usually time windows are also connected to the depot.

The time windows may be hard meaning that a solution is considered to be infeasible if the time window constraints are not met. If the time windows are soft is is allowed to deliver to the customer outside the time window. However a penalty is then imposed. In this paper problems with one hard time window per customer will be considered.

Surveys on the literature on VRPTW are given in Solomon and

Desrosiers (9) and in Desrochers, Lenstra, Savelsbergh and Soumis (3). The exact approaches that the author is aware of can be divided in the following four classes:

1. Approaches based on dynamic programming. This line of research has been followed by Kolen, Rinnooy Kan and Trienekens (8) and can be regarded as an extension of the Christofides, Mingozzi and Toth (1) state space relaxation method. Problems with up to 14 customers have been solved to optimality.

2. Approaches based on column generation and set partitioning. In this class Desrochers, Desrosiers and Solomon (4) recently presented an exact method with the capability of solving 100-customer problems. The algorithm is based on a combination of LP relaxed set covering and column generation.

3. Lagrangean relaxation based methods. Madsen et al. (6,7) have applied various Lagrangean relaxation schemes to the VRPTW in order to produce lower bounds. They are currently capable of solving 105-customer problems to optimality using a combination of Lagrangean relaxation and Branch and bound.

4. An extension of Fisher's (5) exact algorithm for the classical vehicle routing problem to the case with time window constraints. The method is based on a K-Tree relaxation. Problems with up to 100 customers have been solved to optimality.

In this paper an exact solution method to the vehicle routing

problem with time windows (VRPTW) will be presented. The method is from class 3 and is based on a special version of Lagrangean relaxation in which variables are splitted into multiple copies. The purpose is to form a problem that can be separated into a number of subproblems with known usable structures.

The VRPTW is split into two types of subproblems. A semiassignment problem and a series of shortest path problems with time windows, capacity constraints, and thee possibility of containing negative cycles. The multiplier updating in the coordinating problem is done by subgradient optimization. Branch and bound is used to close the duality gap.

PROBLEM FORMULATION

The problem can be defined by the following parameters:

m = number of vehicles
n = number of customers, index 0 denotes the depot
Q_k = capacity of vehicle k
q_i = demand of customer i
c_{ij} = cost of travel from customer i to j
t_{ij} = time for travel from customer i to j
s_i = service time at customer i
e_i = earliest time allowed for starting delivery at customer i
u_i = latest time allowed for starting delivery at customer i
T = a scalar than the travel time of any feasible route

We are required to assign each customer to a vehicle and to sequence the set of customers assigned to each vehicle so as to minimize cost subject to vehicle capacity constraints and the requirement that the time to begin delivery at customers lies in the time interval prescribed. The decision variables are:

x_{ijk} = 1 if vehicle k travels directly from customer i to

customer j; 0 otherwise

y_{ik} = 1 if customer i is visited by vehicle k; 0 otherwise

t_i = the time to begin delivery at customer i

t_{0k} = departure time of vehicle k from the depot

t_{0k} = arrival time of vehicle k at the depot

In words the problem can now be formulated in the following way:

1. Minimize the total travel costs
2. If route k visits a point it has to leave the point again
3. Each route originates and terminates at the depot
4. The time to begin delivery at the customer shall be within the limits of the time window. The same applies for the depot.
5. If a vehicle travels directly from i to j then t_j should be compatible to t_i.
6. Each route's demand is within the capacity limit of the vehicle serving the route.
7. Each customer is visited exactly once.

If the connection between x_{ijk} and y_{ik} is relaxed the problem is for example decoupled in a semiassignment problem (containing the y_{ik}'s) and m shortest path problems with time windows and capacity constraints (SPTWCC) (containing x_{ijk}, t_i, t_{0k}, and t_{0k}). The semiassignment problem is easy to solve by inspection. The SPTWCC is a difficult and time consuming problem to solve. It is done by generalizing Desrochers and Soumis' Generalized Permanent Labeling Algorithm (2) by extending the three labels by a fourth one. The solution method is based on dynamic programming. The solution of SPTWCC may contain negative cycles limited by the time windows. The coordinating problem arising from the relaxation is solved iteratively by subgradient optimization. Due to the integrality of x and y there may be a gap so that only a lower bound for the VRPTW is obtained. If a gap is occuring branch and bound is used by fixing a y to one or zero. Then the subproblems are solved again.

COMPUTATIONAL RESULTS

The 100-point test problems developed by Solomon (8) were used as benchmark problems. Furthermore a 31-point problem (a reduced version of a Solomon problem) and a 105-point problem (a combination of two Solomon problems) were used. Also subsets of the Solomon problems consisting of 25 and 50 points were used. We solved one 105 point problem, 3 100 point problems, 10 50 point problems, one 31 point problem, and 22 25 point problems to optimality. The clustered problems were the easiest to solve, while random problems and mixed random and clustered problems were more diffucult. If the time windows were too wide the SPTWCC algorithm failed. This algorithm has a complexity depending on the square of the vehicle capacity and the square of the sum of the time window widths. The column generation algorithm mentioned under class 2 seems at present to be more effective. However there are some of the testproblems which can only be solved by the column generation algorithm, and some which can only be solved by the Lagrangean algorithm, and some which can not be solved at all. Therefore more research is needed within this subject area.

REFERENCES

1. N. Christofides, A. Mingozzi and P. Toth (1979). The vehicle routing problem. In Combinatorial Optimization by Christofides, Mingozzi, Toth and Sandi (eds.), John Wiley and Sons, New York, USA.
2. M. Desrochers and F. Soumis (1988). A generalized permanent labeling algorithm for the shortest path problem with time windows. INFOR 26, 191-212.
3. M. Desrochers, J.K. Lenstra, M.W.P. Savelsbergh and F. Soumis (1988). Vehicle routing with time windows: Optimization and approximation. In Vehicle Routing: Methods and Studies by Golden and Assad (eds.), North Holland, pp 65-84.
4. M. Desrochers, J. Desrosiers and M.M. Solomon (1992). A new optimization algorithm for the vehicle routing problem with time windows. Operations research 40, 342-354.

5. M.L. Fisher (1989). Optimal solution of vehicle routing problems and minimum K-trees. Working paper 89-12-13, Department of Decision Sciences, The Wharton School at the University of Pennsylvania, Philadelphia, USA.
6. K. Halse (1992). Modeling and solving complex vehicle routing problems. PhD dissertation from IMSOR, The Institute of Mathematical Statistics and Operations Research, The Technical University of Denmark, DK 2800 Lyngby, Denmark.
7. K. Jornsten, O.B.G. Madsen and B. Sørensen (1986). Exact solution of the vehicle routing and scheduling problem with time windows by variable splitting. Research report 5/1986, IMSOR, The Institute of Mathematical Statistics and Operations Research, The Technical University of Denmark, DK 2800 Lyngby, Denmark.
8. M.M. Solomon(1987). Algorithms for vehicle routing and scheduling problems with time window constraints. Operations Research 35, 254-265.
9. M.M. Solomon and J. Desrosiers (1988). Time window constrained routing and scheduling problems. Transportation Science 22, 1-13.

Recent exact and approximate algorithms for the Quadratic assignment problem

Bernard Mans, Thierry Mautor, Catherine Roucairol

INRIA
Domaine de Voluceau, Rocquencourt BP105
F-78153 LE CHESNAY Cedex, FRANCE

1.The Quadratic assignment problem

The QAP arises in facilities location and layout problems when for exemple, n facilities are to be assigned to n sites and when the interactions between the facilities depend upon their location. It can be formulated as follows, if $F=(f_{ij})$ is the matrix of flow between facilities i and j, $D=(d_{kl})$ the matrix of distances between locations k and l:

find a permutation p of the set $N = \{1, 2,...,n\}$
which minimizes the global cost function.

$$\min \text{Cost}(p) = \sum_i \sum_j f_{ij}\, d_{p(i)p(j)}$$

The QAP, known to be NP-complete in complexity, has shown itself to be a very difficult problem computationally.We will show that only problems of moderate size ($n \leq 20$) can be solved exactly.

2.New exact algorithms

There exists two approaches to solve exactly the QAP. The first one which consits to reformulate the problem as a linear mixed integer program and to solve it by cutting plane methods, has not been very successful in the past. It manages to solve exactly only problems of size up to eight (Kaufman and Broeckx[18], Balas and Mazzola[2], Bazaraa and Sherali[3]). The second one is based upon the concept of Branch and Bound (B&B) enumeration.
This approach yields better results so we will present the last recent B&B algorithms.

2.1.Lower bounds

To solve exactly QAPs the computation of the lower bound represents one of the main difficulties. Either the bound is too loose and the number of B&B nodes becomes too high, either the computational time to bound one node is prohibitive.
We will show that recent lower bounds based upon:

- eigenvalue approach (introduced by Finke,developped by Rendl [30])
- equivalent reformulations of the problem (Carraresi and Malucelli [11,12])
- reduction approach (introduced by Burkard [9] and Roucairol[31,32] , developped recently by Pardalos[14]),

improve of course, the results obtained with the oldest and commonly used bound, the Gilmore-Lawler bound[17, 20] which is based on the notion of ranked product.
But, this improvement remains low, and the average relative error (in comparison with the best solution) computed at a node of the B&B tree is quite important and decreases slowly when considered at a node of higher level.

α = Ameliorative Rate = (New bound - GLB) / (Best Value - GLB)

Bounds :	Finke,Burk.,Rendl	Carraresi,Malucelli	Pardalos	
Complexity	$o(n^3)$	$o(kn^5)$	$o(n^3)$	$o(n^6)$
Size				
5	0	0		+100
6	-300	0		+32
7	-63	+27		
8	-43	+7		+12
12	+2	+2		+11
15	+14	+5		+12
20	+41	+2		+8

Table1. Ameliorative rates of other bounds in comparison with GLB

Until now, it is very difficult to measure the effectiveness of these lower bound improvements in a B&B algorithm; for most of them the results have been announced in a forthcoming paper...

2.2.Branching scheme and reduction tests

As it is very difficult to obtain efficiently good lower bounds, an other approach consists to still use GLB bound but to concentrate the effort on an other way to reduce enumeration. Thus, we have proposed [23] a new exact algorithm which has been able to solve, for the first time exactly, problems of size up to twenty in quite a reasonable time. The idea is to define a new branching

scheme together with appropriate branching rules and reduction tests. We will first describe one of the reduction test: the *symmetry test.*

We use symmetric properties of the real word application, like symmetries in the implementation sites. On Nugent's problems [27], for instance, the sites are on a grid and the distances are rectangular ones. Therefore, symmetrical equivalent solutions go by group of four and will have the same cost.

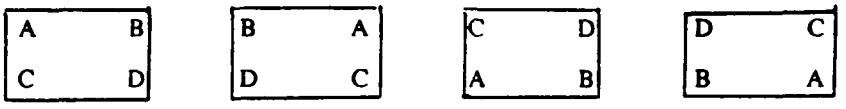


Figure1.

Our B&B method will not create, study and bound equivalent nodes with symmetrical solutions in different branches of the tree. In order to test quickly and efficiently the property of symmetry between two sites , we will give the following definition :

Two sites i and j are symmetrically equivalent if

1.i and j belong to the same equivalence class:
their vector of distances with other available sites,where the components are ranked increasingly are equal,

2.their vector of repartition in the different equivalence classes of all available sites being at a distance d are equal .

This test is easily computed and will produce an important decrease of the size of the B&B to search. As we will show further, heuristics are able to provide good solutions (with a cost close to the value of the optimal solution). Based upon theses remarks, our B&B algorithm uses a *depth first search strategy* to examine the nodes which probably belong to the critical tree (nodes whose evaluation is lower than the value of the optimal solution). It also uses at a B&B node *a reduction test based upon the search gap* (the difference between the value of the best known solution and the value of the lower bound of solutions belonging to this node): each assignment of a facility to a site which has a cost at least greater than the gap is forbidden.

The *branching scheme* is polytomic: the facilty with the higher number of forbidden assignments is placed on all available sites..

2.3.Results of sequential and parallel B&B algorithms

Firstly, we present the results obtained by the fastest sequential B&B codes available: Burkard and Derigs [9], Pardalos [14], Mautor and Roucairol[23].

The test data used are a classical benchmark for QAP, the first ones as the second, assign facilities on a grid. As the distance between two adjacent sites is equal to one, these problems have a very special structure and are not representative of a general QAP (as pointed out at the QAP's day organized in

Bologne, september 92). But, waiting a QAP test problems library, it is the only present way to compare results. Our algorithm is the fastest and obtains the best results even on problems on which symmetries cannot be pointed out (Nugent 7, Elshafei 19).

The optimality of solution for famous problem of size less than 20 (size 19-Elshafei [15]., Size 20-Armour and Buffa [1].) is proved.

Problem	Nugent	Nugent	Nugent	Nugent	Elshafei	Armour
Size	8	12	15	16	19	20
Sites plant	Grid	Grid	Grid	Square	-	Grid
	4 * 2	4*3	53	4*4		5 * 4
Best value	214	578	1150	1550	17.212.548	110.030
ALGORITHMS						
Maut., Rouc.						
Cray 2	0.04	3.4	121	969	1.4	1189
Burkard/ Cyber 76	0.26	46.7	2947			
Burkard/ Cray 2	0.11	24.2	1290	NEVER		
Pardalos / IBM 3090	0.29	34.1	2005			
				PROVED		

The recent developpement both of commercially available multiprocessors and of theoritical analysis of parallelization of B&B algorithms suggested that such parallel algorithms may be a fruitful area of investigation for QAP.

Only few experiments have been made: two by our team in 87 (Roucairol [33], and 92 Mans, Mautor,Roucairol [22]), one by Pardalos in 89[14]. They consist to parallelize on synchronous multiprocessors machine a B&B algorithm using GLB lower bound. Our last proposition is of course, the parallelization of the new B&B algorithm presented in the previous section [23]. From the point of view of parallelism, these B&B algorithms differ by the way they allocate tasks to the different processors.

In Roucairol's first algorithm, a global heap is used to memorize the B&B nodes to be explored by the processors. A task consists to explore a node (creation and evaluation of successor node); each

processor will require an access to the shared data structure to obtain such a node or to insert new generated nodes. In order to limit global accesses, Crouse and Pardalos [14], build initialy several heaps (initiated with one node) in the shared memory so that each processor selects one and can fully explored it locally Since the heaps are smaller the maximum time that a processor could be idle (waiting for a task) is smaller.

Our parallelization uses a different way to distribute the work to processors through the notion of *feeding tree*.

The feeding tree is the uppert part of the B&B tree developped until depth (or level) i; the leaves of the feeding tree are the roots of the subtrees allocated to the processors, the tree stumps.

The first free processor initializes the left part of the feeding tree: the leftest node at the chosen depth i is generated by successives branching.

Only informations about the path from the root to the last allocated nodes (facilities assigned to reach this depth and set of the remaining available locations) have to be kept in the global memory.

Each idle processor will access to this shared structure, completes the exploration of a B&B subtree which has, as root, the feeding tree stumps, and a maximal height of (n-i).

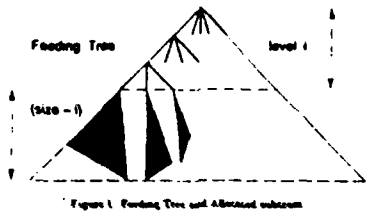


Figure 1: Feeding Tree and Allocated subtrees

Differents results obtained are resumed in the tables below.

Algorithm	Machine	# procs	Nugent size 12	Nugent size 15	Nugent size 16	Elshafei size 19	Scriabin-Vergin size 20
Optimal solution			578	1150	1550	17,212,548	110,030
Burkard	Cray 2	1	24	1290	not proved	not proved	not proved
Pardalos	IBM 3090	1	34	2005	not proved	not proved	not proved
Pardalos	IBM 3090	4	10	out of space	not proved	not proved	not proved
Mans and al.	Cray 2	1	2.68	109	969	1.04	1189
Mans and al.	Cray 2	4	0.99	28	436	0.68	560
Mans and al.	Cray YMP	1	not tested	62	not tested	not tested	not tested
Mans and al.	Cray YMP	6	not tested	11	not tested	not tested	not tested

Table 1: Comparison of Running Times (in Seconds).

For our parallel algorithm, the parameter i, called the shared level, involved in the definition of the feeding tree, allows to tune granularity and load balancing between tasks.

3.New heuristics

Of course, heuristics have been introduced to solve larger problems than those solvable by exact approaches. The average results of the earliest heuristic solution methods:

-construction methods (they reach iteratively by locating one or more facilities at each step a complete assignment),

- approximate exact methods (they reduce B&B search to obtain only good solutions),

- exchange methods (they improve the cost of a complete assignment by interchanging the locations of several facilities),

were rather good but on some instances, their results could be very bad. Thus, the most recent approximate methods use sophisticated meta-heuristics like:

- simulated annealing (Burkard and Rendl [10]., Lutton and Bonomi [21]., Wilhem and Ward [37].)

- tabu search (Skorin-Kapov [34]., Taillard [35].),

- evolution strategy (genetic algorithms Brown et al.[5]., Mulhenbein [24]., ant system, neural networks...)

We must say that the best solutions is always found by these algorithms on classical benchmark of test problems for size n less than twenty.

As previously argued, these problems have a very special structure; a lot of optimal or near optimal solutions exist. It has been pointed out that any approximate method which is able to focus sometimes around the current best solution and explore new region, could perform very well.

We will briefly report some first experiments we made, in the context of QAP, with massively parallel algorithms implemented on the Connection machine CM-2.

This results are now encouraging but not yet comparable with other results from literature. Mainly because until now we implemented a sketch of Tabu search which does not contains all the refinements. We will just indicate here the main characteristics of this approach.

Our idea was to use more than n^2 processors when n is the size of the QAP problem. So, two tasks will run in parallel: the first one follows an intensification strategy whereas the second performs a diversification phase. They will synchronize just to exchange an improved solution before a new iteration.

4.Conclusion

Assinment problem with quadratic objective function remains very hard to solve. Even if we have proved that good branching scheme and branching rules drastically reduce the number of nodes of the search tree, a sharp lower bound is still needed to construct efficient B&B and thus, to solve

exactly the QAP. Meta-heuristics seems to give very good results but a QAP library of new test problems must be constructed to measure correctly their effectiveness.

References

[1] G. Armour and E. Buffa, A heuristic algorithm and simulation approach to the relative allocation of facilities, Management Science 9 (1963) 294-309.

[2] E. Balas and J.B. Mazzola, Quadratic 0-1 Programming by a new linearization, Presented at the TIMS/ORSA Meeting, Washington D.C (1980).

[3] M. S. Bazaraa and O. Kirca, A branch and bound based heuristic for solving the quadratic assignment problem, Naval Research Logistics Quarterly 30 (1983) 287-304.

[4] M.S. Bazaraa and M.D. Sherali, Benders' partitioning scheme applied to a new formulation of the Quadratic Assignment Problem, Naval Research Logistics Quarterly 27 (1980) 29-41.

[5] D. Brown, C. Huntley and A. Spillane, A parallel genetic heuristic for the quadratic assignment problem, Proceedings on the 3rd conference on Genetic Algorithms, Arlington (1989) 406-415.

[6] E. Buffa, G. Armour and T. Vollmann, Allocating facilities with CRAFT, Harvard Business Review 42 (1962) 136-15

[7] R. Burkard, Some recent advances in quadratic assignment problems, Proceedings of the Congress on Math. Programming, Rio de Janeiro (1984) 53-68.

[8] R. Burkard and T. Bönniger, A heuristic for quadratic boolean programs with applications to Quadratic Assignment Problems, European Journal of Operational Reseach 13 (1983) 374-386.

[9] R.Burkard and U.Derigs; *Assignment and matching problems: solution methods with Fortan program.* SpringerVerlag,1980.

[10] R. Burkard and F. Rendl, A thermodynamically motivated simulation procedure for combinatorial optimization problems, European Journal of Operational Research 17 (1984) 169-174.

[11] P. Carraresi and F. Malucelli, A new lower bound for the quadratic assignment problem, Operations Research 40 (1992) S22-S27.

[12] P. Carraresi and F. Malucelli, A new branch and bound algorithm for the Quadratic Assignment Problem, Presented at the 3rd ECCO Meeting, Barcelona (1990).

[13] J.Chakrapani J.Skorin-Kapov, Massively parallel Tabu search for the quadratic assignment problem, Annals of Operations Research, to appear;

[14] J. Crouse and P. Pardalos, A parallel algorithm for the quadratic assignment problem, presented at the 'Proc. of Supercomputing 89' conference, ACM Press (1989) 351-360.

[15] A. Elshafei, Hospital Lay-out as a quadratic assignment problem, Operational Research Quarterly 28 (1977) 167-179.

[16] G. Finke, R. Burkard and F. Rendl, Quadratic assignment problems, Annals of Discrete Math. 31 (1987) 61-82.

[17] P. Gilmore, Optimal and suboptimal algorithms for the quadratic assignment problem, SIAM Journal on Applied Math. 10 (1962) 305-313.

[18] L. Kaufman and F. Broeckx, An algorithm for the quadratic assignment problem using Benders' decomposition, European Journal of Operational Research 2 (1978) 204-211.

[19] T. Koopmans and M. Beckmann, Assignment problems and the location of economic activities, Econometrica 25 (1957) 53-76.

[20] E. Lawler, The quadratic assignment problem, Management Science 9 (1963) 586-599.

[21] J.L. Lutton and E. Bonomi, The asymptotic behaviour of quadratic sum assignment problems : a statistical mechanics approach, European Journal of Operational Research 26 (1986) 295-300

[22] B.Mans,T.Mautor,C.Roucairol, A parallel depth first search Branch and Bound algorithm for the quadrztic assignment problem, Technical report INRIA,.1992.

[23] T.Mautor,C.Roucairol, A new exact algorithm for the Quadratic assignment problem, Discrete Applied Mathematics ,to appear .

[24] H.Mulhenbein,.Parallel genetic algorithms, population genetics and combinatorial optimization in Parallelism, learning, evolution WOPPLOT89, Springer-Verlag,1989.]

[25] Li, P.Pardalos, Ramakrishnan, Resende, A lower bound for the Quadratic assignment problem, Technical report, Pennsylvannia Univ., 1992.

[26] V.Nissen, Solving the quadratic asignment problem with clues from nature, Technical report Universitat Gottingen, Germany, presented at the QAP-worshop, Bologna,Italy, september 92.

[27] C. Nugent, T. Vollmann and J. Ruml, An experimental comparison of techniques for the assignment of facilities to locations, Operations Research 16 (1968) 150-173.

[28] P. Pardalos and K. Murphy, A polynomial-time approximation algorithm for the quadratic assignment problem, Technical Report, The Pennsylvania State University (1990).

[29] F. Rendl, Ranking scalar products to improve bounds for the quadratic assignment problem, European Journal of Operational Research 20 (1985) 363-372.

[30] F. Rendl and H. Wolkowicz, Applications of parametric programming and eigenvalue maximization to the quadratic assignment problem, Math. Programming 53 (1992) 63-78.

[31] C. Roucairol, A reduction method for quadratic assignment problems, Operations Research Verfahren 32 (1979) 183-187.

[32] C. Roucairol, Un nouvel algorithme pour le problème d'affectation quadratique, RAIRO 13 (1979) 275-301.

[33] C. Roucairol, A parallel branch and bound algorithm for the quadratic assignment problem, Discrete Applied Mathematics 18 (1987) 211-225.

[34] J. Skorin-Kapov, Tabu search applied to the quadratic assignment problem, ORSA Journal of Computing 2 (1990) 33-45.

[35] E. Taillard, Robust tabu search for the quadratic assignment problem, Parallel Computing 17 (1991) 443-455.

[36] J.Wang, A parallel distributed processor for the quadratic assignment problemin Poceedings of the INNC90,vol 1,278_281,France,1990.

[37] M. Wilhelm and T. Ward, Solving quadratic assignment problems by simulated annealing, IIE Transactions 19 (1987) 107-119.

A Numerically Exact Implementation of the Simplex Method

István Maros and Csaba Mészáros

Computer and Automation Institute
Hungarian Academy of Sciences
P.O. Box 63, H-1518 Budapest, Hungary
E-mail: h107mar@huella.bitnet and h4225mes@huella.bitnet

Extended Abstract

1. Introduction

Numerically difficult linear programming (LP) problems have always been a challenge for developers of simplex based LP codes. Considerable effort has been spent on improving the numerical behavior and the robustness of such systems. Early it became clear that the additive floating-point arithmetic operations (addition, multiplication) are the sources of the problems, since their relative error can be any big in the framework of the finite and normalized number representation of floating-point numbers. The approach to overcome the difficulties came from two directions. The first one tried to analyze and modify the way these operations are performed (c.f. [OH68], [BGHR77], [Mar89]). The second one worked out numerical procedures that posses proven better numerical characteristics. In this respect a very important step was the introduction of the *LU* factorization of the inverse of the basis. Several variants of it have been developed with different pivot selection strategies (c.f. [BGHR77], [SS90]).

2. Additive operations in the simplex method

Additive operations in the simplex method occur during *FTRAN* and *BTRAN* ([OH68]) type operations. This is true for the inversion/factorization, as well. In FTRAN, the operation is of $a = b+c$ type, while in BTRAN the typical operation is the inner product of two vectors where a number of additions take place. Numerical problems may cause either the creation of a small number (***white noise***) in place of a zero (type-1 error), or creation of a zero in place of some significant value (type-2 error).

A simplified example for a type-1 error can be a computation like $a = 1-3*b$, where a was computed earlier as $b = 1/3$. Here a will not necessarily be zero. In subsequent transformations this white noise can grow beyond the magnitude of the pivot tolerance and can cause serious problems.

It can be equally problematic if an algebraic nonzero element appears to be zero (type-2 error). This always happens if the relation of b and c is such that

$|b|\varepsilon_a > |c|$, where ε_a is the relative accuracy of the number representation, and $a = (b + c) - b$ is to be computed. Here, we will get $a = 0$ instead of $a = c$. Type–2 error is a consequence of the lack of associativity of floating point additive operations.

To reduce the probability of the occurence of type–1 errors, a remarkable step was made by ORCHARD–HAYS [OH68] who introduced the *relative tolerance* ($\varepsilon_{\text{rel}} > 0$, small) for additive operations within the simplex method. Verbally, its use can be described as follows. If there are too many significant digits lost during an additive operation, or in other words, the magnitude of the result of an additive operation with two operands is much smaller (ε_{rel} times less) than the larger magnitude of the operands then the result is set to zero.

More formally, the algebraic expression $a = b + c$ is numerically evaluated so that if

$$\frac{|a|}{\max\{|b|, |c|\}} < \varepsilon_{\text{rel}}, \tag{2.1}$$

then $a = 0$ is set. Clearly, the choice of ε_{rel} is critical. Typical values for ε_{rel}, in the case of double precision (8–byte) real numbers, are between 10^{-12} and 10^{-10}. If a high level programming language (Fortran, C) is used, checking (2.1) is costly, because of the division or the equivalent multiplication. At the same time, its use in FTRAN is unavoidable unless some other technique is applied.

Benichou et al. ([BGHR77]) selected an alternative method in FTRAN. They set an α_i value in a transformed α vector to zero if $|\alpha_i| < \varepsilon u$ with an appropriate ε, where u is the absolute value of the element with largest magnitude created during the $\alpha = B^{-1}a$ operation.

The main purpose of the above techniques is to help distinguish between white noise and zero to avoid the selection of a zero pivot in the transformed vector α.

During BTRAN, inner products of vectors are computed. Inner products are well known for their bad numerical characteristics. Their computation requires an accumulator collecting the component wise products. This computation is very much prone to type–2 error. Using (2.1) cannot prevent this. Among several possibilities, a relatively simple idea proved to be quite efficient. Namely, an inner product $s = p'a$ can be computed in such a way that the negative and positive terms are accumulated separately and added together at the and, thus giving chance to type–2 error only once per inner product:

$$\sum_i p_i a_i = \sum_{p_i a_i > 0} p_i a_i + \sum_{p_i a_i < 0} p_i a_i. \tag{2.2}$$

Clearly, the above and some other techniques that do not go beyond the traditional number representation and operations will be never be able to guarantee the exact outcome of the critical operations in the simplex method.

3. An accurate floating–point arithmetic

The idea of (2.2) can be refined. If we divide the possible range of the $p_i a_i$ values into N consecutive intervals (buckets) defined by points $t_0, t_1, \ldots, t_N$ then the inner product can be computed in the following way:

$$\sum_i p_i a_i = \underbrace{\sum_{t_0 \le p_i a_i < t_1} p_i a_i}_{bucket_1} + \underbrace{\sum_{t_1 \le p_i a_i < t_2} p_i a_i}_{bucket_2} + \ldots + \underbrace{\sum_{t_{N-1} \le p_i a_i \le t_N} p_i a_i}_{bucket_N} \qquad (3.1)$$

Here, the choice of $t_0, t_1, \ldots, t_N$ can control the generation and propagation of error. Addition of values falling into the same bucket will have smaller error. Addition of the contents of the buckets must be made in increasing bucket order to help reduce problems due to the lack of associativity. The implementation of this idea requires N accumulators and additional logic, while it still does not guarantee full accuracy in all cases.

To restore associativity, we propose a further refinement of (3.1). Now we assume that the exponents of the terms, intermediate, and final results on the left hand side of (3.1) taken in any order fall into the $[e_l, e_u]$ interval. In such a case one single *super register* will be able to accumulate the sum of the terms with *full accuracy* if it is large enough.

The idea of our super register (SR) can be sketched as follows. The length L of SR is defined in units of 4 Bytes. One unit is reserved for overflow. L is a parameter that can be assigned different values in advance depending on the estimated range of the values. SR is divided into the following parts:

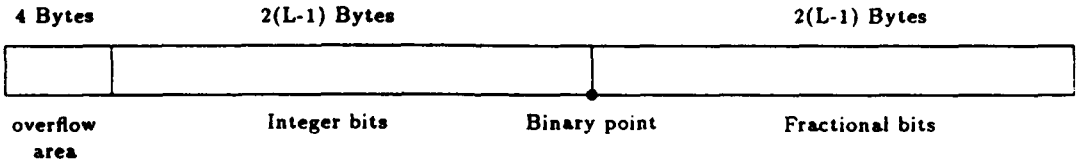


FIG. 1. The Super Register

SR has been designed to accumulate inner products. Multiplication is performed by the numeric coprocessor and the result is added to SR. For the case when SR turns out to be too small, a 10–byte long register (ER, not shown in Fig. 1.) is added to it. If the value to be added to SR falls outside the range of SR, ER will be used. Of course, in such a case the accuracy will be reduced to that of the traditional representation.

Clearly, the maintenance and operations with SR practically can not be achieved in a high level language. Therefore, we decided to use 386/486 based

PCs as target machines and assembly language. We defined three operations with SR: (1) `aclear` to clear SR, (2) `aadd(a,b)` to add the product of `a` and `b` to SR, and (3) `aread(c)` to retrieve the contents of SR and store it in `c` as an 8-Byte double precision number. These operations can be activated by subroutine calls from a Fortran program. Details of implementation are omitted here.

At the beginning we had to answer the very important question: Will the operations be fast enough to make the whole idea workable? To answer it, we made test runs on a 486/33 Mhz (256 K on-board Cache) PC with $L = 16$ to compare the new operations with the corresponding traditional ones using some accuracy control technique. This value of L resulted in a SR with exponent range of $\approx$ [-76,76], and with a resolution of 10^{-76} throughout the whole range. The following table shows execution times in micro seconds. For the new operations the overhead of subroutine calls is included, but we also give timing for subroutine calls with 0, 1, and 2 parameters.

Operation	Time
aclear	3.90
aadd	5.43
aread	8.79
call-0	2.75
call-1	2.74
call-2	2.75
(2.1)	7.03
(2.2)	4.17

The above values were obtained as averages of 10^6 attempts. Here, call-0, call-1, and call-2 mean the overhead of subroutine calls with 0, 1, 2 parameters, respectively.

In another series of experiments we computed 1,000 times inner product ($s = p'a$) of two vectors with 1,000 nonzero elements each. The next table gives average times (in microseconds) of one $s = p'a$ inner product.

Method	Time
(2.1)	7250
(2.2)	3410
SR	6540

The conclusion of these figures is that the use of the super register will not necessarily slow down the computations in the simplex method, especially if the computer is equipped with on-board cache memory.

4. Accurate arithmetic in the simplex method

Though the size of one super register is negligible, the memory requirement of vectors of super registers can be prohibitivly large. Therefore, we attempted to use one single super register throughout the simplex method and carry out all the critical floating point operations with this register. This idea entails that the operations are to be reorganized to become inner products, when possible.

In the case of the LU form of the basis (both L and U are stored colum wise) the critical operations are *inversion (factorization), BTRAN, and FTRAN.* In addition to them, after inversion and before recomputing the basic solution the right–hand–side is to be adjusted to account for nonbasic variables at upper bound and for super basic variables:

$$\bar{b} = b - \sum_{j \in J} a_j u_j, \tag{4.1}$$

where b is the original right–hand–side vector, a_j is column vector of variable x_j, u_j is its current nonbasic value, and J is the index set of nonzero nonbasic variables. This operation can also be a source of numerical problems.

In BTRAN the operations are originally inner produts, therefore nothing is to be reorganized.

Regarding the nature of operations, the LU factorization and FTRAN are similar, namely, they both require FTRANs.

After factorization the basis is written as $B = LU$, with $B^{-1} = U^{-1}L^{-1}$. The factorization itself can be carried out using one SR with heavy logic but without any compromise. The details of it are omitted here.

During simplex iterations the situation is a bit more complicated. If k basis changes have been made after factorization then $B_k^{-1} = E_k \dots E_1 U^{-1} L^{-1}$, where E_i is an elementary transformation matrix. When this formation is used for FTRAN to compute $\alpha = B^{-1}a = E_k \dots E_1 U^{-1} L^{-1} a$ the transformation with L^{-1} and U^{-1} can be performed after one another with the SR using a dynamic linked list (DLL). For the transformation with $E_k \dots E_1$, we have two options: (1) this operation is carried out in some traditional way, e.g. (2.1), (2) at the expense of some additional administration and operations the use of SR can be extended to this case.

The additional complications require additional memory, since DLL needs integer arrays of size m (the number of constraints in the LP problem) which can be too much in certain cases.

5. Computational experiences

To test the above ideas, we included the super register technique in our MILP linear programming optimizer [Mar91]. Presently (November, 1992), this implementation is temporary, not full–scale, and experimental. "Not full–scale" means

that our first version uses alternative (1) of the previous section which is clearly the simplest and the poorest version. In this way, the present implementation can be considered a hybrid one.

In addition to this, it important to note that the code is not tuned yet, and a number of measuring instructions are also present influencing the actual performance.

The purpose of tests was twofold: (1) to see the extent of improvement in numerical accuracy, (2) to get an idea about the speed of the new technique in comparison with the earlier, "traditional" version. MILP originally uses (2.1) type relative zeroing.

For testing we used some numerically nontrivial problems from NETLIB [Gay85]. A smaller representative of this category of problems is `GROW7`.

We made a special test with `GROW7`. First, we used the 80 bit extended accuracy of the arithmetic coprocessor, and applied two alternative ways to compute FTRAN. The only difference was the order how the inner produts were computed. At iteration 371 this resulted in a different basis change. One of them was a wrong one due to accumulated errors. The correction of this error took several iterations after the next factorization. When the SR technique was used with $L = 16$, the basis changes were not influenced by different ways of FTRAN, showing that associativity is better achieved with the present environment of SR.

In the table below we summarize our experiences with the original and SR version of MILP, where the SR version is a temporary one as described above. The old version of MILP uses the product form of the inverse. Programming language is Fortran77, program was compile by Lahey F77L-EM/32 V4.0. Solution times of a 486 PC are given seconds.

Problem	Version	Iter.	Time	Remark
GROW7	Old	526	30.04	Minor numerical troubles
GROW7	SR	538	40.59	—
GROW15	Old	2625	340.30	* Num. troubles, solution abandoned
GROW15	SR	893	155.16	One tiny corrected inaccuracy
GROW22	Old	1753	397.66	* Serious troubles, solution abandoned
GROW22	SR	1400	446.71	Few small corrected inaccuracies
STAIR	Old	752	127.81	Opt. sol. achieved after numerical troubles
STAIR	SR	543	123.08	—
PILOT4	Old	775	107.77	* Num. troubles, solution abandoned
PILOT4	SR	1049	267.54	—

The occasional minor troubles with the SR version mean that after factorization very small deterioration in the objective value or feasibility might have

occured. The reason for that is that operation (4.1) is not performed with SR yet.

The brief message of the above table is that the iteration speed does not slow down seriously by using SR, while the accuracy expectations tend to be fulfilled. In the "non–temporary" version the speed of factorization will be doubled, while the implementation of additional refinements will have a slight counter–effect. After all, the complete implementation of the idea of SR will tell us the final answer to its applicability and usefulness.

It is worth mentioning here that we tested SR also in the framework of our interior point algorithm. The algorithm is of primal affine scaling type. SR was used in factorization only. We observed an increase in accuracy of half to 2 orders of magnitude on problems GROW7, GROW22, and PILOT4.

References

[BGHR77] Benichou, M., Gauthier, J.M., Hentges, G., Ribiere, G., The efficient solution of large scale linear programming problems, *Mathematical Programming*, 13 (3) (1977) pp. 280-322.

[Gay85] Gay, D.M., Electronic mail distribution of linear programming test problems, *COAL Newsletter*, 1985, December (13), pp. 10–12.

[Mar89] Maros, I., A Three–Parameter–Arithmetic (TPA), MTA SZTAKI Working Paper, MO/85, April, 1989, 12 pages.

[Mar91] Maros, I., MILP linear programming optimizer for personal computers under DOS, Research Report, MTA SZTAKI, Budapest, 41/1991, 16 pages.

[OH68] Orchard-Hays, W., "Advanced Linear Programming Computing Techniques", McGraw Hill, 1968.

[SS90] Suhl, Uwe, H., Suhl, Leena, M., Computing Sparse LU Factorization for Large–Scale Linear Programming Bases, *ORSA Journal on Computing*, Vol. 2, No. 4, Fall 1990, p. 325–335.

THE BOTTLENECK

GENERALIZED ASSIGNMENT PROBLEM

Silvano Martello, *Dipartimento di Informatica, Università di Torino, Italy*

Paolo Toth, *DEIS, Università di Bologna, Italy.*

1. Introduction

Given n *items* and m *units*, the *penalty*, p_{ij}, and the *resource requirement*, r_{ij}, corresponding to the assignment of item j to unit i ($j = 1,\cdots,n; i = 1,\cdots,m$), and the amount of resource a_i available at unit i ($i = 1,\cdots,m$), the *Bottleneck Generalized Assignment Problem* (BGAP) is to assign each item to one unit so that the total resource requirement for any unit does not exceed its availability and the maximum penalty incurred is minimized. We will assume in the following that all numerical input data (p_{ij}, r_{ij}, a_i) are non-negative integers. By introducing binary variables x_{ij}, with

$$x_{ij} = \begin{cases} 1 & \text{if item } j \text{ is assigned to unit } i; \\ 0 & \text{otherwise,} \end{cases}$$

the problem can be formulated as

$$\text{minimize} \quad z = \max_{i,j}\{p_{ij}x_{ij}\} \tag{1}$$

$$\text{subject to} \quad \sum_{j=1}^{n} r_{ij}x_{ij} \le a_i, \qquad i \in M = \{1,\ldots,m\}; \tag{2}$$

$$\sum_{i=1}^{m} x_{ij} = 1, \qquad j \in N = \{1,\ldots,n\}; \tag{3}$$

$$x_{ij} \in \{0,1\}, \qquad i \in M,\ j \in N. \tag{4}$$

The problem has applications in the fields of scheduling and allocation, among others. Suppose for example that each of n urban areas has to be served by one of m emergency centres. Let p_{ij} be the travel time between centre i and area j, r_{ij} the expected workload for centre i if area j is allocated to it, and a_i the maximum workload that centre i can support. The solution to BGAP will then give the feasible solution minimizing the worst-case travel time between an area and the associated centre.

BGAP is the min-max version of the well-known (min-sum) *Generalized Assignment Problem* (GAP), given by

$$\text{minimize} \quad \sum_{i=1}^{m} \sum_{j=1}^{n} p_{ij} x_{ij}$$

$$\text{subject to} \quad (2), (3), (4).$$

It is known that GAP is NP-hard in the strong sense, since even its feasibility question is so (see, e.g., Martello & Toth (1990), Ch.7). Hence the same results apply to BGAP.

Several contributions to the solution of GAP can be found in the literature (Ross & Soland(1975), Martello & Toth(1981,1990), Fisher, Jaikumar & Van Wassenhove(1986), Jörnsten & Näsberg(1986), Guignard & Rosenwein (1989), among others). Mathematical models and a technique for transforming BGAP into GAP have been given by Mazzola & Neebe(1988); to our knowledge no other result has been published in the literature.

In the next section we introduce lower bounds for the problem, which require the solution of bottleneck knapsack problems. Approximate algorithms are examined in Section 3. In Section 4 we describe a branch-and-bound algorithm for exact solution of the problem, and in Section 5 examine its computational behaviour.

2. Relaxations and Lower Bounds

We consider lower bounds obtained by relaxing constraints (2), either directly (by decreasing the resource requirements) or through surrogate techniques. In any case, we never allow an item j to be assigned to a unit i if, in the non-relaxed instance, $r_{ij} > a_i$.

Relaxation of the Resource Requirements

An immediate lower bound can be obtained by eliminating constraints (2). It is then evident that the resulting problem can be exactly solved, in $O(nm)$ time, by determinig

$$i_1(j) = \arg \min_i \{p_{ij} : r_{ij} \leq a_i\} \quad (j \in N)$$

and computing the lower bound value

$$L_0 = \max_j \{p_{i_1(j),j}\}.$$

If the corresponding solution (obtained by assigning each item j to unit $i_1(j)$) satisfies constraints (2), then this is clearly the optimum. Otherwise, a better bound L_1 can be obtained by imposing one of the violated constraints (as shown in Martello & Toth (1991)). The time complexity for the computation of L_0 and L_1 is $O(nm)$.

Surrogate Relaxation

For a given vector (π_i) of non-negative multipliers, we define the *surrogate relaxation* of BGAP, $S(BGAP, \pi)$, as

$$\text{minimize} \quad \hat{z}(\pi) = \max_{i,j} \{p_{ij} x_{ij}\}$$

$$\text{subject to} \quad \sum_{i=1}^{m} \pi_i \sum_{j=1}^{n} r_{ij} x_{ij} \leq \sum_{i=1}^{m} \pi_i a_i,$$

$$(3), (4).$$

For any non-negative vector (π_i),

$$L_2(\pi) = \hat{z}(\pi)$$

is then a lower bound for BGAP. It is proved in Martello & Toth (1991) that, for any vector (π_i) of multipliers, lower bound $L_2(\pi)$ can be computed in $O(nm)$ time.

3. Approximation Algorithms

A feasible solution to BGAP of value not greater than a given threshold ϑ can be heuristically found through a procedure which considers the items according to decreasing values of the difference between second smallest and smallest resource requirement for a feasible assignment, and assigns the considered item to the unit having the smallest resource requirement. If an item is found for which no feasible assignment is possible, the procedure returns no solution; otherwise it returns the feasible solution found. This procedure, which can be implemented to run in $O(nm \log m + n^2)$ time, can be used to determine an approximate solution to BGAP by searching for the lowest value ϑ for which a feasible solution is returned. If this is done through binary search, the overall time complexity of the resulting approximation algorithm is $O(nm \log m + \beta(nm + n^2))$, where β denotes the number of bits required to encode $\max_{i,j}\{p_{ij}\}$.

Other approximation algorithms are described in Martello & Toth (1991).

4. A Branch-and-Bound Algorithm

The results of the previous sections have been used to obtain a branch-and-bound algorithm for the exact solution of BGAP.

Branching Scheme

The algorithm consists of a depth-first search in which, at each level, an item j^* is selected for branching and assigned, in turn, to all feasible units.

The branching item is selected according to the following criterion. Let z denote the best incumbent solution value, L the current lower bound value, $\overline{a}_i$ $(i = 1, \ldots, m)$ the amount of resource currently available for unit i, and U the set of currently unassigned items. For each $j \in U$, $M_j = \{i \in M : r_{ij} \leq \overline{a}_i \text{ and } p_{ij} \leq L\}$ is the set of units to which j can be assigned without increasing the lower bound. Hence

$$r_j = \frac{\sum_{i \in M_j} (r_{ij}/\overline{a}_i)}{|M_j|}$$

represents the average percentage resource requirement of item j, while

$$\delta_j = \min{}_2\{r_{ij} : i \in M_j\} - \min\{r_{ij} : i \in M_j\}$$

is the minimum additional resource requirement if item j is not assigned to the unit with minimum requirement. Since the higher r_j or δ_j, the more critical is item j, the branching item is selected through

$$j^* = \arg \max_{j \in U} \{r_j(1 + \delta_j)\}.$$

Now let $\overline{M}_j = \{i \in M : r_{ij} \leq \overline{a}_j \text{ and } p_{ij} < z\}$ denote the set of feasible units for item $j \in U$. $|\overline{M}_{j^*}|$ son nodes are generated by assigning j^* to all $i \in \overline{M}_{j^*}$, according to increasing values of r_{ij^*}, and the search is resumed from the first of these nodes.

Fathoming Decision Nodes

Consider a decision node generated, say, by assigning item $j_a \in U$ to unit i_a. Before computing the corresponding lower bound value, the following dominance criterion is applied. The node can be fathomed if there exists an item $j_b \notin U$, currently assigned to a unit $i_b \neq i_a$, such that by interchanging the assignments the current lower bound and resource requirements do not increase.

Initialization Phase

At the root node of the branch-decision tree, a lower bound L^* on the optimal solution value is first computed. The approximation procedure of Section 3 is then applied, obtaining a first incumbent solution of value z. If $z > L^*$, the first branching item is determined, and the enumeration process begins.

5. Computational Experiments

The branch-and-bound algorithm of the previous section was coded in Fortran and computationally tested on a Digital VAX station 3100.

The computational experiments were performed on four classes of randomly-generated problems. Table 1 gives, for different values of n and m, the average number of decision nodes and the average CPU times (expressed in seconds) computed over ten problem instances. For each instance, the execution was halted as soon as the number of nodes reached 10^6. For such cases, we give (in brackets) the number of solved problems and compute the average values over them.

Class (1) was introduced for GAP by Ross & Soland(1975):

(1) r_{ij} uniformly random in [5,25], $i \in M$, $j \in N$,
p_{ij} uniformly random in [10,50], $i \in M$, $j \in N$,
$a_i = 9(n/m) + 0.4 \max_{k \in M}\{\sum_{j:i_1(j)=k} r_{kj}\}, i \in M$.

The results show that the problems of this class are very easy. Most of the instances were solved by the initialization phase. The computing time increases almost linearly with both n and m.

More difficult problems can be obtained by decreasing the a_i values:

(2) r_{ij} and p_{ij} as for Class (1),
a_i= 60% of the value obtained for Class (1), $i \in M$.

The computational results show indeed a considerable increase in the computing times, especially for m=5 and m=10. Most of the instances of this class admit no feasible solution for m=2 or 3, and very few feasible solutions for m=5 or 10.

For both Classes (1) and (2) the range of the penalty values is very limited. In order to test the behaviour of the algorithm when the optimal solution value must be found in a larger range, we considered the following class:

(3) r_{ij} uniformly random in [1,1000], $i \in M$, $j \in N$,
p_{ij} uniformly random in [1,1000], $i \in M$, $j \in N$,
$a_i = 0.6 \sum_{j \in N} r_{ij}/m, i \in M$.

The results show a satisfactory performance of the algorithm. For almost all values of m the difficulty of the instances increases for n going from 10 to 100.

The last class was obtained by introducing a correlation between penalties and resource requirements:

Table 1. Exact solution. VaxStation 3100 seconds.
Average times / Average numbers of nodes over 10 problems.

		Class (1)		Class (2)		Class (3)		Class (4)	
m	n	time	nodes	time	nodes	time	nodes	time	nodes
2	10	0.01	1	0.01	0	0.01	0	0.01	2
	25	0.03	3	0.01	0	0.01	0	0.01	0
	50	0.02	0	0.02	0	0.02	0	0.03	0
	100	0.03	0	0.03	0	0.03	0	0.03	0
3	10	0.01	0	0.07	13	0.10	6	0.07	5
	25	0.03	3	0.09	27	0.34	67	0.14	16
	50	0.02	0	0.30	76	0.21	19	0.09	4
	100	0.04	0	0.05	0	3.02	404	0.04	0
5	10	0.01	0	0.11	18	0.10	3	0.10	7
	25	0.02	0	5.69	1856	1.58	275	0.60	129
	50	0.03	0	8.35	1950	5.92	1162	0.04	0
	100	0.06	0	70.99	13078	10.25	1557	0.06	0
10	10	0.04	1	0.07	2	0.02	1	0.04	1
	25	0.03	0	1.21	177	0.76	74	2.77	491
	50	0.05	0	41.96	5951	7.46	786	367.40	54926
	100	0.11	0	338.05	41311	0.22(9)	0	0.12	0

Table 2. Approximate solution. VaxStation 3100 seconds.
Average times / Average percentage errors over 10 problems.

		Class (1)		Class (2)		Class (3)		Class (4)	
m	n	time	% err.	time	% err.	time	% err.	time	% err.
2	10	0.01	0.00	0.01	0.00	0.01	0.00	0.02	0.00
	25	0.02	0.00	0.01	0.00	0.01	0.00	0.01	0.00
	50	0.02	0.00	0.02	0.00	0.02	0.00	0.02	0.00
	100	0.03	0.00	0.03	0.00	0.03	0.00	0.03	0.00
3	10	0.01	0.00	0.05	0.00	0.09	0.00	0.06	0.00
	25	0.03	0.00	0.03	0.00	0.16	0.12	0.09	0.00
	50	0.02	0.00	0.04	0.00	0.12	0.00	0.06	0.00
	100	0.04	0.00	0.04	0.00	0.41	0.00	0.04	0.00
5	10	0.01	0.00	0.07	0.00	0.10	0.00	0.08	0.00
	25	0.02	0.00	0.21	0.31	0.28	1.03	0.19	0.65
	50	0.03	0.00	0.25	0.26	0.51	0.29	0.04	0.00
	100	0.06	0.00	0.41	1.09	0.26	0.02	0.06	0.00
10	10	0.04	0.00	0.06	0.00	0.02	0.00	0.04	0.00
	25	0.03	0.00	0.30	1.54	0.40	0.48	0.37	0.62
	50	0.06	0.00	0.20	0.00	0.95	1.02	0.22	0.00
	100	0.11	0.00	0.31	0.00	0.54	1.34	0.12	0.00

(4) r_{ij} uniformly random in [1,800], $i \in M$, $j \in N$,
p_{ij} uniformly random in [1,1000-r_{ij}], $i \in M$, $j \in N$,
$a_i = \sum_{j \in N} r_{ij}/m$, $i \in M$.

The computational results show good behaviour of the algorithm for these problems too, with comparatively higher computing times for $n \leq 50$.

In Table 2 we analyse the performance of the approximation algorithm of Section 3. For the same instances as Table 1 we give the average CPU time (expressed in seconds) and the average percentage error $100(z^a - z)/z$, where z^a is the approximate value, and z the optimal solution value or the lower bound value computed in the initialization phase (for the instance not solved exactly). The results show very good behaviour of the approximation algorithm, both for running time and the quality of the solutions found.

References

M.L. Fisher, R. Jaikumar, L.N. VanWassenhove (1986). A multiplier adjustment method for the generalized assignment problem. *Management Science* 32, 1095-1103.

M.M. Guignard, M.B. Rosenwein (1989).An improved dual based algorithm for the generalized assignment problem. *Operations Research* 37, 658-663.

K. Jörnsten, M. Näsberg (1986).A new Lagrangian relaxation approach to the generalized assignment problem. *European Journal of Operational Research* 27, 313-323.

S. Martello, P. Toth (1981).An algorithm for the generalized assignment problem. In J.P. Brans (ed.), *Operational Research*'81, North-Holland, Amsterdam, 589-603.

S. Martello, P. Toth (1990).*Knapsack Problems: Algorithms and Computer Implementations*, Wiley, Chichester.

S. Martello, P. Toth (1991).The bottleneck generalized assignment problem. Research report DEIS OR/5/91, University of Bologna.

J.B. Mazzola, A.W. Neebe (1988).Bottleneck generalized assignment problems. *Engineering Costs and Production Economics* 14, 61-65.

G.T. Ross, R.M. Soland (1975).A branch and bound algorithm for the generalized assignment problem. *Mathematical Programming* 8, 91-103.

NEURAL NETWORK ALGORITHMS FOR COMBINATORIAL OPTIMIZATION PROBLEMS: THEORY AND EXPERIENCE

Igor I. Melamed

Many combinatorial optimization problems are hard to treat, so new approaches would be fruitful. Over the past years,many attempts have been made in order to solve some combinatorial optimization problems, such as the traveling salesman and the graph partitioning problems, by applying algorithms based on neural networks. It is certainly of great interest to try to understand how a parallel structure, like neural networks, can solve this type of problems. It is likely that these kinds of studies could be useful in understanding of some mechanisms of nervous system of animals. Effectiveness of this approach has been discussed in [1].

In this paper, we present a theoretical study and computational experiments for solving the assignment, the quadratic assignment, the matching, the graph partitioning,the graph coloring, the knapsack , the set covering, partitioning and packing, the Weber - Fermat, the transport and the transshipment, the

maximal flow, the linear, the integer linear and the convex programming problems using Hopfield's neural networks.

Hopfield's neural network (HNN) is an undirected graph $G=(V,E)$, where V is the set of nodes, $|V|=n$ and E is the set of edges of G, with threshold $u_i \geq 0$ and state x_i associated with each node (neuron) i and connection of strength T_{ij} associated with each edge (i,j). $T=(T_{ij})$ is the $n \times n$ symmetric matrix, and can be $T_{ij} \geq 0$ or $T_{ij}<0$. The state of a HNN in time t is n-vector $X(t)=(x_i(t))$.

There are three types of HNN:

DD - neural networks with discrete time $t=0,1,2,...$ and discrete states $x_i \in \{0,1\}$, $i=\overline{1,n}$;

DC - neural networks with discrete time and continious states $x_i \in [0,1]$, $i=\overline{1,n}$;

CC - neural networks with continious time and continious states.

The time evolution of HNN can be described by the following systems of coupled nonlinear equations:

DD - networks

$$x_i(t+1)=H(\sum_j T_{ij}x_j(t)-u_i),\ i=\overline{1,n}$$

where

$$H(z)=\begin{cases} 0, & z \le 0 \\ 1, & z > 0 ; \end{cases}$$

DC - networks

$$x_i(t+1)=g_i(\sum_j T_{ij}x_j(t)-u_i),\ i=\overline{1,n}\ ,$$

where $g_i(z)$ - monotonic sigmoidal function with 0 and 1 t-asymptotes, for example $g_i(z)=0.5(1+\mathrm{th}z)$;

CC - networks

$$dW_i/dt = \sum_j T_{ij}x_j - u_i - W_i/\tau\ ,$$

$$x_i=g_i(W_i),\ i=\overline{1,n}.$$

HNN can be considered a nonlinear dissipative dynamic system. This system has the Liapunov function - the energy of the state. All attractors of the HNN are either fixed points or period - two limit cycles. To enable the HNN to compute a solution to the combinatorial optimization problem, the problem must be described by an energy function in which the lowest energy state (only fixed point HNN) corresponds to the best solution.

Basic theoretical results of the paper are:

1. A natural way of solving combinatorial optimization

of HNN, for which an infinitesimal breach of symmetry of the matrix T causes appearance of attractors that are large - period cycles, may be constracted.
2. If the matrix T is positive semidefinite, then in DD and CD-networks all attractors, that are limit cycles, disappear and only fixed-points remain. But for many combinatorial optimization problems the matrix T is not positive semidefinite.
3. By analyzing the eigenvalues and corresponding subspace of the matrix T, parameters in the energy function for combinatorial optimization problems may be selected.

The mentioned combinatorial optimization problems were states as energy of state HNN minimization problems. Many of the problems were studied with different energy functions. Presumably, it was determined that the neuroalgorithms find local optimal solutions for all the problems studied.A more detailed computational experiment was carried out for the linear and quadratic assignment, matching, graph partitioning, graph coloring, node covering, knapsack, Weber-Fermat, linear programming problems.

Due to the fact that the aim of experiment was the study of solving of combinatorial optimization

problems in principle small-size problems were solved. The number m of graph nodes, as well as the number of variables in linear programming problems didn't exceed 10. The knapsack and the Weber-Fermat problems were solved with up to size m =100.

Optimal solutions have been obtained in every problem after no more then 20 attempts with different initial states. Over 85% of attempts achieved local optimal solutions. The number of iterations for one attempt was about m for DD and CD-networks and about $50m$ for CC-networks. But DD and CD-networks not always obtained optimal solutions. Best results were achieved for the knapsack and the Weber-Fermat problems. The number of iterations required to find a local optimum didn't exceed three.

The traveling salesman problems were solved using Kohonen's neural networks. The results are very good. The optimal solutions were obtained for ten problems with 50 nodes within 2000 iterations.

So, the neuroalgorithms are a new interesting class of algorithms for combinatorial optimization problems.

[1] Hopfield J.J. The effectiveness of analoque neural network hardware. Network v.1,# 1, pp.27-40, 1990

NUMERICAL SOLUTION OF CERTAIN DECOMPOSITION - COORDINATION PROBLEMS IN NONLINEAR PROGRAMMING

T.Meressoo
Estonian Software and
Computer Service Company
Kadaka pst. 165,
EE0108, Tallinn

O.Vaarmann
Institute of Cybernetics
Estonian Academy of Sciences
Akadeemia tee 21,
EE0108, Tallinn

With a certainity large scale computing of the future will certainly be to a large extent parallel in one form or another. One way to break a problem up into smaller subproblems which may be treated independently is the use of decomposition-coordination schemes, i.e. the problem of the adjustment of aggregated problem in each iteration decomposes it into independent subproblems of smaller dimension. If the original problem has a block structure with coupling parameters, then these subproblems are formulated in accordance with the blocks. Computing requirements in several applications areas are unlikely to be satisfied solely by uniprocessors in the future. Decomposition of an original problem into smaller ones can greatly simplify its solution on parallel computers of various architectures.

Numerical solution of certain decomposition-coordination problem in convex programming involves often the solution of systems of nonlinear equations or minimization problems to obtain proper values for coordination parameters. The decomposition-coordination problem has some specific features:

- the user has his disposal only values of functions;
- the evaluation of function values includes, as a rule, the solution of certain subproblems and therefore it can be accompanied with a great computational effort;
- frequently these functions are continuous but not necessarily differentiable at all points of the region under consideration that causes additional difficulties.

Hence to obtain a method that is robust, stable and computationally convenient and efficient, in practice, this is far from a trivial task. In other words one principal problem is the

choice of a good starting point, the second problem is how to perform when the Jacobi matrix is singular or ill-conditioned in a region. Storage and computer time economy is also usually highly desirable. Unfortunately none of existing methods satisfies simultaneously all the above mentioned requirements. In this connection a polyalgorithmic approach can be fruitful, i.e. one combines best features of various methods and it is required that a single numerical algorithm involving in the combination is efficient in a stage of computational process or for a class of problems.

1. S t a t e m e n t o f t h e p r o b l e m. Consider a system of nonlinear equations

$$H(x,\beta) = 0 \tag{1}$$

where $H=(H_1,\ldots,H_m)^T$ and $\beta=(\beta_1,\ldots,\beta_m)^T$ while $x=(x_1,\ldots,x_n)^T$ is to be determined as a solution of nonlinear problems

$$F_i(x_i,\beta) \to \min,\ x_i \in \Gamma_i(\beta), \tag{2}$$

depending on the parameter β and where F_i is the performance index of the i-th subproblem and $\Gamma_i(\beta) \subset R^{n_i}$ is its feasible region. Assuming that the problems (2) have the solutions $x_i=x_i(\beta)$, we shall study the problem for determining the vector $\beta^* \in R^m$ from the equation

$$H(x,(\beta^*),\beta^*) = 0 \tag{3}$$

such kind of problems arise just not seldom in convex programming with a great number of variables or with a complicated structure. Further we shall reformulate the problem (3) in the form more suitable for mathematics

$$F(x) = 0, \tag{4}$$

where $F(x) \equiv H(.,x)$ and x will stand for the desired quantity (the parameter vector).

2. M e t h o d s. In general, methods involving derivatives turn often to be more effective, due to the additional information provided, than derivative free ones. For decomposition-coordination problems only function values are available. For obtaining information about derivatives finite-difference approximations are needed to implement more efficient methods in the region where F(x) is sufficinetly smooth. In the

latter case the main amount of computational work is usually spent in evaluating the Jacobian. Sacrificing accuracy in favour of simplicity one can use rank-1 approximation to the Jacobian the evaluation of which is comparatively more economical. As a rule the total number of iterations is then usually increased and such kind policy of economy is questionable. Moreover a serious disadventage of Newton method, which is in fact shared with all Newton-like methods, is the possibility of divergence in the cases that the Jacobian is (nearly) singular or ill-conditioned at some iterative points since they are based on a linear model. Methods of order $p \geq 3$ are taking adventage, at least, of a quadratic model and therefore they are capable to solve systems of nonlinear equations with the singular or ill-conditioned Jacobian. The total cost of an iterative method is determined by the number of iterations needed to achieve the required accuracy and the cost of each iteration. In this respect the implementation of the iterative methods with the convergence order higher than that of the Newton method appears to be promising since high order methods need, as a rule, for computing a solution with prescribed accuracy comparatively less iterations and therefore likely less total arithmetic for calculting the function values and its derivatives than method based on a linear model.

If a sufficiently good initial estimate for the solution is available then one can for purpose of economy to use the modification of the method of tangent parabolas

$$x_{k+1} = y_k - [F(2y_k - x_k; x_k)]^{-1} F(y_k) \tag{5}$$

$$y_k = x_k - [F(2y_{k-1} - x_{k-1}; x_{k-1})]^{-1} F(x_k) \tag{6}$$

which has the asymptotic convergence rate equal to 3 provided the second order derivatives F" is Lipschitz continuous and other reasonable assumptions are fulfilled [1]. If F has only the first derivatives and the corresponding divided differences are Lipschitz continuous then its convergence rate is at least equal to $1+\sqrt{2}$ [2]. The procedure (5)-(6) requires few information per iteration: two values of F and one value of the divided difference (except the first iteration).

The method (5)-(6) can also be interpreted as a limit case

($\rho \to 0$) of a parametric family of iterative methods

$$x_{k+1} = v_k - 2A_k F(v_k) + \frac{1}{\rho}\bar{A}_k[F(v_k + \rho A_k F(v_k)) - F(v_k)] \quad (7)$$

$$v_k = x_k - \tilde{A}_k F(x_k), \quad k=0,1,\ldots . \quad (8)$$

where A_k, $\bar{A}_k$ and $\tilde{A}_k$ are certain linear operators approximationg $(F')^{-1}$ and ρ is a nonzero real parameter [1].

If the dimensionality of the problem is not large then it may for some reason be suitable to use the following modification of tangent hyperbolas

$$x_{k+1} = x_k - F(2u_k - x_k; x_k)]^{-1} F(x_k) \quad (9)$$

where $u_k = x_k - \frac{1}{2}B_k F(x_k)$ and $B_k = [F(2x_k - x_{k-1}; x_k)]^{-1}$.

In particular, the symbols A_k, $\bar{A}_k$, $\tilde{A}_k$ and B_k not only denote finite difference approximations to $\Gamma_k = [F'(x_k)]^{-1}$ and $[F'(x_k - \frac{1}{2}\Gamma_k F(x_k)]^{-1}$ respectively but they can also express the fact that the corresponding linear equations are solved approximately [3].

In addition to stability and economy of the method a crucial problem is the choice of a good starting point. A serious defect of high order methods is their pretentiousness with respect to an initial guess, i.e. the adventages of those become evident mostly in the close vicinity of a solution.

One of the most effective ways to guarantee the global convergence or at least greatly to expand the domain of convergence of a method is the "continuation strategy". According to this strategy one first replaces $F(x) = 0$ by a one-parameter family of problems $G(x,\lambda) = 0$, $\lambda \in [0,1]$, such that $F(x) = G(x,1)$ and the solution of $G(x,0) = 0$ is known secondly one solves a series of problems as the parameter λ is slowly varied by a locally convergent iterative method using the solution to the previous problem as a starting point for the current problem. The rules for changing the value λ can be suggested by the physical nature of the problem but some standard mathematical algorithms exist for this as well. All continuation (homotopy) methods suffer from the disadventage that a Jacobian at some points could become singular. Therefore the homotic strategy needs stable local algorithms to be successful [4,5]. In this respect methods

based on a quadratic model meet this requirement and using them in homotopy methods can be justified even if the implementation of high order methods for solving some particular problems in the homotopy methods is not efficient due to a great number of computational effort involved.

This report suggests another approach to ensure the convergence from a poor starting point. This strategy consists in combining global and local methods to obtain more robast ones. This approach may be more successful for decomposition-coordination problems due to their possible nonsmoothness at some iterative points, i.e. the functions determining the problem may belong to almost differentiable functions [8]. The strategy of the polyalgorithmic procedure under consideration is use of a high order method ($p>2$) if it works otherwise to switch on a modification of the Newton method with finite-difference approximation of the Jacobian [cf.9] bearing in the mind that the Newton method requires a good initial guess only for guaranteering quadratic convergence but in many cases it may progress starting from a poor initial estimate [7]. If the Newton-type method also does not work then to switch on a more slower but sure global method depending on smoothness of the problem: on a method based on the steepest direction provided involving functions are differentiable or otherwise on a method using subgradients (e.g. methods from [8]). After accomplishing certain number of iterations by the global method one attempts to start with a high order method once again.

In the case of smooth problems one can take as a global method the one

$$x_{k+1} = x_k - \alpha_k g(x_k), \qquad \alpha_k>0, \tag{10}$$

where $g(x) = [\Delta F(x,h)]^T F(x_k)$ and $\Delta F(x,h)$ is a finite-difference approximation to F' and h is the step of discretization or the two parametric methods [cf.9]

$$x_{k+1} = x_k - 2\varepsilon_k g(x_k) + \varepsilon_k \alpha_k^2 [\Delta F(x_k,h_k)]^T \Delta F(x_k,h_k) g(x_k), \quad \varepsilon_k>0 \tag{11}$$

Likewise (10) and(11) can be used for solving linear equations in the inexact local methods.

3. N u m e r i c a l e x a m p l e s . It is well known that mathematically efficient methods do not necessary result in efficient computer programs. On the other hand the worst case bounds for a method may be too pessimistic and, in practice, it may, on the average, to perform better than expected.

The performance of the above method (5)-(6), (7)-(8) and (9) is tested on a small set of test problems containing 14 problems for systems of nonlinear equations of Argonne National Laboratory plus Freudenstein and Roth function and Box three-dimensional function for nonlinear least squares taken from [10]. The methods under review were compared with the following methods which were tested on the same set of test problems.

1. The Newton method.
2. The modified Newton method ($\Gamma_o=\Gamma_k$, $k=1,2,\ldots$).
3. The version of the Newton method with finite-difference approximation.
4. The Bartysh method ($p=1+\sqrt{2}$) [11].
5. The Kogan method [12]

$$v_k = x_k - \tfrac{1}{2}\Gamma_k F(x_k).$$

$$x_{k+1} = x_k - [F'(v_k)]^{-1}F(x_k).$$

6. The modification of the Kogan method

$$v_k = x_k - \tfrac{1}{2}\Gamma_k F(x_k)$$

$$x_{k+1} = x_k - \Gamma_k \sum_{i=0}^{q-1} (I - F'(v_k)\Gamma_k)^i F(x_k)$$

where I denotes the identity mapping.

7. Shacham method (CONLES) which combines the Newton and the Levenberg-Marquart ones [13].

Further the following numeration is used.

8. The method (5)-(6).
9. The combination of the method (5)-(6) and Newton's one.
10. The method (7)-(8) with $\rho=-1$, $\rho=-\frac{1}{2}$ and $\rho=-5$.

All the problems were run on a EC1060 computer under a FORTRAN-IV compiler. The calculus was performed in double precision and stopped $\|x_{k-1} - x_k\| \leqslant 10^{-9}$. Termination of the routine also occured when the number of iterations exceeded the given maximum value 300. The table 1 gives iteration number N at

which the presigned accuracy was achieved and where "-" denotes the failure (nonconvergence). The table 1 indicates that the methods (5)-(6) and (7)-(8) require a good initial estimate. But when started from an initial point, which was close enough to the solution they would converge very fast. Besides they guaranteed almost the same results, both in single and double precision, while the Newton method yielded worse results with the single precision.

Table 1

Test	Method 1	2	3	4	5	6	7	8	9	10 $A_\rho : A_{-\frac{1}{2}} : A_{-5}$		
										$\rho=-1$	$\rho=-\frac{1}{2}$	$\rho=-5$
1	3	3	3	3	2	2	3	2	2	2	2	2
2	32	k>300	31	27	21	21	18	-	38	14	14	13
3	13	-	19	12	9	9	12	-	12	8	8	8
4	15	k>300	47	14	28	24	14	-	17	-	-	140
5	11	-	-	-	6	8	9	-	10	-	-	-
6 (m=6)	13	k>300	45	15	9	9	12	-	14	8	8	-
7	6	-	6	6	4	5	5	-	5	4	-	-
8	91	-	80	-	k>300	-	6	-	92	-	-	-
9	4	6	3	4	3	3	3	3	3	3	3	2
10 (m=10)	4	6	4	4	3	3	3	3	3	3	3	2
11	8	k>300	6	12	7	-	8	-	6	-	-	-
12	15	k>300	20	13	10	10	14	-	16	9	9	9
13	5	22	4	5	4	4	5	-	5	3	3	3
14	7	60	6	6	5	5	6	-	5	4	4	4
15	43	k>300	42	10	25	60	49	23	23	67	51	229
16	6	30	5	5	4	4	5	4	4	4	4	3

The comparison of the columns 5 and 6 shows that the Kogan method and its inexact version give approximately the same results.

The performance of the hybrid method (5)-(6) which combines the best features of (5)-(6) and the Newton method as well as method CONLES was slightly superior both in speed and accuracy to the Newton method. These promising results encourage to carry on (extend) the investigation of properties of polyalgorithmic procedures.

References

1. Ваарманн О., Полль В. О решении нелинейных уравнений методами высокого порядка скорости сходимсоти и их устойчивости Изв. АН ЭССР. Физ. Матем., 1977, 26, №2, с.123-127.
2. Мереccоо Т. Об одном методе порядка $1+\sqrt{2}$ решения нелинейных уравнений. Изв. АН ЭССР. Физ. Матем. 1983, 32, №4, с.368-372.
3. Vaarmann, O. Some approximate variants of classical iterative methods with high order convergence. In: Numerical Methods and Optimization (Estonian Academy of Sciences), Tallinn, 1992, v.3, pp.78-87.
4. Watson, L.T. Numerical linear algebra aspects of globally convergent homotopy methods. SIAM Rev. 1986, v.28, №4, pp.529-545.
5. Gomes-Ruggiero, M.A., Martinez, J.M. and Moretti, A.C. Comparing algorithms for solving sparse nonlinear systems of equations. SIAM J. Sci. Stat. Comput, 1992, v.13, №2, pp.459-483.
6. Szabô, Z. Combined iteration method for solving equations. Numerical Methods (Miskolc, 1986); Colloq. Math. Sci. Janos Bolyai, 50, North-Holland, Amsterdam-New York, 1988, pp.581-587.
7. Gill, P.E., Murray, W. and Wright, M.H. Practical Optimization, London-New York, Academic Press, 1981.
8. Шор Н.З. Методы минимизации недифференцируемых функций и их приложения. Киев, Наукова Думка, 1979.
9. Vaarmann, O. Towards inverse problems in nonlinear dynamical systems and some methods for their numerical solution. In: Numerical Methods and Optimization, Tallinn, Valgus, 1990, v.2, pp.36-48.
10. More, J.J., Garbow, B.S. and Hillstrom, K.E. Testing unconstrained optimization software. ACM Trans. Math. Soft., 1987, 7, №1, pp.17-41.
11. Бартиш М.Я. Об одном итерационном методе решения функциональных уравнений - ДАН УРСР, 1968,сер.А, №5, с.387-391.
12. Коган Т.И. Об одном итерационном процессе для функциональных уравнений. Сиб. Мат. журн., 1967, т.8, №4, с.958-960.
13. Shacham, M. Numerical solution of Constrained nonlinear algebraic equations. International Journal for Numerical Methods in Engineering, 1986, v.23, pp.1455-1481.

Extended Abstract

Some large scale LP relaxations for the graph partitioning problem and their optimal solutions

M. Minoux
Université Paris 6[1]

1 Introduction

The so-called Graph Partitioning Problem (GPP) has applications in many areas such as data analysis and clustering, VLSI circuit layout, block decomposition of large linear systems, etc ...

The purpose of this paper is to investigate a family of large scale linear programming relaxations for (GPP) which extends the one considered in Minoux [4] and Minoux and Pinson [5]. Some of the problems in the family are shown to be polynomially solvable, in which case characterizations of their optimal solutions are given.

As an outcome, bounds are derived which may be of use e.g. to validate approximate (heuristic) solutions, or to implement Branch and Bound procedures.

We consider here the unconstrained version of the graph partitioning problem which may be stated as follows. Suppose we are given an undirected connected graph $G = [X, U]$ with node set X ($|X| = N$) and edge set U ($|U| = M$), and an integer $p < N$. For any $S \subset X$, we denote $\gamma(S)$ the total number of edges in U having both endpoints in S.

(GPP) is then to find a partition of X into p subsets $S_1, S_2, \ldots, S_p$ such that the quantity:

$$z = \gamma(S_1) + \gamma(S_2) + \cdots + \gamma(S_p)$$

is *maximized.*

[1]MASI, Université Paris 6, 4 Place Jussieu, 75005 Paris, France

2 A family of large scale (LP) relaxations

For any integer $q \quad (N-p+1 \leq q \leq N)$ we consider the large scale set partitioning problem $(R[q])$ defined as follows.

Suppose that the nonempty subsets of X are numbered $1, 2, \ldots, J \quad (J = 2^N - 1)$ and let J_q denote the number of subsets having cardinality $\leq q$.

Let us denote $A = (a_{ij})_{\substack{i=1,\ldots,N \\ j=1,\ldots,J_q}}$ the incidence matrix of all the subsets with cardinality $\leq q$. Thus, for the subset S_j having index number $j \quad (1 \leq j \leq J_q) \quad a_{ij} = 1$ if node i is an element of S_j, $a_{ij} = 0$ otherwise.

To each subset S_j we let correspond a binary variable $x_j \quad (x_j = 1$ if subset S_j is selected in the optimum partition to be found, $x_j = 0$ otherwise) and a cost c_j defined by:

$$c_j = \gamma(S_j)$$

where $\gamma(S_j)$ denotes the total number of edges in U having both endpoints in S_j. With this notation, the large scale set partitioning problem $(R[q])$ reads:

$$(R[q]) \quad \begin{cases} \text{Maximize} \sum_{j=1}^{J_q} c_j x_j \\ \text{subject to:} \\ A.x = \mathbf{1} & (1) \\ \sum_{j=1}^{J_q} x_j = p & (2) \\ x \in \{0,1\}^{J_q} & (3) \end{cases}$$

($\mathbf{1}$ denotes the N vector with all components 1).

Since any partition solving (GPP) should composed only of subsets having cardinality $\leq N-p+1$, for any $q \geq N-p+1 \quad (R[q])$ and (GPP) are equivalent. Even for graphs having moderate number of nodes (> 15 say) $(R[q])$ will have an enormous number of columns. So for $q \geq N-p+1$ we are led to consider the linear relaxations $(\overline{R[q]})$ to $(R[q])$ which are simply obtained by replacing (3) by:

$$x_j \geq 0 \quad (j = 1, \ldots, J_q).$$

We note here that for the special case $q = N$ the set partitioning problem $(R[N])$ and its relaxation $(\overline{R[N]})$ were introduced and studied in Minoux [4]. In particular it was shown there that $(\overline{R[N]})$ is polynomially solvable by means of the "Ellipsoïd Algorithm" (Khachian [3]). In Minoux and Pinson [5] a generalized linear programming algorithm (Dantzig [1]) was applied to derive bounds on optimal solution values to (GPP) and computational results were presented.

3 Polynomially solvable cases and associated optimal solutions

The first result concerns the special case $q = N$. The polynomial solvability of $(\overline{R[N]})$ was already established in Minoux [4], but the proof there was based on the use of the Ellipsoïd algorithm; thus the existence of a purely combinatorial algorithm for solving $(\overline{R[N]})$ was left open. Theorem 1 below shows that, at least for some values of p, the question may be answered positively. Given a subset $S \subset X$ such that $|S| \geq 2$, we define $B[S]$ as the $(N+1) \times (N+1)$ matrix:

$$B[S] = \begin{bmatrix} I_N & \vdots & v \\ \dots & \vdots & \dots \\ \mathbf{1}^T & \vdots & 1 \end{bmatrix}$$

with v the incidence vector of S in X.

Theorem 1 *Consider* $S^* \subset X \quad |S^*| \geq 2$ *such that:*

$$\gamma(S^*)/(|S^*|-1) = \max_{\substack{S \subset X \\ |S| \geq 2}} \{\gamma(S)/(|S|-1)\} \tag{4}$$

and suppose that $|S^*| \geq N - p + 1$.

Then $B[S^*]$ *is an optimal feasible basis for* $(\overline{R[N]})$ *and the corresponding optimal objective function value (an upper bound to (GPP)) is:*

$$(N-p) \times \frac{\gamma(S^*)}{|S^*|-1}. \tag{5}$$

The maximization problem (4) is a *maximum ratio problem* which can be efficiently solved in polynomial time, by a purely combinatorial algorithm (through a sequence of maximum flow problems). Therefore, when the conditions of Theorem 1 apply (namely $p \geq N - |S^*| + 1$) this result leads to a purely combinatorial algorithm for solving $(\overline{R[N]})$ both practically and theoretically more efficient than the other previously known approaches (Ellipsoïd algorithm or generalized linear programming).

The second result below is a generalization of the previous one to other problems in the family $(\overline{R[q]})$, for some values of q in the range $[N-p+1, N]$. Since the strength of the relaxations $(\overline{R[q]})$ improves for smaller values of q, it is not surprising that the resulting (upper) bounds are never worse (but usually better) than those given by (5).

Theorem 2 *For* $\lambda > \lambda^* = \dfrac{\gamma(S^*)}{|S^*|-1} = \max\limits_{\substack{S \subset X \\ |S| \geq 2}} \left\{ \dfrac{\gamma(S)}{|S|-1} \right\}$ *let* $\overline{S}$ *be a subset satisfying:*

$$\gamma(\overline{S}) - \lambda(|\overline{S}|-1) = \max_{S \subset X} \{\gamma(S) - \lambda(|S|-1)\} \tag{6}$$

and $(\overline{S}) = q$.

Then for $q \geq N-p+1$, $B[\overline{S}]$ is an optimal feasible basis for $(\overline{R[q]})$ and the corresponding optimal objective function value (an upper bound for (GPP)) is:

$$(N-p) \times \frac{\gamma(\overline{S})}{|\overline{S}|-1}. \tag{7}$$

Since there are only finitely many subsets of X, (6) has only finitely many distinct optimal solutions when λ varies in the range $]\lambda^*, +\infty[$. Let $\overline{S}_1, \overline{S}_2, \ldots, \overline{S}_t$ denote those solutions with cardinality $\geq N-p+1$ and let $|\overline{S}_1| = q_1$, $|\overline{S}_2| = q_2, \ldots, |\overline{S}_t| = q_t$. Theorem 2 implies that $(\overline{R[q]})$ is polynomially solvable (by a purely combinatorial algorithm) for $q \in \{q_1, q_2, \ldots, q_t\}$. Moreover the best possible (upper) bound for (GPP) which may be derived from (7) is:

$$(N-p) \times \frac{\gamma(\overline{S}_r)}{|\overline{S}_r|-1} = (N-p) \times \min_{i=1,\ldots,t} \left\{ \frac{\gamma(\overline{S}_i)}{|\overline{S}_i|-1} \right\}.$$

References

[1] Dantzig G.B. (1963) "Linear programming and extensions". Princeton University Press.

[2] Grötschel M., Lovász L., Shrijver A. (1981) "The Ellipsoïd method and its consequences in combinatorial optimization". Combinatorica, Vol. 1, n°2, pp. 169-197.

[3] Khachian L.G. (1979) "A polynomial algorithm in linear programming". Soviet Math. Dokl., Vol. 20, n°1, pp. 191-194.

[4] Minoux M. (1987) "A class of combinatorial problems with polynomially solvable large scale set covering/partitioning relaxations". RAIRO, Vol. 21, n°2, pp. 105-136.

[5] Minoux M., Pinson E. (1987) "Lower bounds to the graph partitioning problem through generalized linear programming and network flows". RAIRO, Vol. 21, n°4, pp. 349-364.

Reinhard Neck
Department of Economics, University of Bielefeld
Bielefeld, Germany
Mailing Address: Josefstaedter Strasse 87/2/34,
A - 1080 Vienna, Austria

Optimal Budgetary Policies under Uncertainty:

A Stochastic Control Approach

Extended Abstract

Since the mid-eighties the size of the federal budget deficit has been of much concern to policy-makers in Austria. By and large, they now generally agree upon the necessity of consolidating the federal budget to prevent a loss of credibility of fiscal policies. Nevertheless, there are trade-offs and side-effects associated with a policy of gradually or even suddenly diminishing the budget deficit. So far, no quantitative informations are available about the effects of budgetary measures on the main objectives of Austrian economic policy, such as growth, full employment, price stability, and balance-of-payments equilibrium. Moreover, neither the intertemporal trade-offs nor the issue of policy-makers' limited information about future events has received serious attention in the political debate in Austria so far. The question of how to design budgetary policies under these conditions can be seen as a typical problem of the theory of quantitative economic policy. If we are ready to postulate an objective function to be optimized by policy-makers, we can apply stochastic control theory to derive and analyse optimal budgetary policies for past or future periods.

In this paper we determine optimal budgetary policies for Austria using a small macroeconometric model. This model, called FINPOL1, is based on traditional Keynesian macroeconomic theory in the sense of IS-LM/aggregate demand-aggregate supply models. Stochastic behavioral equations for the demand side of the economy include a consumption function, an investment function, an import function, and an interest-rate equation as a reduced-form money market model. Prices are largely determined by aggregate demand variables. Disequilibrium in the labor market, as measured by the excess of unemployed persons over vacancies, is modeled to depend upon the real GDP growth rate and the rate of inflation, embodying both an Okun's law-type relation and a rudimentary Phillips curve. The main objective variables of Austrian economic policies, such as real GDP, the labor market disequilibrium variable

(related to the rate of unemployment), the rate of inflation, the balance of payments and the ratio of the federal net budget deficit to GDP, are related to those fiscal and monetary policy instruments which are used as control variables. Particular attention is given to the influence of variables of the federal budget (revenues and expenditures) on the endogenous variables of the model. The model FINPOL1 is a nonlinear model; it was estimated by ordinary least squares using annual data. Several tests and tentative simulation experiments indicate that its tracking capability is reasonable and that it provides a satisfactory framework for analysing effects of budgetary policies on the Austrian economy.

Next, we apply the algorithm OPTCON to calculate optimal fiscal policies for the eighties, during which the problem of rising federal budget deficits was most pronounced. OPTCON has been developed by Josef Matulka and the author; it determines approximately optimal control paths for nonlinear stochastic dynamic systems under quadratic objective functions. We give a short description of this algorithm and discuss its strengths and present limitations. An objective function is formulated which serves as expression of the preferences of an hypothetical policy-maker in charge of the federal budget. Target paths and preference weights for the objective variables are determined in an ad-hoc way after some trial and error. Using this objective function and the dynamic constraints as given by the equations of the model FINPOL1, we determine approximately optimal time paths for the budgetary control variables and the endogenous (especially the target) variables under the assumption of complete information for the policy-maker. In addition to this deterministic optimization run, different assumptions about parameter uncertainties are introduced in order to assess the influence of various kinds of uncertainty on the design of optimal budgetary policies. In particular, we investigate the effects of making several key parameters determining fiscal and monetary policy multipliers uncertain. The results of these different stochastic optimum control experiments are compared to each others and to the historical values of the control and endogenous variables. Interpretations are given for the effects of uncertainty on optimal budgetary policies and for the potential of improving policy performance by using optimization techniques such as stochastic optimum control theory. We conclude that, contrary to popular assertions, there is a considerable potential for welfare gains to be obtained by using optimization methods for the design of economic policies, even and particularly in the presence of uncertainty about policy effects. Some suggestions on how to exploit that potential for the formulation of budgetary policies in Austria in the future are given.

ABSTRACT

Genetic Search in Multi-Depot Routing

Kendall E. Nygard (*)
Department of Computer Science and Operations Research
North Dakota State University
Fargo, ND, USA 58105-5075
nygard@plains.nodak.edu
(701)-237-8203

Rhonda K. Ficek
Department of Computer and Information Systems
Moorhead State University
Moorhead, Mn
ficek@plains.nodak.edu

In the multiple depot vehicle routing problem (MDVRP), a fleet of heterogeneous vehicles with known capacities originate from and return to depots with fixed locations, and service a given set of nodes with known demands for service. The objective is to minimize the variable cost of vehicles traveling to serve the nodes. The well-known vehicle routing problem (VRP), is a special case in which there is a single depot. The VRP is known to be an P-hard problem. Thus, computational effort for known exact solution procedures increases exponentially in problem size.

Cluster-First/Route-Second heuristic methods approach the solution of multiple vehicle problems by assigning nodes to vehicles in a first phase, then determining the sequences for each vehicle in a second phase. Seed-setting methods are a particularly successful cluster-first/route-second approach to the VRP. These methods utilize a set of geographical locations called seeds, one for each vehicle, that model the nominal directions and distances from a depot that the vehicles travel. The seeds are used to set parameters for a procedure that assigns the nodes to vehicles without exceeding their capacities. Examples of such parameters include: i) the straight line distances from node locations to the seeds, and ii)the extra distances incurred if vehicles traveling to a seed and back deviate to serve the node. Using parameters such as these, a variety of node assignment procedures have been developed, including generalized assignment integer programming solvers and various heuristics. As the final step, a traveling salesman problem (TSP) must be solved for each set of nodes associated with each vehicle. In the MDVRP, the process of assigning nodes to seeds (vehicles) implicitly assigns the nodes to depots as well.

Recent research has proven successful in employing genetic search to iteratively seek seed location patterns that generate good solutions to the VRP. Genetic search is a general purpose heuristic procedure that uses concepts of selection and inheritance to artificially evolve good solutions, within a framework inspired by the genetics of biological systems. In a genetic search, three items must be suppied - a representation of a candidate solution as an artificial chromosome, a means of evaluating the fitness of a candidate solution, and recombination operators. In our application, an artificial chromosome consists of (x,y) coordinate locations for seeds expressed in binary. Fitness evaluation is through generalized assignement integer programming models and other methods that assign nodes to seeds, followed by a traveling salesman heuristic. Recombination adjusts seed point locations through two-point crossover. With this method, best-known solutions to many well-known vehicle routing problems have been generated. In this study, we extend the use of genetic search for seed setting to the multi-depot case. Several heuristics were developed for assigning the nodes to seed points. Exact and heuristic solvers for generalized assignment problems were developed and tested for this purpose, as well as greedy clustering heuristics. Local tour improvement operators were devised to modify the tours given by the TSP solver at each iteration. Both network optimization and r-opt types of operators were effective in local tour improvement. In some of the experiments, several node assignment heuristics were run at each iteration, and the best method was used in the genetic selection process within the population. In essence, this provides several alternative evaluation methods evaluation for each population member, a new technique that we call multiple sharing evaluation functions.

The methods were experimentally tested on suites of test problems that are extensions of problems from the literature, as well as on randomly generated problems. Problems with up to 250 nodes and 25 vehicles were tested on a DEC 5000/133 desktop workstation computer. A visualization interface and statistical measures show that efficent routes are consistently generated by the procedure. For problems with small numbers of depots, rough comparisons with single depot solutions can be obtained. We determined that our procedure is competive with known VRP solvers for such problems. Although the MDVRP is of high practical importance, heuristics previously developed specifically for the problem are unsophisticated, and our new methods produce much better results.

(*) Contact author

Order Preserving Assignments

Manfred Padberg*and Dimitris Alevras
New York University

Extended Abstract

1 Introduction

The *order preserving assignment* problem is the following variation of the classical assignment (or marriage) problem: given an *ordered* list of n "items", p possible "positions" and integer "profits" c_{ij} for the assignment of item i to position j find a profit-optimal assignment of items to positions that uses *at most p contiguous* positions (starting with position 1) and that *preserves* the order of the list of items, i.e. the highest ranked item that gets assigned to a position is assigned to position 1 and so forth. Note, however, that we do *not* require that any item be assigned to a particular position. We are *merely* requiring that (the straight) "lines do not cross". Figure 1 shows an example of an order preserving assignment (OPA) for $n = 4$ and $p = 3$ that uses two positions. We assume that the order of the items agrees with their indexing, i.e. item 1 is the highest ranked item, item 2 the second highest and so on and, of course, that the profits are *additive*.

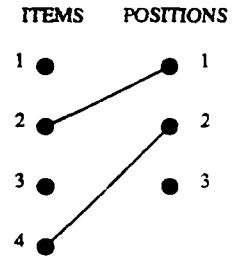


Figure 1. OPA with n=4 and p=3.

Since to every *combination* of j items out of n distinct items there corresponds exactly one order preserving assignment, it follows that there are exactly $\sum_{j=0}^{p} \binom{n}{j}$ OPAs for n items and at most p positions. This is a substantially smaller number than the number of possible assignments (without regard to order and contiguity) of which there are exactly $\sum_{j=0}^{p} j!\binom{n}{j}\binom{p}{j}$. However, the number of OPAs for reasonably sized n and p puts the optimization problem out of reach for complete enumeration and – as small examples show – simple greedy procedures do not solve the problem either. In section 3 of this paper, we *formulate* the order preserving assignment problem, give a *minimal description* of the problem in terms of linear inequalities, show that a profit-optimal order preserving assignment can be found in *strongly polynomial time* and prove that the *diameter* of the associated polytope equals two. In section 4 we derive the corresponding results for the case when all p positions must be assigned. In section 5 we show how to *find* a profit-optimal order preserving assignment in *linear time*, i.e. in time that is linear in the number of variables of the problem, for both cases that we consider. So, as far as assignments are concerned, "order and contiguity" reduce

*Supported in part by grants from ASFOR, ONR and the Alexander-von-Humboldt Stiftung.

the set of feasible assignments, do *not* bring down the diameter of the polytope (compare to Balinski and Russakoff [1972] for the classical assignment problem), but they make algorithmic "life" easier (compare e.g. to Balinski[1983,1985], Kuhn[1955], Munkres [1957]). In section 6 we discuss two modifications of the basic model. Throughout the paper we assume that the reader is familiar with the fundamental concepts of graph theory, linear programming and the analysis of algorithms. For a survey of the polyhedral theory that we employ we refer the reader to Grötschel and Padberg [1985]. This note is a summary of the principal results that can be found in detail in our paper [9].

2 Some definitions

The problem is defined on a bipartite graph $G = (N, P, E)$ where $N = \{1, \ldots, n\}$, $P = \{1, \ldots, p\}$ and $1 \leq p \leq n$. An edge of G is an ordered pair $(k, t) \in N \times P$ and is denoted by a single letter $(e, a, \ldots$ etc.) or by its defining pair of indices. We assume at first that G is the *complete* bipartite graph, but we will redefine the edge set E after the following definition.

Definition 1 *A subset $A \subseteq E$ is an order preserving assignment (OPA) if*

(i) for every $k \in N$ there is at most one $a \in A$ such that $k \in a$.
(ii) for every $t \in P$ there is at most one $a \in A$ such that $t \in a$.
(iii) if $a = (k, t) \in A$ and $t \geq 2$, then there exists $\kappa \in N$, $\kappa < k$ such that $(\kappa, t-1) \in A$.
(iv) if $a = (k, t), b = (\kappa, \tau) \in A$ and $a \neq b$, then either $k < \kappa$ and $t < \tau$ or $k > \kappa$ and $t > \tau$.

Properties (i) and (ii) ensure that A is an assignment, property (iii) ensures that positions are assigned *contiguously* starting with position 1, whereas property (iv) ensures that the assignment preserves the order of the items. It follows from the definition of an OPA that item k cannot be assigned to a position $t > k$. Hence there is no need to consider edges (k, t) with $t > k$ and thus the edge set E of G is given by

$$E = \{(k, t) \in N \times P \mid t \leq k \leq n, 1 \leq t \leq p\}.$$

We index the edges *sequentially* when necessary by

$$e = (t-1)n + j - t(t-1)/2\,, \tag{1}$$

where $t \leq j \leq n$ and $1 \leq t \leq p$.

If $A \subseteq E$ is an OPA and $e = (k, t) \in A$, then we say that item k is assigned to position t. Let $\Re^E$ (rather than $\Re^{|E|}$) denote the space of real vectors of length $|E|$. The *support* of $x \in \Re^E$ is the index set of the nonzero components of x. For $A \subseteq E$, we denote $x^A = (x_e^A) \in \Re^E$ the characteristic vector (or incidence vector) i.e.

$$x_e^A = \begin{cases} 1 & \text{if } e \in A \\ 0 & \text{if } e \notin A. \end{cases}$$

We define the *order preserving assignment polytope* OP_p^n to be the *convex hull* of the characteristic vectors of all OPAs, i.e.

$$OP_p^n = conv\left\{x^A \in \Re^E \mid A \subseteq E \quad \text{is an} \quad OPA\right\}.$$

The order preserving assignment problem then is the optimization problem

$$(OP) \qquad max\{c^T x \mid x \in OP_p^n\}.$$

We call an OPA *exact* if $|A| = p$. It follows from the definition of an OPA that in an exact OPA item k cannot be assigned to position t if $k > n - p + t$, where $1 \leq t \leq p - 1$, since otherwise there are not

enough items left to be assigned. For the case of exact OPAs we are thus led to consider a bipartite graph $G = (N, P, D)$ where

$$D = E - \{(k,t) \mid n - p + t + 1 \leq k \leq n\ ,\ 1 \leq t \leq p-1\}\ .$$

We define the *polytope of exact OPAs* $OP^{=}_{np}$ to be the convex hull of the characteristic vectors of all exact OPAs, i.e.

$$OP^{=}_{np} = conv\left\{x^A \in \Re^D \mid A \subseteq D \text{ is an } OPA \text{ with } |A| = p\right\}\ .$$

Note that $|D| = |E| - p(p-1)/2$ and that $OP^{=}_{np}$ is clearly not full dimensional. We denote $\text{aff}(OP^{=}_{np})$ the *affine hull* of the polytope $OP^{=}_{np}$. The corresponding optimization problem over $OP^{=}_{np}$ is denoted by $(OP^{=})$.

A *linear description* of a polytope is a set of linear inequalities and/or equations whose set of solutions equals the polytope. A linear description is *minimal*, if none of the inequalities and/or equations can be dropped from it without changing the solution set, i.e. every inequality defines a *facet* of the polytope. Evidently, we are interested in finding minimal descriptions of the polytopes OP^{n}_{p} and $OP^{=}_{np}$.

3 Assigning at most p positions

We show first that OP^{n}_{p} is a polytope of full dimension.

Proposition 1 *dim* $OP^{n}_{p} = |E| = np - p(p-1)/2$.

To *formulate* the problem we propose the following system of linear inequalities in zero-one variables which on first sight has little to do with "assignments".

$$\sum_{j=1}^{n} x_{j1} \leq 1 \tag{2}$$

$$\sum_{j=t+1}^{k+1} x_{j,t+1} \leq \sum_{j=t}^{k} x_{jt} \quad t \leq k \leq n-1\ ,\ 1 \leq t \leq p-1 \tag{3}$$

$$x_{jt} \in \{0,1\} \quad t \leq k \leq n \quad 1 \leq t \leq p\ . \tag{4}$$

Inequality (2) states that at most one item can be assigned to position 1, while (3) expresses the condition that if some items $t+1, \ldots, k+1$ are assigned to position $t+1$ then at least that many items from $t, \ldots, k$ must be assigned to position t where t and k are as specified. Inequalities (3) are intuitively not readily understood and were obtained by calculating *by computer* all of the facets of OP^{n}_{p} for small values of n, p and subsequent generalization.

Proposition 2 *The system* (2), (3) *and* (4) *is a* formulation *of the order preserving assignment problem, i.e. every solution to* (2), (3) *and* (4) *is the incidence vector of an OPA and vice versa.*

Proposition 2 establishes, in particular, the *validity* of inequalities (2) and (3) for the polytope OP^{n}_{p}. There are $n(p-1) - p(p-1)/2 + 1$ constraints of the form (2) and (3) and the question is whether or not we need all of them. The following proposition shows more than that.

Proposition 3 *Every inequality (2) or (3) defines a facet of the polytope* OP^{n}_{p}. *Moreover, the facets defined by (2) and (3) are all distinct.*

The inequalities (2) and (3) are certainly *not all* the facets of OP^{n}_{p} as we will need nonnegativity as well. However, inequalities (3) for $k = t$ read $x_{tt} \geq x_{t+1,t+1}$ and thus we are lead to consider

$$x_{pp} \geq 0\ , \quad x_{jt} \geq 0 \quad \text{for} \quad t+1 \leq j \leq n\ ,\ 1 \leq t \leq p\ . \tag{5}$$

Proposition 4 *The inequalities (5) define distinct facets of OP_p^n. Moreover, these facets are distinct from the ones given by (2) and (3).*

Theorem 1 *$OP_p^n = \{x \in \Re^E | x$ satisfies (2), (3) and (5)$\}$ for all $1 \leq p \leq n$. The linear system (2), (3) and (5) is a minimal description of OP_p^n and the constraint matrix given by (2) and (3) is totally unimodular.*

Sketch of Proof: Consider the $|E| \times |E|$ matrix T given by the transformation

$$y_{kt} = \sum_{j=t}^{k} x_{jt} \quad \text{for} \quad t \leq k \leq n\,,\ 1 \leq t \leq p\,. \tag{6}$$

The matrix T is nonsingular and the inverse transformation T^{-1} is given by

$$x_{tt} = y_{tt} \quad t = 1, \ldots, p \tag{7}$$

$$x_{kt} = y_{kt} - y_{k-1,t} \quad t+1 \leq k \leq n\,,\ 1 \leq t \leq p\,. \tag{8}$$

Applying the transformation (6) to inequalities (2), (3) and (5) we find

$$y_{k+1,t+1} - y_{kt} \leq 0 \quad t \leq k \leq n-1\,,\ 1 \leq t \leq p-1 \tag{9}$$

$$y_{k-1,t} - y_{kt} \leq 0 \quad t+1 \leq k \leq n\,,\ 1 \leq t \leq p \tag{10}$$

$$-y_{pp} \leq 0 \tag{11}$$

$$y_{n1} \leq 1 \tag{12}$$

The matrix corresponding to the constraints (9),...,(12) is totally unimodular. Consequently, all the extreme points of $X = \{x \in \Re^E \mid x$ satisfies (2), (3) and (5)$\}$ are zero-one valued and hence, by Proposition 2, $X = OP_p^n$. It follows that the linear system (2), (3) and (5) is a linear description of OP_p^n. By Propositions 3 and 4 we have its minimality. The total unimodularity of the constraint matrix given by (2) and (3) is a consequence of the total unimodularity of the matrix T of our transformation and the total unimodularity of (9),...,(12), see e.g. Cook [1983]. □

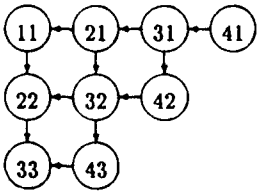


Figure 2. OPA network for n=4 and p=3.

The proof of Theorem 1 suggests a "reformulation" of the order preserving assignment problem in terms of variables y_{kt}, where

$$y_{kt} = \begin{cases} 1 & \text{if one of the items } t, \ldots, k \text{ is assigned to position } t \\ 0 & \text{if not} \end{cases}$$

and $t \leq k \leq n$, $1 \leq t \leq p$. The constraints (9) and (10) can then be interpreted accordingly, while constraints (11) and (12) ensure that all variables assume values between zero and one. More precisely, the optimization problem (OP) can be written as follows:

$$max\{c^T x \mid x \in OP_p^n\} \tag{13}$$

$$= max\{c^T x \mid x \in \Re^E, x \text{ satisfies (2), (3) and (5)}\} \quad (14)$$

$$= max\{d^T y \mid y \in \Re^E, y \text{ satisfies (9), ..., (12)}\} \quad (15)$$

where the vector d has components $d_{nt} = c_{nt}$, $d_{kt} = c_{kt} - c_{k+1,t}$ for $t \leq k \leq n-1$, $1 \leq t \leq p$. Any optimal solution to (15) is translated to an optimal solution to (OP) via (7) and (8).

Corollary 1 *Problem (OP) can be solved in strongly polynomial time.*

Let Y be the image of X under the transformation (6), i.e. $Y = \{y \in \Re^E | y \text{ satisfies } (9), \ldots, (12)\}$. Denote $H = (E, F)$ the OPA network associated with the linear program (15) where the nodes E of H correspond to the edges of the graph G, the arc set F is given by (9) and (10) and rather than having edge weights we have the node weights defined above. We define a *cut* $(S : E - S)$ in H to be *feasible* if $S = \emptyset$ or $S = E$ or if $\emptyset \neq S \subset E$ then (i) $(n, 1) \in S$, $(p, p) \notin S$ and (ii) for all arcs $a = (h_a, t_a) \in (S : E - S)$ $h_a \notin S$, $t_a \in S$, where h_a stands for "head" and t_a for "tail" of the arc a. The "tail" t_a of an arc a corresponds to a "minus", the "head" h_a to a "plus" in the corresponding inequality (9) or (10). Furthermore, for $S \subseteq E$ we denote $H_S = (S, F_S)$ the sub-network of H with nodes in S and all arcs of F having both endpoints in S. To prove the following we use repeatedly the fact that the incidence matrix of a tree with $k \geq 1$ nodes has a rank of $k - 1$.

Proposition 5 *Every feasible cut $(S : E - S)$ in H defines an extreme point of Y and vice versa.*

The linear program (15) thus consists of finding a *maximal weighted feasible cut* in the network H where the weight of a cut $(S : E - S)$ is given by $\sum_{e \in S} d_e$.

Two extreme points $y^1 \neq y^2$ of a polyhedron P are called *adjacent* if the face of minimal dimension of P that contains both y^1 and y^2 has a dimension of one. For any two extreme points $y^1 \neq y^2$ of P denote $d(y^1, y^2)$ the *smallest* number of faces of dimension one of P that – in a simplex-type algorithm – must be "traversed" in order to get from y^1 to y^2. The *diameter* of a polyhedron P denoted *diam(P)* is the maximum of $d(y^1, y^2)$ over all pairs of extreme points y^1 and y^2 of P.

Proposition 6 *Let y^0, y^1 be the two extreme points of Y corresponding to $S = \emptyset$, $S = E$ respectively.*
(i) y^0 and y^1 are adjacent on Y and every other extreme point y of Y is adjacent to both of them.
(ii) Two extreme points $y^2 \neq y^3$ of Y, both different from y^0 and y^1, are adjacent if and only if $S \subset T$ and H_{T-S} is connected or $T \subset S$ and H_{S-T} is connected where $S = \{(k,t) \in E \mid y^2_{kt} = 1\}$ and $T = \{(k,t) \in E \mid y^3_{kt} = 1\}$.
(iii) $diam(Y) = 2$ for all $n \geq 3$ and $p \geq 2$.

For any order preserving assignment $A \subseteq E$ let p_A be the *last* position assigned by A. $A = \emptyset$ is called the *trivial* and $A = \{(1,1), \ldots, (p,p)\}$ the *canonical OPA*. An *OPA B dominates* the *OPA A* if $p_B \geq p_A$ and there is a position π such that for every $(k,t) \in A$ there exists $h < k$ such that $(h,t) \in B$ for all $1 \leq t \leq \pi$ and for all $\pi + 1 \leq t \leq p_A$ $(k,t) \in A$ implies $(k,t) \in B$ where $1 \leq \pi \leq p_A$. Note that the canonical OPA dominates *all* (nontrivial) *OPAs*. We say that two *OPAs* are *adjacent* if their incidence vectors x^A, x^B are a pair of adjacent extreme points of OP^n_p.

Theorem 2 *(i) The trivial and the canonical OPAs are adjacent and every other OPA is adjacent to both of them.*
(ii) Two nontrivial, noncanonical OPAs A and B with $p_A \leq p_B$ are adjacent if and only if B dominates A.
(iii) The diameter of OP^n_p equals two for all $n \geq 3$ and $p \geq 2$.

The proof of Theorem 2 follows directly from Proposition 6 since the transformation T given by (6) is nonsingular and by interpreting the condition for adjacency given by Proposition 6 on the bipartite graph $G = (N, P, E)$.

4 Assigning exactly p positions

To deal with the problem of assigning exactly p positions in an order preserving manner we use zero-one variables x_{jt} as before, but here $t \le j \le n-p+t$ and $1 \le t \le p$ as we are working in a lower-dimensional space. The exactness of the assignment implies that every zero-one vector $x \in \Re^D$ that is the incidence vector of such an OPA satisfies

$$\sum_{j=t}^{n-p+t} x_{jt} = 1 \quad 1 \le t \le p\,. \tag{16}$$

Proposition 7 *$dim OP^{=}_{np} = |D| - p = p(n-p)$ and $aff(OP^{=}_{np}) = \{x \in \Re^D \mid x$ satisfies (16)$\}$.*

Consider now the inequalities

$$\sum_{j=t+1}^{k+1} x_{j,t+1} \le \sum_{j=t}^{k} x_{jt} \quad t \le k \le n-p+t-1\,,\ 1 \le t \le p-1 \tag{17}$$

$$x_{pp} \ge 0\,, \quad x_{n-p+1,1} \ge 0\,, \quad x_{jt} \ge 0 \quad t+1 \le j \le n-p+t-1\,,\ 1 \le t \le p\,. \tag{18}$$

Theorem 3 *$OP^{=}_{np} = \{x \in \Re^D \mid x$ satisfies (16), (17) and (18)$\}$ for all $1 \le p \le n$. Moreover, the linear system (16), (17) and (18) is a minimal description of $OP^{=}_{np}$ if $p < n$.*

Corollary 2 *The inequalities (17) and (18) define distinct facets of $OP^{=}_{np}$ and every facet of $OP^{=}_{np}$ is defined by one of the inequalities (17) or (18).*

As the polytope $OP^{=}_{np}$ is not full dimensional, there are, of course, many different linear descriptions of $OP^{=}_{np}$ which are all equivalent modulo linear combinations of the equations (16) and multiplication by positive scalars. With respect to the optimization problem $(OP^{=})$ over the polytope $OP^{=}_{np}$ we show *mutatis mutandis* the following corollaries.

Corollary 3 *Problem $(OP^{=})$ can be solved in strongly polynomial time.*

Corollary 4 *The diameter of $OP^{=}_{np}$ equals two for all $p \ge 2$ and $p+2 \le n$.*

5 Finding optimal assignments

The dual to the linear program (15) is a minimum cost flow problem on the OPA network having a particular objective function. In fact, rather than calling our problem a minimum cost flow problem we shall refer to it as a *minimal flow problem*. We utilize this structure to give fast algorithms for the problems (OP) and $(OP^{=})$. The basic idea of our approach is to solve (15) by reducing the overall problem to a sequence of p smaller problems. An optimal solution to (OP) is then obtained using the duality theorem of linear programming. The case of $(OP^{=})$ is similar.

Denote v_{kt}, u_{kt} the dual variables associated with constraints (9) and (10) and v_{pp}, $u_{n+1,1}$ those associated with the constraints (11) and (12), respectively. At a typical node (k,t) we have the "flow" situation as illustrated in Figure 3. Consequently, the *flow conservation* equation at node (k,t) is given by

$$-u_{kt} + u_{k+1,t} + v_{k-1,t-1} - v_{kt} = d_{kt} \qquad t \le k \le n\,,\ 1 \le t \le p\,, \tag{19}$$

where at the "border" of the OPA network certain nodes do not exist, for instance the nodes $(k,0)$ for $t = 1$ and $0 \le k \le n-1$. We model the corresponding variables anyway and require them to be zero in feasible solutions and the remaining ones to be nonnegative.

$$u_{11} = u_{tt} = u_{n+1,t} = v_{n,t-1} = 0 \qquad 2 \le t \le p \tag{20}$$

$$v_{k0} = 0 \;\; 0 \le k \le n-1\,, \quad v_{kp} = 0 \;\; p+1 \le k \le n \tag{21}$$

$$u_{n+1,1} \ge 0\,,\; u_{kt} \ge 0\,,\; v_{kt} \ge 0 \qquad t \le k \le n\,,\; 1 \le t \le p\,. \tag{22}$$

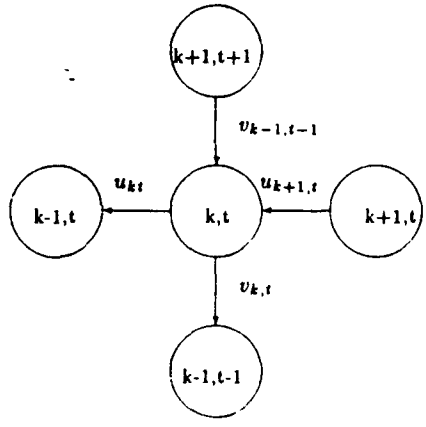


Figure 3. Flow at node (k,t).

The minimal flow problem that we have to solve is given by

$$(NET_p) \qquad \begin{array}{l} min \quad u_{n+1,1} \\ \text{subject to } (19),\ldots,(22)\,. \end{array}$$

For $1 \le q < p$ we denote (NET_q) the following *relaxation* of (NET_p):

$$(NET_q) \qquad \begin{array}{l} min \quad u_{n+1,1} \\ \text{subject to} \end{array}$$

$$-u_{kt} + u_{k+1,t} + v_{k-1,t-1} - v_{kt} = d_{kt} \quad t \le k \le n\,,\; 1 \le t \le q \tag{23}$$

$$u_{11} = u_{tt} = u_{n+1,t} = v_{n,t-1} = 0 \quad 2 \le t \le q \tag{24}$$

$$v_{nq} = v_{k0} = 0 \quad 0 \le k \le n-1 \tag{25}$$

$$u_{n+1,1} \ge 0\,,\; u_{kt} \ge 0\,,\; v_{kt} \ge 0 \quad t \le k \le n\,,\; 1 \le t \le q\,. \tag{26}$$

(NET_q) is thus a minimal flow problem on the *partial* network given by the nodes (k,t) with $t \le k \le n$, $1 \le t \le q$ and the nodes $(q+1,q+1),\ldots,(n,q+1)$ for which we have, however, no flow conservation equations and thus *no* demands $d_{k,q+1}$. Rather, the variables $v_{qq},\ldots,v_{nq}$ are "surplus" variables in the problem (NET_q). For notational convenience we write a feasible solution to (NET_q) in vector form (u^q, v^q) where we use the following *indexing* of the components of u^q

$$u^q = (u_{n1},\ldots,u_{11},u_{n2},\ldots,u_{22},u_{nq},\ldots,u_{qq}) \tag{27}$$

and likewise for v^q where $1 \le q \le p$. The "flow value" $u^q_{n+1,1}$ is kept separately from the vectors. Note that the sequential indexing of u and v is *not* by increasing sequential index (1). It follows that if (u^{q+1}, v^{q+1}) is a feasible solution to (NET_{q+1}), then the "truncated" vector (u,v), say, that is obtained from (u^{q+1}, v^{q+1}) by suppressing the $n-q$ last components in u^{q+1} and v^{q+1}, respectively, defines a feasible solution to (NET_q). Among the optimal solutions to (NET_q) we want to single out a particular one. We recall that a vector $x \in \Re^n$ is *lexicographically greater* than $y \in \Re^n$ if $x_i = y_i$ for $i = 1,\ldots,k$ and $x_{k+1} > y_{k+1}$ for some $0 \le k \le n-1$.

Definition 2 *An optimal solution* (u^q, v^q) *to* (NET_q) *is called* lex-max *if the vector* $v^q = (v^q_{n1}, \ldots, v^q_{11}, v^q_{n2}, \ldots, v^q_{22}, \ldots, v^q_{nq}, \ldots, v^q_{qq})$ *is lexicographically* maximal *among all optimal solutions to* (NET_q) *where* $1 \le q \le p$.

Proposition 8 *Problem* (NET_q) *has a unique lex-max solution for* $1 \le q \le p$.

Proposition 9 *A feasible solution* (u^q, v^q) *to* (NET_q) *is lex-max if and only if*

$$u^q_{k+1,t} v^q_{kt} = 0 \quad t \le k \le n-2\,,\ 1 \le t \le q\,. \tag{28}$$

Proposition 10 *If* (u^q, v^q) *and* (u^{q+1}, v^{q+1}) *are lex-max solutions to* (NET_q) *and* (NET_{q+1}), *then*

$$u^{q+1}_{kt} = u^q_{kt} \qquad t \le k \le n-1\,,\ 1 \le t \le q \tag{29}$$

$$v^{q+1}_{kt} = v^q_{kt} \qquad t \le k \le n-2\,,\ 1 \le t \le q \tag{30}$$

$$u^{q+1}_{nt} \ge u^q_{nt}\,,\ v^{q+1}_{n-1,t} \ge v^q_{n-1,t} \qquad 1 \le t \le q\,. \tag{31}$$

To formulate a procedure that finds a lex-max solution (u^{q+1}, v^{q+1}) to (NET_{q+1}) given a lex-max solution (u^q, v^q) to (NET_q) we need the following two *auxiliary* problems. In the first problem we want to find a lex-max solution (u, v) which solves the problem

$$\begin{array}{lll} & \min & u_{n+1} \\ (P_{q+1}) & \text{subject to} & -u_k + u_{k+1} - v_k = d_k \qquad q+1 \le k \le n \\ & & u_{q+1} = v_n = 0\,,\ u_k\,,\ v_k \ge 0\,, \end{array}$$

i.e. a solution such that the vector $v = (v_n, \ldots, v_{q+1})$ is lexicographically maximal among all optimal solutions to (P_{q+1}). The second problem is to find an optimal solution u which solves the problem

$$\begin{array}{lll} & \min & u_{n+1} \\ \text{(P2)} & \text{subject to} & -u_k + u_{k+1} = d_k\,,\ u_k \ge 0 \quad p \le k \le n\,. \end{array}$$

In the following procedure u, v and d are assumed to be vector arrays of sixe $|E|$ that are indexed as in (34) and the arrays D, U, V are of size $n+1$ and are "local" variables. Instructions that are separated by a colon are to be interpreted sequentially (left to right). "FLO" is the flow value $u_{n+1,1}$.

Procedure LEXMAX (n, p, q, u, v, d, FLO)

Input: (u^q, v^q) *a lex-max solution to* (NET_q) *with flow value* FLO, $d_{k,q+1} \in Z$ *for* $q+1 \le k \le n$ *and* $0 \le q < p$.
Output: (u^{q+1}, v^{q+1}) *a lex-max solution to* (NET_{q+1}) *with flow value* FLO.

```
Step 1:  if q = 0 then
            do for k = 1 to n
               D_k := d_k1
            enddo
            FLO := 0
         else
            do for k = q + 1 to n
               D_k := d_k,q+1 − v_k−1,q
            enddo
         endif
Step 2:  if q + 1 < p then
```

call procedure P1(n, q, D, U, V)
else
call procedure P2(n, p, D, U, V)
endif

Step 3: **do for** $k = q + 1$ **to** n
$u_{k,q+1} := U_k$; $v_{k,q+1} := V_k$
enddo
do for $t = 1$ **to** q
$u_{nt} := u_{nt} + U_{n+1}$; $v_{n-1,t} := v_{n-1,t} + U_{n+1}$
enddo
$FLO = FLO + U_{n+1}$
return.

The procedure LEXMAX reduces the problem of solving the minimal flow problem (NET_p) to a sequence of optimization of problems (P_{q+1}) for $0 \leq q \leq p-2$ and one optimization problem $(P2)$, all of which can be solved fast by the following two procedures.

Procedure P1 (n, q, d, u, v)

Input: $d_k \in \mathcal{Z}$ *for* $q + 1 \leq k \leq n$.
Output: *a lex-max solution* (u, v) *to* (P_{q+1}).
$u_{q+1} := 0$
do for $i = q + 1$ **to** $n - 2$
$u_{i+1} := max\{0, d_i + u_i\}$; $v_i := u_{i+1} - u_i - d_i$
enddo
$u_n := max\{0, -d_n, d_{n-1} + u_{n-1}\}$; $v_{n-1} := u_n - u_{n-1} - d_{n-1}$
$u_{n+1} := d_n + u_n$; $v_n := 0$
return.

It is a routine matter to check that procedure P1 returns a feasible solution (u, v) to (P_{q+1}). Moreover, $u_{i+1} > 0$ implies $v_i = 0$ and $v_i > 0$ implies $u_{i+1} = 0$ in the solution for $q + 1 \leq i \leq n - 2$. Since (P_{q+1}) is the problem (NET_1), except that we have fewer variables, Proposition 9 applies with the necessary changes and thus the solution returned by procedure P1 is lex-max. Procedure P1 requires $O(n - q)$ time.

Procedure P2 (n, p, d, u, v)

Input: $d_k \in \mathcal{Z}$ *for* $p \leq k \leq n$.
Output: *an optimal solution* u *to* $(P2)$ *and a vector* v.
$u_p := 0$
do for $j = p$ **to** n
$u_{j+1} := u_j + d_j$; $v_j := 0$
if $-u_{j+1} > u_p$ **then** $u_p := -u_{j+1}$
enddo
do for $j = p + 1$ **to** $n + 1$
$u_j := u_j + u_p$
enddo
$v_p := u_p$; $u_p := 0$
return.

The interchange of v_q and u_p is done in procedure P2 to facilitate the updating in the procedure LEXMAX. With $u_p = v_p$ the solution returned by procedure P2 is feasible for $(P2)$. To see that it is

optimal eliminate all variables except u_p. The remainder is a trivial linear program in the variable u_p with the value as given in the procedure. Procedure P2 requires $O(n-p)$ time.

Proposition 11 *Procedure LEXMAX returns a lex-max solution to (NET_{q+1}) in $O(n)$ time and a lex-max solution to (NET_p) can be found in $O(np)$ time.*

We are now ready to formulate an algorithm that solves (OP). In the following algorithm we assume that u, v, c and d are arrays of length $|E|$, w is an array of length p and that the rest are scalars. All auxiliary arrays and scalars are initialized at zero, the user supplies the data n, p and $c \in \mathcal{Z}^E$ and provides his own output statements.

Algorithm OPA (n, p, c)

Input: $n \in \mathcal{N}$, $p \in \mathcal{N}$, $1 \leq p \leq n$, $c \in \mathcal{Z}^E$.
Output: *an optimal solution vector x for (OP).*
Step 0: **do for** $t = 1$ **to** p
  $d_{nt} := c_{nt}$
  do for $k = t$ **to** $n-1$
    $d_{kt} := c_{kt} - c_{k+1,t}$
  enddo
enddo
Step 1: **do for** $q = 0$ **to** $p-1$
  call LEXMAX(n, p, q, u, v, d, FLO)
enddo
if $FLO = 0$ **stop.**
$q := 1$
Step 2: **if** $u_{nq} = 0$ **go to** Step 3 ; **if** $v_{n-1,q} = 0$ **go to** Step 3 ; $q := q+1$; **if** $q < p$ **go to** Step 2
Step 3: $t := q+1$; $w_t := n+1$
Step 4: $t := t-1$; **if** $t = 0$ **go to** Step 6 ; $w_t := w_{t+1}$
Step 5: $w_t := w_t - 1$; **if** $u_{w_t t} = 0$ **go to** Step 4 ; **go to** Step 5
Step 6: **do for** $t = 1$ **to** q
  $x_{w_t t} = 1$
enddo
stop.

Theorem 4 *Algorithm* OPA *finds an optimal order preserving assignment in $O(np)$ time.*

We obtain an algorithm for $(OP^=)$ by slightly modifying the algorithm OPA and thus the corolary.

Corollary 5 *Algorithm* OPA$^=$ *finds an optimal solution to $(OP^=)$ in $O((n-p)p)$ time.*

6 Some extensions

Suppose that we would like to include a cost of using a position, but that we otherwise insist on order preserving assignments. So let $z_t = 1$ if position t is used, $z_t = 0$ otherwise, and f_t be the cost of using position t, where $1 \leq t \leq p$. Then the linear programming formulation for costly positions becomes

$$max \sum_{t=1}^{p} \sum_{k=t}^{n} c_{kt} x_{kt} - \sum_{t=1}^{p} f_t z_t$$

$$\text{subject to (2), (3), (5) and } \sum_{j=t}^{n} x_{jt} = z_t \text{ for } t = 1, \ldots, p.$$

One proves the implicit assertion that all extreme points (x, z) of this linear program are zero-one valued by showing that for all integer $c \in \mathcal{Z}^E$ and $f \in \mathcal{Z}^p$ the objective function value is an integer number. To do so, it suffices to eliminate the variables z_t from the problem and to reduce the above problem to (OP) with a changed vector c. Of course, by Proposition 2 the z_t variables will be either zero or one. So, in particular, there is also another proof by discussing *feasible bases* for the problem and using Theorem 1.

Another variation that we have considered is the following. Suppose we want an order preserving assignment such that at least one of the q first items gets assigned to position 1. One proves by the methods of Section 3 that intersecting OP_p^n with the single constraint $\sum_{j=1}^{q} x_{j1} = 1$ does the job, i.e. the resulting polytope has zero-one valued extreme points only. Of course, the variables $x_{q+1,1}, \ldots, x_{n,1}$ must be zero in every solution and can be dropped from the problem. It is also not difficult to determine a minimal description for the resulting problem. If we intersect OP_p^n with several constraints of the above or a similar form we can, of course, *not* expect to get a polytope with zero-one extreme points only.

Note: Following the presentation of the original paper at the ORSA/TIMS Joint National meeting in October 1991, Maurice Qeyranne (University of British Columbia) and independently, Arie Tamir (Technion, Tel Aviv) have found direct "combinatorial" algorithms – based on dynamic programming – for the solution of the optimization problem (OP) that have the same running time as the algorithm proposed by us.

References

[1] R.K.Ahuja, T.L.Magnanti and J.B.Orlin "Some recent advances in network flows", *SIAM Review*, **33**, 175-219 (1991).

[2] M.L.Balinski "Signatures des points extrêmes du polyèdre dual du problème de transport", *Compt. Rend. Acad. Sci. Paris*, **296**, Ser.I 456-459 (1983).

[3] M.L.Balinski "Signature methods for the assignment problem", *Operations Research*, **33**, 527-536 (1985).

[4] M.L.Balinski and A.Russakoff "Some properties of the assignment problem", *Mathematical Programming*, **3**, 257-258 (1972).

[5] W.Cook "Operations that preserve total dual integrality" *Operations Research Letters*, **2**, 31-35 (1983).

[6] M.Grötschel and M.Padberg "Polyhedral theory", in: E.L.Lawler, J.K.Lenstra, A.H.G.Rinnoy Kan and D.Shmoys (eds.), *The Traveling Salesman Problem*, Wiley, Chichester, 251-305 (1985).

[7] H.W. Kuhn "The Hungarian method for the assignment problem" , *Naval Research Logistics Quarterly*, **2**, 83-97 (1955).

[8] J. Munkres "Algorithms for the assignment and transportation problems", *Journal of the Society for Industrial and Applied Mathematics*, **5**, 32-38 (1957).

[9] M. Padberg and D. Alevras "Order preserving assignments", Working paper SOR-91-19, New York University.

Nonlinear Min–max Optimization via Smooth Penalty Functions

Mustafa Ç. Pınar
Department of Numerical Analysis
The Technical University of Denmark
Lyngby, DK 2800, Denmark.

Stavros A. Zenios
Decision Sciences Department
University of Pennsylvania
Philadelphia, PA 19104.

November 16, 1992

In this report we are concerned with the numerical solution of the nonlinear min–max problems using smooth penalty functions. The min–max problem is stated as follows:

$$\min_{x \in \Re^n} \max_{i \in \mathcal{M}} f_i(x) \tag{1}$$

where the $f_i : \Re^n \mapsto \Re$ are continuously differentiable functions and $\mathcal{M} = \{1, \ldots, m\}$. The min–max problem is known to be equivalent to the following nonlinear program:

[MINMAX]

$$\begin{aligned} \underset{x,z}{\text{minimize}} \quad & z \\ \text{subject to} \quad & f_i(x) \leq z \ \forall \ \ i = 1, \ldots, m \end{aligned}$$

We propose to compute an approximate solution to problem [MINMAX] by solving the following problem:

[MINSEP]

$$\min_{(x,z)} z + \mu \sum_{i=1}^{m} \tilde{r}(f_i(x) - z) \tag{2}$$

where $\tilde{r}$ is a smooth penalty function and μ is its controlling parameter. We consider two choices of the penalty function $\tilde{r}$: (1) A linear–quadratic penalty function

$$\tilde{r}(t) = \begin{cases} 0 & \text{if } t \leq 0 \\ \frac{t^2}{2\epsilon} & \text{if } 0 \leq t \leq \epsilon \\ t - \frac{\epsilon}{2} & \text{if } t \geq \epsilon \end{cases} \tag{3}$$

and, (2) A linear–logarithmic penalty function

$$\tilde{r}(t) = \begin{cases} 0 & \text{if } t \leq 0 \\ (t+a)\left[\ln(\frac{t+a}{a}) - 1\right] + a & \text{if } 0 \leq t \leq \epsilon. \\ t - (\epsilon - a) & \text{if } t \geq \epsilon. \end{cases} \tag{4}$$

The penalty functions (3)–(4) can be obtained as smooth approximations to the ℓ_1 penalty function

$$r(t) = \max\{0, t\}. \tag{5}$$

We note that the penalty functions (3)–(4) have continuous first derivatives whereas the ℓ_1 penalty function does not. The parameter ϵ controls the accuracy of the approximation. We give the following definition before we proceed:

Definition A vector $(\tilde{x}, \tilde{z})$ is ϵ-feasible if

$$f_i(\tilde{x}) \leq \tilde{z} + \epsilon \qquad \forall \ i = 1, \ldots, m.$$

We established in [5] that the threshold value of the penalty parameter μ required in order to achieve ϵ-feasibility is a function of the maximum of the Lagrange multipliers to the original problem. However, the Lagrange multipliers associated with the inequalities in problem [**MINMAX**] are in the interval $[0, 1]$. This fact is readily verified from the first order Karush-Kuhn-Tucker optimality conditions for [**MINMAX**]. As a result of this observation in an iterative scheme where the solution of problem [**MINMAX**] is approximately obtained by solving a finite number of penalty problems one can start with a value of μ in the range $[0, 1]$. This property enables one to alleviate the problems associated by forcing too much ill-conditioning on the penalty problem from the beginning. We construct such an iterative scheme below to solve min–max problems by means of unconstrained minimizations. To start the algorithm, an initial point $(\tilde{x}_0, z_0)$ is needed and initial values μ_0, ϵ_0 for the penalty parameter μ and the smoothing parameter ϵ need to be specified. We consider the following iterative steps:

Step 1 Using the violation $(\tilde{x}_k, \tilde{z}_k)$ as the starting point for evaluating the penalty function solve the problem [**MINSEP**]. Let $(\tilde{x}_{k+1}, \tilde{z}_{k+1})$ denote the optimal solution.

Step 2 If $(\tilde{x}_{k+1}, \tilde{z}_{k+1})$ is ϵ-feasible and $\epsilon \leq \epsilon_{fin}$ terminate. Otherwise, update the penalty parameter μ and and the accuracy parameter ϵ, set $k \leftarrow k + 1$ and proceed from Step 1.

The parameter ϵ_{fin} is a final infeasibility tolerance. At Step 2, we update the penalty parameter μ and the smoothing parameter ϵ as follows:

$$\epsilon_{k+1} = \eta_1 \max \epsilon^k$$

and

$$\mu_{k+1} = \eta_2 \mu_k$$

where $\eta_1 = 0.1$ and $\eta_2 = 2.0$. The above choices were found to work best throughout the computational tests performed. Obviously the effectiveness of the iterative scheme is contingent upon the success of the unconstrained minimizations in Step 1. We use a modified Newton method due to Gay [2] to solve the penalty problem. The modified Newton algorithm iterates by computing an optimal locally constrained step with a trust region on the step. It uses a finite-difference estimation of the Hessian. It is available as a subroutine in the IMSL Library.

We summarize our computational experience using a test problem from [3].

$$\min_x \max_{i=1,\ldots,m} f_i(x)$$

with

$$\begin{aligned}
f_1(x) &= x_1^2 + x_2 + x_3 + \cdots + x_n - 1, \\
f_2(x) &= x_1 + x_2^2 + x_3 + \cdots + x_n - 1, \\
f_3(x) &= x_1 + 2x_2^2 + x_3 + \cdots + x_n - 1, \\
&\cdots \\
\left.\begin{aligned} f_i(x) &= x_1 + x_2 + x_3 + \cdots + x_{\frac{i+2}{2}}^2 + \cdots + x_n - 1 \\ f_{i+1}(x) &= x_1 + x_2 + x_3 + \cdots + 2x_{\frac{i+2}{2}}^2 + \cdots + x_n - 1 \end{aligned}\right\} & \; i \text{ even} \\
f_m(x) &= x_1 + x_2 + x_3 + \cdots + x_n^2 - 1,
\end{aligned}$$

where $m = 2n-2$, n even. We have run the example with $n = 4, 10, 20, 30, 40, 60, 70$, and 80.

This problem was chosen to test the behavior of the algorithm on problem with many variables and functions. We also solved some of these problems using (1) The nonlinear programming software MINOS 5.1 of Murtagh and Saunders [4] to verify the accuracy of our solutions, and (2) The recently developed nonlinear programming software Lancelot of Conn et al. [1]. The results are reported in Table 5. The starting point in all runs was taken to be $(10, 10, \ldots, 0)$.

(n, m)	LQP					MINOS	Lancelot	
	MIT	NIT	NF	NG	NH	NF	NF	NG
(4,6)	4	33	59	37	33	NA	50	44
(10,18)	4	43	81	47	43	96	196	166
(20,38)	3	27	54	30	27	175	288	247
(30,58)	3	43	69	46	43	249	288	249
(40,78)	3	39	81	42	39	NA	362	349
(60,118)	3	40	82	43	40	NA	465	418
(70,138)	3	41	98	44	41	NA	533	478
(80,158)	3	25	68	28	25	NA	656	581

Table 5: Solution Statistics for the Minimax using the Linear-Quadratic Penalty (LQP), MINOS and Lancelot. (MIT = Number of Penalty Minimizations, NIT = Number of Newton Iterations, NF = Number of Function Evaluations, NG = number of Gradient Evaluations, NH = Number of Hessian Evaluations, NA = Not Available)

We note that all three codes were run with default parameters. We do not report the solution computed by the LQP algorithm since we find that we have on the average four digits of accuracy with respect to the MINOS

solution. The accuracy of the solution reported by Lancelot is controlled using the parameter **gradient_tolerance**. In our experiments its value is set to 10^{-3}

References

[1] Conn, A.R., N.I.M. Gould and Ph. L. Toint. 1992. *Lancelot: A Fortran Package for Large-Scale Nonlinear Optimization (Release A).* Springer-Verlag, Heidelberg, Berlin, New York.

[2] Gay, D.M. 1983. Algorithm 611: Subroutine for Unconstrained Optimization using a Model Trust Region Approach. *ACM Transactions on Mathematical Software* 9, pp. 503–524.

[3] Madsen, K. and H. Schjær-Jacobsen. 1978. Linearly Constrained Minimax Optimization. *Mathematical Programming* 14, pp. 208–223.

[4] Murtagh, B.A. and M.A. Saunders. 1987. MINOS 5.1 User's Guide. Report SOL 83–20R, December 1983, revised January 1987, Stanford University.

[5] Pınar, M.Ç. and S.A. Zenios. 1992. Smoothing Techniques for a Class of Exact Penalty Functions and Nonlinear Min–max Optimization. Report 92–05–02. Decision Sciences Department. University of Pennsylvania. Philadelphia PA 19104 (in preparation).

THE ESIS PROJECT: AN INTELLIGENT DECISION SUPPORT SYSTEM FOR ASSISTING INDUSTRIAL WASTE MANAGEMENT

- Extended abstract -

János Pintér, Dirk J. Meeuwig and Jay W. Meeuwig
School for Resource and Environmental Studies, Dalhousie University
1312 Robie Street, Halifax, Nova Scotia B3H 3E2, CANADA

Mort Fels and David S. Lycon
Department of Chemical Engineering, Technical University of Nova Scotia
P.O. Box 1000, Halifax, Nova Scotia B3J 2X4, CANADA

The objective of the Environmentally Sensitive Investment System (ESIS) Project is to provide industry, as well as government departments and agencies, with possible means to assess the environmental and economic implications of capital-intensive projects and policies. More specifically, ESIS helps to find wastewater management alternatives that meet stated environmental regulatory standards in a technologically sound and cost-efficient manner. The use of this intelligent decision support system will enhance the ability of managers and planners to explore the quantitative implications of a wide range of options.

ESIS incorporates a combination of artificial intelligence, expert system and operations research techniques, database management and graphic presentation tools, integrated into a user friendly, dialog and menu driven software system. ESIS is targeted primarily for top-of-the-line personal computers and, possibly at a later stage, for workstations.

Keywords: environment-economy integration, intelligent decision support systems, pulp and paper industry, waste management.

1. Introduction

Major industries - such as mining, chemical production, pulp and paper manufacturing, food processing, etc. - typically have a significant negative impact on the ambient environmental quality. These industries face a serious challenge (expressed by a growing public concern and more stringent environmental legislation) to find efficient methods of waste treatment and disposal, while acting under the financial constraints of market realities.

ESIS, the Environmentally Sensitive Investment System, is designed to provide quantitative assistance in this complex decision making process. Specifically, ESIS assists in selecting waste water management options that meet jointly considered technical and economic constraints, as well as environmental regulatory criteria in an efficient manner. With the help of ESIS, senior industry and government decision makers will be able to systematically analyze and compare strategic investment and operational choices in an interactive, computer-assisted environment.

Although the ESIS prototype is initially focused on the economically mature Canadian pulp and paper industry - more precisely, mechanical pulping (TMP/CTMP) mills - its broader conceptual relevance is evident with respect to a number of capital-intensive industries that have a potentially large, negative environmental impact. Several possible extensions and generalizations of the prototype system will be highlighted later.

The ESIS Project is supported by a consortium of industry, consulting, research and government partners which represents a range of interests, expertise and a variety of multi-disciplinary professional assistance to the Project.

2. Environment-Economy Integration: The ESIS Model Structure

As stated above, ESIS will be a quantitative decision support tool that assists in finding harmonized economic-environmental policies. This objective is reflected by the generic system scheme shown in Figure 1: waste management in the pulp and paper industry can be considered as one of its possible realizations.

Figure 1 indicates that there are several system components which allow for implementing diverse types of management and control options. Because of the adopted scope limitations of ESIS, a particular emphasis is placed on the selection of technologically feasible, environmentally satisfactory and economically efficient waste water treatment alternatives. An attempt is, however, made to include - at least on the level of quantitative sensitivity analysis - some other management opportunities: plan modifications, material recycling and reuse options, non-standard waste treatment and disposal practices, and adaptively chosen environmental regulations (effluent quality standards).

3. Waste Water Treatment Engineering System Model

The waste water treatment systems considered in the ESIS prototype are determined by a given configuration of unit operations applicable to mechanical pulp and paper mills. Figure 2 depicts the scheme of an investigated treatment system.

The following unit treatment processes (UTP's) are included in ESIS (their role will be briefly described in the full version of the paper):

- primary clarifier
- spill basin
- equalization basin
- activated sludge treatment

- aerated lagoon
- secondary clarifier
- sludge mixer
- sludge thickener
- sludge dewatering

For each UTP, a quantitative steady-state description has been developed which makes possible the formulation of a corresponding mathematical system model of complete treatment configurations. The input of this system is basically determined by the pulp and paper manufacturing process which results in a given production level and corresponding generated raw waste stream. The output (treated) waste stream affects the spatial and temporal evolution of the ambient environmental quality.

In the ESIS prototype four selected basic waste water treatment configurations can be investigated. The pollution removal efficiency and cost of each UTP are governed by choosing a few (2-6) principal design variables and operational characteristics. Hence, finding feasible or best investment and operational decisions necessitates the simultaneous choice of several tens of decision variables.

4. Analytical Optimization Model Formulations

Environmentally sensible planning means the integral consideration of multiple (partially conflicting and non-commensurable) objectives, criteria and aspects, which include those of engineering (technical feasibility), economics (cost-efficiency of design and operations) and environment ("satisfactory" quality, sustainable development). This inherently "soft" problem structure excludes the possibility of finding a unique best representation, as reflected by a single mathematical programming model. Therefore the verbal problem statement below is only a possible formulation of the basic ESIS paradigm.

ESIS Optimization Model Frame

OBJECTIVE

{find the most cost-efficient combination of unit waste water treatment processes}

STATE EQUATIONS AND CONSTRAINTS

{production process driven initial conditions}
{input waste stream characteristics}
{explicit ranges of decision variables: waste water treatment equipment types and sizes}
{explicit ranges of decision variables: basic operational characteristics of UTP's}
{implicit technological constraints and relations between UTP's}
{analytic description of the waste water treatment process}
{implied resource (construction, operation and maintenance) demands of the waste water treatment system}
{resulting effluent and solid waste stream}
{effects of waste output on environment}
{environmental quality standards}

The concrete numerical realization(s) of the above optimization paradigm typically involve several tens of decision variables and constraints, plus hundreds of constants and parameters. In addition, essential model features include complicated ("black box" like) structure, nonlinearity, absence of analytical derivatives of a number of implicit constraints, significant uncertainties and fluctuations (stochasticity) of system behaviour, and - as a direct consequence of the above facts - computationally intensive evaluation of the engineering system optimization model. In the terminology of mathematical programming, even the deterministic base model versions can merely be classified as (potentially)

multiextremal Lipschitz optimization problems, with no obvious possibility of directly justifying a more narrow model-class. Summing up these observations, the exact finding of the (mathematically) optimal decision, or better, of a "menu" of alternative efficient decisions, typically leads to complicated optimization issues.

5. Solution Approaches

Given the spectrum of targeted applications (from obtaining fast, preliminary evaluation of system performance to detailed analysis and "fine tuning"), ESIS offers a number of tools that can be applied to formulate, investigate and solve system models of diverse levels and depths. These techniques include

- knowledge-based reasoning and spreadsheet calculations
- nonlinear programming model versions and optimization techniques
- statistical uncertainty analysis and stochastic model extensions

The solution approaches listed above are capable to provide response pertinent to a large variety of problem statements. The information obtained in the different search modes is communicated to the user via a dialogue-based, seamless interface which also enables the adaptive, sequential application of the solution techniques. (Illustrative test results obtained by applying these solution approaches will be provided in the full version of the paper.)

6. Implementation Aspects

The combination of optimization, statistical analysis, expert system, database management, and visualization concepts and techniques is realized applying a modular approach. This structure makes possible the adaptive decomposition, aggregation and substitution of the ESIS

software components. (The parallel testing of different development versions still continues.) The current prototype implementation runs on top-of-the-line personal computers (IBM PS/2 386 and 486 machines). A workstation (IBM RISC 6000) development is being proposed for a subsequent stage.

7. Conclusions and Further Research

The increasing concern for environmentally responsible waste management indicates that the ESIS concept and its implementation have remarkable potential. The immediate future work includes additional testing and improvement of the prototype, and further contacts with pulp and paper mill personnel, suppliers and equipment manufacturers, in order to obtain additonal, site-specific information. This work will also be related to supplementing ESIS with in-plant process modification options and novel waste treatment/disposal/processing/use techniques.

The currently active features of ESIS include an information base on environmentally sensitive planning (in the context of the pulp and paper industry); feasibility analysis of (pre-)selected management options; preliminary screening and optimization of treatment configurations; quantification of system uncertainties and fluctuations; statistical verification (ex-post analysis) of selected management options; and report writing (executive summary and detailed technical report levels).

Possible future extensions and generalizations can lead to site-specific implementations of the ESIS concept; refinements of the environmental impact analysis; adaptation of the ESIS concept to other industries; and educational/professional training versions.

(Acknowledgements, appendix and an extensive reference list will be provided in the complete version of the paper.)

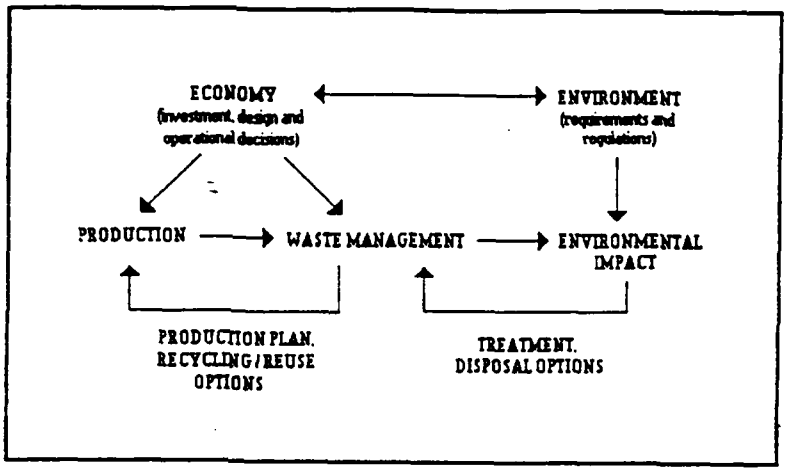


Figure 1. Environment-Economy Integration in ESIS

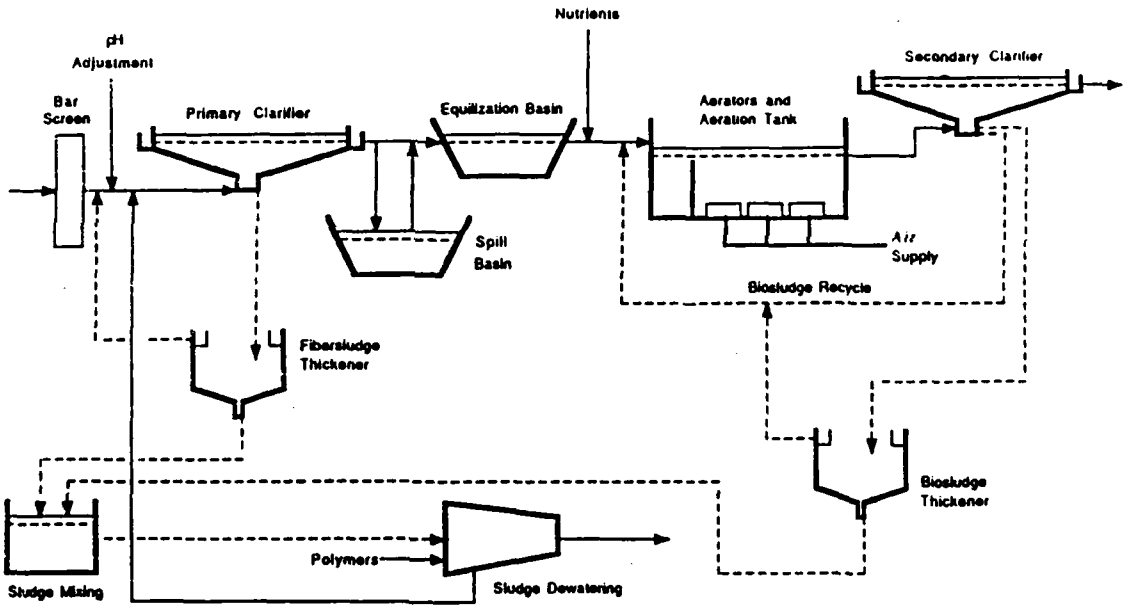


Figure 2. A Waste Water Treatment System Configuration, Applicable in Mechanical Pulp and Paper Mills

Forcing a Vertex Optimal Solution in Interior Point Methods Using an Auxiliary Function

Kumaraswamy Ponnambalam
Department of Systems Design Engineering
and
Anthony Vannelli
Department of Electrical and Computer Engineering
University of Waterloo, WATERLOO, Ontario
Canada N2L 3G1

Abstract: In interior point methods it is generally not possible to achieve a vertex optimal solution for degenerate linear programming problems. Two of the most popular methods for attempting to achieve a vertex (near) optimal solution are the method of random perturbations and the method of controlled random perturbations. When a basis needs to be recovered in addition, there is need for a number of Gaussian elimination steps which depends on the degree of degeneracy. In this paper, an alternative method to force a vertex solution using the dual affine method is presented. The dual affine method is an interior point method which is computationally efficient but not commonly used when a basis needs to be recovered. The proposed method uses a quadratic auxiliary function based on the properties of convex sets and does not require any random perturbations. Results from test problems are presented and compared with an existing method.

1. INTRODUCTION

Interior point methods are now proven to be computationally efficient for many classes of large linear programming problems, Lustig et al. (1992). However, for degenerate linear programming problems the solution from interior point methods converge to a relative interior of the optimal face providing an optimal solution with a *minimal* number of zero coordinates in the solution. When compared with this solution, a basic (vertex) solution available from simplex method has a *maximal* number of zero coordinates, a solution which may sometimes be more favourable for certain operational reasons. In addition, for some integer programming problems as well as, for the reasons of sensitivity analysis, parametric programming, and for providing warm starts in some nonlinear programming algorithms, it may be necessary to have a basic optimal solution. The primal-dual interior point methods are usually favoured for recovering a basic solution, Megiddo (1988) and Mehrotra (1989). However, there have been successful

attempts in using the dual affine method for basis recovery as well, for example, in Ponnambalam and Vannelli (1989), and Ponnambalam, et al.(1992).

Basis recovery procedure is expected to be more efficient if a vertex solution is available at least for either the primal or the dual or preferably for both. In addition to achieving a vertex optimal solution, when the basic set has to be determined it is necessary to perform an uncertain number of Gaussian elimination steps which depend on the degeneracy of the problem. The two methods that are popular for forcing a vertex solution using interior point methods are (i) perturbation method, for example, as in Ponnambalam and Vannelli (1989), and (ii) controlled perturbation method of Mehrotra (1989). Both methods depend on random perturbations and there is some uncertainty as to the effectiveness of these methods in achieving a vertex optimal solution. In this paper, we use a recently proposed method of Ponnambalam (1992), wherein, a quadratic auxiliary function is used to encourage a vertex solution using a dual affine method. In this proposed method, a bidimensional subspace is determined based on maximizing an auxiliary function which, in addition to the original objective function, includes terms for favouring a vertex optimal solution whose 2-norm squared has the maximum value in the optimal face. Test results are presented for comparison.

2. Linear Programs and Degeneracy

Let the primal linear programming problem be in the standard form

$$(P) \qquad \min \{c^T x \ : \ Ax = b \ , \ x \geq 0\} \tag{1}$$

where, A is an $m \times n$ matrix, b and c are m- and n- dimensional vectors, respectively, x is n-dimensional vector and assume that rank$(A) = m$. The dual of the linear program (P) is

$$(D) \qquad \max \{b^T y \ : \ A^T y + s = c \ , \ s \geq 0\} \tag{2}$$

where, y and s are m- and n- dimensional vectors, respectively. The pair of primal (P) and dual (D) problems is called **primal degenerate** if there exists a primal feasible x with less than m positive coordinates, and **dual degenerate** if there exists a dual feasible s with less than $n-m$ positive coordinates, Güler et al. (1991). At this time it is worth noting that, in the case of interior point methods and when a vertex optimal solution is required, the main concern is degeneracy at the optimal face, unlike in the simplex method, where degeneracy is of no serious concern once the optimal vertex is reached. In this paper, we propose to use the dual affine method of Dikin (1967) and Adler, et al. (1989), on the dual problem (D) where due to dual degeneracy the optimal solution is in the interior of the optimal face. We do not attempt to overcome the

primal degeneracy in the proposed algorithm as it can be resolved best during the Gaussian elimination steps.

3. Vertex Solution using Dual Affine Method

For the following sections we will use the inequality form of the dual problem.

$$(D) \qquad \max \{ b^T y \ : \ A^T y \le c \} \tag{3}$$

Using Proposition 1 we form an auxiliary function and solve a new optimization problem that will provide a vertex solution for the dual problem.

Proposition 1: In linear programming problems with dual degeneracy, there is at least one optimal vertex solution y^* with the following condition:

$$\|y^*\|_2^2 \ge \|y\|_2^2 \ , \ \text{for } \textit{all } y \textit{ in the optimal face.}$$

Proof. This property follows from the property of the convex hull that defines the feasible region in the Dual (D).

Therefore, to obtain a vertex optimal solution of the Dual the following problem needs to be solved. Solve

$$\max\{ y^T y \ : \ A^T y \le c \ , b^T y \ge z^* \} \tag{4}$$

where z^* is the solution of the dual problem in equation (3). Although at the outset the problem in equation (4) looks like a hard problem, we show how the dual affine algorithm can be modified to solve the problem with minimal additional effort.

Algorithm

Let $\mathbf{y}^0$ and γ be given such that $\mathbf{A}^T\mathbf{y}^0 < \mathbf{c}$ and $\gamma = 0.9$.
Set $k := 0$
While stopping criterion is not satisfied, *do*

$\mathbf{v}^k := \mathbf{c} - \mathbf{A}^T\mathbf{y}^k$

$\mathbf{b}_v := 2\mathbf{y}^k$

$\mathbf{D}_k := diag\ (\frac{1}{v_1^k}, \cdots, \frac{1}{v_m^k})$

$\mathbf{H} := (\mathbf{A}\mathbf{D}_k^2\mathbf{A}^T)$

$\mathbf{w}_1 := \mathbf{H}^{-1}\mathbf{b}$

$\mathbf{w}_2 := \mathbf{H}^{-1}\mathbf{b}_v$

When $\mathbf{w}_1$ and $\mathbf{w}_2$ are linearly independent

Solve LP1 and LP2 to get λ_1 and λ_2

$\mathbf{y}^{k+1} := \mathbf{y}^k + \gamma \times (\lambda_1 \mathbf{w}_1 + \lambda_2 \mathbf{w}_2)$

set $k := k+1$

end

Problems LP1 and LP2 are small linear programming problems of 2 unknowns each, respectively, and are easily solved. Similar linear programming problems have been solved in Boggs, et al (1989) to get weights for centering and affine directions in one of the many versions of the dual affine algorithm.

$$LP1 : \max \mathbf{b}^T(\lambda_1 \mathbf{w}_1 + \lambda_2 \mathbf{w}_2) \tag{5}$$

subject to

$$\mathbf{A}^T(\lambda_1 \mathbf{w}_1 + \lambda_2 \mathbf{w}_2) \leq \mathbf{v}^k$$

and

$$LP2 : \max \mathbf{b}_v{}^T(\lambda_1 \mathbf{w}_1 + \lambda_2 \mathbf{w}_2) \tag{6}$$

subject to

$$\mathbf{A}^T(\lambda_1 \mathbf{w}_1 + \lambda_2 \mathbf{w}_2) \leq \mathbf{v}^k$$

$$\mathbf{b}^T(\lambda_1 \mathbf{w}_1 + \lambda_2 \mathbf{w}_2) \geq z_\lambda^*$$

where z_λ^* is the solution of LP1. It is noted that, if $\mathbf{w}_1$ and $\mathbf{w}_2$ are linearly dependent then problems LP1 and LP2 may have unbounded solutions. The problem LP1 retains the degenerate characteristics of the original problem in equation (3); that is, parallel lines in equation (3) remain parallel in equation (5). In the case of dual degeneracy in (3), there is dual degeneracy in (5) but, the optimal face in (5) is defined by two vertex solutions only, Ponnambalam (1992). Therefore, if simplex method is used to solve problem LP1, we get a solution for λ_1 and λ_2 and we get the step length for the dual problem in equation (3). Because of LP2, we also solve the problem in equation (4) subject to $\mathbf{b}^T\mathbf{y}^k \geq z^*(k)$, where $z^*(k)$ is the objective function value found at the k^{th} iteration. It is easy to show that $z^*(k) \rightarrow z^*$ (the solution of the dual problem in equation 3) when $k \rightarrow \infty$. Thus, we get a dual optimal vertex solution using the algorithm proposed. Although the dual affine method does not seem to have a proof of global convergence for large steps, in practice, the maximum number of iterations required to solve the dual problem in

(3) is small, in the order of 20 to 80.

4. Test Problems and Results

Some preliminary testing of this algorithm has been done for small and medium size problems using the solvers of *MATLAB*^{TM}, Moler et al. (1987) for dense matrices. Due to the limitation of space, only results from solving selected problems are reported here. All problems we report here have degeneracy in dual or dual and primal. The algorithm was also tested on nondegenerate problems and had no significant difference in performance. All the problems reported here are in the dual form as in equation (3).

$$\textit{Problem}\ 1 : \max\ y_1+4y_3$$

Subject to:

$$y_1+4y_3 \le 4$$

$$y_2+4y_3 \le 4$$

$$y_1-y_2 \le 0$$

$$-y_i \le 0\ ,\ \text{for}\ \textit{all}\ i$$

Problem 1 is primal and dual degenerate and the two dual vertex solutions are (4,4,0) and (0,0,1). Starting from different feasible solutions, for example (0.1,0.2,0.1), the solution reached was (3.9996,3.9996,0.0009), a solution close enough to the vertex solution (4,4,0) and was achieved in 10 iterations with a duality gap of less than 10^{-7}. It is noted that different starting solutions could lead to different optimal vertex solutions which maximizes the 2-norm squared in at least in the local sense. The following two problems were used to test the effect of size as well as degeneracy on the performance of the algorithm.

$$\textit{Problem}\ 2 : \max \sum_{i=1}^{m} i\ .\ y_i$$

Subject to:

$$\sum_{i=1}^{m} i\ .\ y_i \le m$$

$$-y_i \leq 0, \text{ for } all\ i$$

Problem 2 has only the dual degeneracy with vertex solutions of type $(m,0,...,0)$, $(0,\frac{m}{2},0,...,0),..,(0,...,\frac{m}{i},0,...,0)$, for all i. Because of the dense solvers used, only problems of size m=5 to m=200 were tested. All problems were solved in about 10 iterations each, respectively, reaching an optimal (near)vertex solution usually of the type $(m,0,...,0)$.

$$Problem\ 3 : \max \sum_{i=1}^{m} i \,.\, y_i$$

Subject to:

$$\sum_{i=1}^{m} i \,.\, y_i \leq m$$

$$\sum_{i=1, i\neq j}^{m} i \,.\, y_i \leq m, \quad j = 1,...,m$$

$$-y_i \leq 0, \text{ for } all\ i$$

The Problem 3 differs from Problem 2 only in the second set of m constraints added to make the problem dual and primal degenerate. For sizes m=5 to m=200 the algorithm was able to converge to an optimal (near)vertex solution in approximately 16 iterations each, respectively, for all sizes, mostly to the solution of type $(m,0,...,0)$. However, it is noted that, the proximity to the vertex in Problem 3 was slightly worser than in Problem 2. This may be expected due to the very high primal-dual degeneracies present in Problem 3.

Many more degenerate and nondegenerate test problems, including the degenerate example suggested by Todd (Tuncel, 1992) were tested successfully as reported in Ponnambalam (1992). Although the solution obtained was dual feasible, optimal, and very near a vertex, the primal solution thus obtained need not be feasible. In order to obtain primal feasible solution, if necessary, the only additional step taken so far in our tests is to perform 2 to 4 additional standard dual affine iterations using only the dual affine direction. Then, a basis recovery procedure as discussed in Ponnambalam et al. (1992) was applied successfully for the above test problems requiring a small number of Gaussian elimination steps which depended on the starting point. However, further research in basis recovery as well as in the following areas are currently being actively pursued.

(i) Using a barrier function model {max $\mu b^T y+\mu^2 \sum \log(c-a_i^T y)+y^T y$} where μ is initially large and tends to zero at the end.

(ii) Using a potential function model {max $(n+1)\log(b^T y-z_L)+y^T y+\sum \log(c-a_i^T y)$} where z_L is the lower bound. In both cases (i) and (ii), the maximization is done using a two or three dimensional subspace.

(iii) Finding the step length over a 3 dimensional subspace, namely, that defined by the affine, centering, and the vertex directions.

(iv) Solving a small quadratic optimization problem as in Singh (1992) to find the step length instead of solving LP1 and LP2, and lastly,

(v) Solving the Netlib suite of problems using the proposed algorithm.

An interesting problem that may be solved using an algorithm similar to that presented here is the bi-level linear programming problem arising in many decision analysis such as Game Theory, Bi (1992).

5. Conclusions

Achieving vertex optimal solutions in interior point methods is expected to aid in the faster recovery of basis set in linear programming applications. The algorithm presented here has potential for achieving such an objective. Further theoretical and practical work in directions suggested here have interesting implications for many types of optimization problems including the multi-objective optimization.

References

Adler, I., M.G.C. Resende, G. Veiga and N. Karmarkar, 1989. An implementation of Karmarkar's algorithm for linear programming, *Math. Programming*, 44, 297-335.

Bi, Z., 1992. Numerical Methods for Bilevel Programming Problems, Ph.D. Thesis, Dept. of Systems Design Engg., Univ. of Waterloo.

Boggs, P.T., P.D. Domich, J.R. Donaldson, and C. Witzgall, 1989. Algorithmic enhancements to the method of centres for Linear Programming problems, NISTIR 89-4225, Center for Computing and Applied Mathematics, U.S. Department of Commerce, Gaithersburg, MD 20899.

Dikin, I.I., 1967. Iterative solution of problems of linear and quadratic programming, *Soviet Mathematics Doklady*, 8, 674-675.

Güler, O, D. den Hertog, C. Roos, T. Terlaky, and T. Tsuchiya, 1991. Degeneracy in interior point methods for linear programming, TUDelft Report 91-102, Delft University of Technology, Netherlands.

Lustig, I.J., R.E. Marsten, D.F. Shanno, 1992. On implementing Mehrotra's Predictor-Corrector interior-point method for Linear Programming, *SIAM J. Opt., 2(3)*, 435-449.

Megiddo N., 1988. On finding primal- and dual-optimal bases, Research Report, RJ 6328 (61997), IBM Yorktown Heights, New York.

Mehrotra, S., 1990. On finding a vertex solution using interior point methods, SIAM Annual Meeting, July 16-20, Chicago.

Moler, C., J. Little, and S. Bangert, 1987. *PRO-MATLAB User's Guide*, The MathWorks, Inc., Sherborn, MA, U.S.A.

Ponnambalam, K., 1992. A multi-dimensional search algorithm for optimization using interior point methods, Technical Report, Department of Systems Design Engineering, University of Waterloo.

Ponnambalam, K., and A. Vannelli, 1989. An inexpensive basis recovery procedure for Karmarkar's dual affine algorithm, Fifth Int'l Conf. on Stochastic Program., Univ. of Michigan, Ann Arbor, Aug 13-18.

Ponnambalam, K., A. Vannelli and S. Woo, 1992. An interior point implementation for solving large planning problems in the oil refinery industry, *Can. J. of Chem. Engrg*, 70(2), 368-374.

Singh, A, 1992. A Trust Region Method for Large-Scale Engineering Optimization Using Barrier/Potential Function Models, M.A.Sc. Thesis, University of Waterloo.

Tuncel, L., 1992. Personal Communication.

Solution of and Bounding in a Linearly Constrained Optimization Problem with Convex, Polyhedral Objective Function*

by

András Prékopa

RUTCOR-Rutgers Center for Operations Research
Rutgers University, New Brunswick, New Jersey 08903
Department of Operations Research, L. Eötvös University
Múzeum krt. 6-8, Budapest, Hungary, 1088

and

Wenzhong Li

RUTCOR-Rutgers Center for Operations Research
Rutgers University, New Brunswick, New Jersey 08903

Extended abstract

In this paper we consider the following linearly constrained optimization problem:

$$\min\{c^T x + \sum_{i=1}^{N} Q_i(x)\}$$

$$\text{subject to} \qquad (1)$$

$$Ax = b$$
$$x \geq 0,$$

*American Mathematical Society 1980 Subject Classification, Primary: 90C15. Secondary: 90C25, 90C30.

where A is an $m \times n$ matrix, x, c, b are vectors of suitable sizes, compatible with the above formulation and $Q_i(x)$, $i = 1, 2, \cdots, N$ are some special polyhedral (in other words: piecewise linear) functions. We assume that

$$Q_i(x) = f_i(\xi^i - T^i x), \ i = 1, 2, \cdots, N,$$

where ξ^i is an r^i-component vector, T^i an $r^i \times n$ matrix and f_i a simplicial function on a bounded convex polyhedron $K^i \subset R^{r^i}$, $i = 1, 2, \cdots, N$.

A function $f(z)$, $z \in R^r$ is said to be polyhedral (simplicial) on the bounded convex polyhedron $K \subset R^r$ if there exists a subdivision of K into r-dimensional convex polyhedra (simplices), with pairwise disjoint interiors, such that f is continuous on K and linear on each subdividing polyhedron (simplex).

A simplicial function is a special polyhedral function. The sum of a finite number of simplicial functions, defined on the same convex polyhedron K, is a polyhedral function.

Let $S_1^i, S_2^i, \cdots, S_{k_i}^i$ designate the subdividing simplices and $z_{i1}, z_{i2}, \cdots, z_{ik_i}$ the set of their vertices in case of the function f_i and the corresponding convex polyhedron K^i. Furthermore, let $f_{ij} = f_i(z_{ij})$, $j = 1, 2, \cdots, k_i$; $i = 1, 2, \cdots, N$. Then problem (1) can be reformulated in the following manner:

$$\min\{c^T x + \sum_{i=1}^{N} \sum_{j=1}^{k_i} f_{ij} \lambda_{ij}\}$$

subject to (2)

$$\begin{aligned} Ax &= b, \\ T^i x + \sum_{j=1}^{k_i} z_{ij} \lambda_{ij} &= \xi^i, \\ \sum_{j=1}^{k_i} \lambda_{ij} &= 1, \quad i = 1, 2, \cdots, N, \\ x \geq 0, \quad \lambda &\geq 0. \end{aligned}$$

Many optimization problems can be cast into the form (2). Below we list some of them.

(i) ***Minimization of a convex, separable objective function with piecewise linear objective function, subject to linear constraints.***

(ii) *The discrete simple recourse stochastic programming problem* is shown to be a special case of problem (2), in Prékopa (1990).

(iii) *The linearly constrained minimum absolute deviation problem* is also shown by Prékopa (1990), as a special case of problem (2).

(iv) *Problems with loose constraints.* Assume that A, b, c, x are as in problem (1) and consider the problem:

$$\min c^T x$$

subject to

$$\begin{aligned} Ax &= b \\ T^1 x &= \xi^1 \\ T^2 x &= \xi^2 \\ &\vdots \\ T^N x &= \xi^N, \end{aligned}$$

where $\xi^1, \xi^2, \cdots, \xi^N$ are $r^1, r^2, \cdots, r^N$ component vectors, and $T^1, T^2, \cdots, T^N$ are $r^1 \times n, r^2 \times n, \cdots, r^N \times n$ matrices, respectively. The constraints $T^i x = \xi^i$, $i = 1, 2, \cdots, N$ are allowed not to be satisfied at the expense of some penalty. We say that these constraints are "loose". If the penalty on the ith constraint is given by $f_i(\xi^i - T^i x)$, where f_i is a piecewise linear function on a bounded convex polyhedron K_i that has the property mentioned in the beginning of this section and $f_i(0) = 0$, then we come to the problem:

$$\min\{c^T x + \sum_{i=1}^{N} f_i(\xi^i - T^i x)\}$$

subject to (3)

$$\begin{aligned} Ax &= b \\ x &\geq 0. \end{aligned}$$

Using the λ-representation for the functions $f_1, f_2, \cdots, f_N$, problem (3) takes the form of problem (2).

(v) *Stochastic programming with recourse in case of a discrete random vector.*

(vi) *Programming under probabilistic constraint.*

The purpose of the paper is to present a dual type method for the solution of the problem, together with a fast bounding technique. The method has been implemented in FORTRAN 77 and numerical results have been obtained on Sun SPARC work stations. The results, especially those concerning the bounding technique, are very promising.

References

[1] Dantzig, G.B., and A. Madansky (1961). On the solution of two-stage linear programs under uncertainty. *Proceedings of the Fourth Berkeley Symposium on Statistics and Probability 1*, University of California Press, Berkeley, CA, 165–176.

[2] LEMKE, C. (1954). The dual method for solving the linear programming problem. *Naval Research Logistic Quarterly 1*, 36–47.

[3] LUSTIG, I., R. MARSTEN, and D. SHANNO (1991). Computational experience with a primal-dual interior point method for linear programming. *Linear Algebra and its Applications* 152, 191–222.

[4] PRÉKOPA, A. (1990). Dual method for a one-stage stochastic programming problem with random RHS obeying a discrete probability distribution. *ZOR - Methods and Models of Operations Research* 34, 441–461.

[5] PRÉKOPA, A. and W. LI (1992). On an optimization problem concerning the stochastic PERT problem, *RUTCOR Research Report 18-92.*

[6] PRÉKOPA, A. and J. LONG (1992). New bounds and approximations for the probability distribution of the length of the critical path, *RUTCOR Research Report 16-92.*

A Parallel Implementation of a Framework for Sensitivity Analysis in Multi-Criteria Decision Making

L.G.Proll[+], A.Salhi[+] and D.Rios Insua[*]

[+] Division of Operational Research and Information Systems,
School of Computer Studies, University of Leeds, Leeds LS2 9JT, UK

[*] Dpt. de Inteligencia Artificial, Universidad Politecnica de Madrid
28660-Madrid, Espana

1 Introduction

Multi-attribute decision models rely on two forms of numeric input: *objective data* defining the physical aspects of the alternatives, states and consequences of the decision model; *judgemental data* relating to the decision maker's (DM) beliefs and preferences. Although variations on both forms of data may have great effects on the outcome of the decision model, the concern here is with variations in the judgemental data.

In its traditional sense, sensitivity analysis allows exploration of the effects of variations in the judgemental input on the ranking of alternatives. However, as pointed out by French [5] and others, it also plays the role of an aid to the elicitation process of decision analysis by focussing discussion and reflection on the judgemental data. Commercially available decision analysis packages such as ARBORIST [12], HIVIEW [1] and VISA [3], however, allow only limited forms of sensitivity analysis.

Recently Rios Insua and French [9, 11] have developed a conceptual framework for sensitivity analysis in multi-criteria decision making which allows simultaneous variation of judgemental data and which applies to any paradigm for decision analysis. The principal problem in implementing the Rios Insua-French framework is the heavy computational load which is required to support

it. This arises because the analysis relies on distance-based tools and involves the solution of many mathematical programming problems, which may be nonlinear and nonconvex. Even though the individual optimisation problems are small, this represents a substantial load, particularly since, in order to provide reliable information from the sensitivity analysis, global optima of the nonconvex problems must be sought. The computational load may inhibit use of the framework particularly as decision analyses are often performed in the context of decision conferences, where it is desirable for sensitivity analyses to be conducted in near real time preferably on a PC. Approaches to reducing the time to perform the analysis include reformulation of some of the mathematical programs involved [7, 10] and exploiting parallelism within the phases [6].

In the following we shall briefly describe the framework and its implementation in a parallel environment consisting of a PC enhanced by the addition of a transputer board. This will allow the necessary analyses to be undertaken in parallel, thus making it possible to handle the computational load in acceptable times. Experimental results with linear and bilinear models will be included.

2 The Sensitivity Analysis Algorithm

2.1 The Evaluation Problem

Consider the evaluation problem in multi-criteria decision making [2] in which a finite set of alternatives $a_i i = 1, ..., N$ are ranked using the evaluation function $\Psi(a_i, w)$ where w is the vector of judgemental data.Then alternative a_i is preferred to alternative a_j if

$$\Psi(a_i, w) \geq \Psi(a_j, w)$$

To detect a^* the alternative which currently ranks first, each judgemental input is provided with an initial value w^0. Other information related to the judgemental data such as monotonicity conditions on utilities, normalisation conditions on probabilities or weights are represented by constraints limiting

the values of w acceptable to the decision maker (DM) to the set S so that $w \in S$. Three cases occur in practice:

- linear models, in which Ψ is linear and S is defined by linear constraints,
- bilinear models, in which Ψ is bilinear and S is defined by linear constraints,
- general models, in which Ψ is nonlinear and S not necessarily convex.

2.2 Algorithm

The general algorithm for sensitivity analysis as described in [9] proceeds through four phases:

1. The *dominance* phase in which the set A_1 of non-dominated alternatives is found;
2. The *potential optimality* phase in which A_2 the set of non-dominated alternatives which could be optimal for some $w \in S$ is found;
3. The *adjacent potential optimality* phase in which A_3 the set of potentially optimal alternatives which are contenders to a^* for optimality if a smooth change of w away from w^0 occurs, is found;
4. The *distance analysis* phase in which the minimum change in w^0 required by each contender a_j before optimality switches from a^* to a_j is computed in both the l_1 and l_∞ metrics. This information, together with that provided by maximum distance problems, is then used to compute an index of sensitivity of the optimum solution [6].

2.3 Computational Aspects

Each phase requires the solution of several mathematical programmes, whose form depends on the phase and the nature of $\Psi(a_i, w)$ and of S. In the linear model, all problems arising in Phases 1-3 are linear programmes. The minimum distance problems are convex and can be transformed to linear programmes. The maximum distance problems are non-convex but the l_∞ problem can be solved by linear programming [7]. Hence only the l_1 maximum distance problem causes difficulty.

In the bilinear model, the l_∞ problem again can be solved as a set of linear programmes but all other problems have a bilinear objective function and/or bilinear constraints and, in general, are non-convex. In the general model all problems are general nonlinear programmes and, in general, are non-convex.

3 Parallel Implementation

The general algorithm for sensitivity analysis is intrinsically sequential due to the precedence relationships between the phases. Parallelism, however, is present at the level of each phase. Given the potentially large number of mathematical programmes to be solved, large grain parallelisation is most appropriate especially when the hardware to be used is a MIMD machine, such as a transputer network, because of the high ratio of time spent in communication. Also, transputers being fast processors with enough local memory, it is attractive to make them operate as autonomously as possible.

In the present implementation, the *processor farm* approach is adopted. Individual problems arising at the level of each phase are farmed out to individual nodes by the *master* process which resides on the root node. The *worker* process is replicated on each node of the network.

The *master* process consists of three threads running concurrently: the *producer* which generates the problems; the *sender* thread which transmits them to the network; the *consumer* thread which collects the results from the network and displays them.

The *worker* process consists of a single thread which communicates with the master and solves linear and nonlinear programmes.

3.1 Equipment

The parallel processing environment consists of an Elonex 386S-200 PC comprising a TMB04 motherboard with a T805 processor and 4 Mbyte RAM as the root node; 4 T805 transputers with 4 Mbyte of RAM each and 3L Parallel Software consisting of a Fortran compiler and a set of extension functions for message passing, a task harness and a configuration system. The topology of the network is a ring with a connection between the free link of the root transputer and one of the two nodes which are not linked to it. The network

is configuring by file submitted to the static configurer or the flood-fill configurer. Synchronisation is either implicit through algorithm design or explicit using descheduling functions.

In order to use this system, codes for linear and general nonlinear programming are required. For linear programming, a simplex-based routine has been developed. For the general nonlinear programming problem, the multi-level single linkage algorithm of Rinnoy-Kan and Timmer [8] was adopted. It relies on a local optimisation routine based on a sequential quadratic programming algorithm [4].

3.2 Parallel Algorithm

The algorithm of the master process consists of three procedures which run concurrently.

Producer thread:

Read data of the considered model of decision analysis;
Broadcast initial data to all workers;
Start *Sender*;
Start *Consumer*;
Rank alternatives according to $\Psi(a_i, w)$ and w^0;
Start sensitivity analysis;
For Phase = 1:4 do
a:Generate problem in Phase;
Put under standard message form;
Wait for *Sender* to be ready;
Pass message to *Sender*;
go to a:;
Endfor;
When all phases have been considered send Stop-message to
Sender and Workers;
Send Stop-message to *Consumer* through a COMMON variable;

Sender thread:

b:Wait for an incoming message from *Producer*;
Receive message;
When the network presents an idle processor, pass message to it;
If message is a Stop-message then stop;
Otherwise go to b:

Consumer thread:

c:Wait for an incoming message from the network;
Receive message;
Process it and display results;
Check for Stop if current phase is the last;
Or else go to c:

4 Conclusion

Computational results show that the framework is viable for linear models of practical size. For the bilinear models, limited experience shows performance gains due to parallelisation. Two strategies for solving the global optimisation problems in parallel were tried; the first strategy kept the algorithm for global optimisation as part of the master process and thus only local optimisation problems were farmed out. In the second the global optimisation problems themselves were sent to the workers, thus making the global optimisation routine part of the worker process. Although, better results were obtained with the second strategy, in both cases the speed up was less than linear.

References

[1] S. Barclay. *A User's Manual to HIVIEW.* Decision Analysis Unit, London School of Economics, UK, 1987.

[2] V. Belton. Multiple criteria decision analysis - practically the only way to choose. In L.C. Hendry and R.W. Eglese, editors, *Operational Research Tutorial Papers*, pages 53–101. Operational Research Society, Birmingham, 1990.

[3] V. Belton and S. Vickers. VISA - VIM for MCDA. In A.G. Lockett and G. Islei, editors, *Improving Decision Making in Organisations*. Springer Verlag, Berlin, 1989.

[4] M.C. Biggs. Constrained minimization using recursive equality quadratic programming. In F.A. Lootsma, editor, *Numerical Methods for Nonlinear Optimization*, pages 411–428. Academic Press, 1972.

[5] S. French. Recent mathematical developments in decision analysis. *IMA Journal of Mathematics Applied to Business and Finance*, 3:1–12, 1991.

[6] L.G. Proll, D. Rios Insua, and A. Salhi. Mathematical programming and the sensitivity of multi-criteria decisions. *Annals of Operations Research*. to appear.

[7] L.G. Proll and A. Salhi. On two non-convex optimisation problems. Technical Report 92.14, School of Computer Studies, University of Leeds, U.K., 1992.

[8] A.H.G. Rinnooy Kan and G. Timmer. Global optimisation. In G.L. Nemhauser, A.H.G. Rinnooy Kan, and M.J. Todd, editors, *Optimization*, chapter IX, pages 631–662. North Holland, Amsterdam, 1989. Handbooks in Operations Research and Management Science Volume 1.

[9] D. Rios Insua. *Sensitivity Analysis in Multi-objective Decision Making*. Springer Verlag, Berlin, 1990. Lecture Notes in Economics and Mathematical Systems no.347.

[10] D. Rios Insua. On adjacent potential optimality. Technical report, Dpt. de Inteligencia Artificial, Universidad Politecnicsde Madrid, 1992. Working Paper.

[11] D. Rios Insua and S. French. A framework for sensitivity analysis in discrete multi-objective decision-making. *European Journal of Operations Research*, 54:176–190, 1991.

[12] D. Samson. *Managerial Decision Analysis.* Irwin, Homewood, Illinois, USA, 1988.

A Comparison of Two Stochastic Algorithms for Global Optimisation

L.G.Proll[+], A.Salhi[+], D.Rios Insua[*] and J.I.M.Jimenez[*]

[+] Division of Operational Research and Information Systems,
School of Computer Studies, University of Leeds, Leeds LS2 9JT, UK

[*] Dpt. de Inteligencia Artificial, Universidad Politecnica de Madrid,
28660-Madrid, Espana

1 Introduction

The *Global Optimisation (Minimisation) Problem* can be expressed as follows: Let f be a function from R^n to R and $A \subset R^n$, then find $x^* \in A$ such that $\forall x \in A$, $f(x^*) \leq f(x)$.

When f is unimodal, a suitable algorithm may be found among the numerous algorithms for *local optimisation*, depending on properties such as continuity and differentiability of f, compactness of A etc. When f is multimodal and/or lacking the above properties, there is no method which is guaranteed to find its global minimum. A difficulty is that no practically useful test for optimality is available, as is the case for smooth local optimisation, so that, in principle, one has to evaluate the function at every point of A, which is not possible. As Schoen [11] says the problem is "inherently intractable".

Despite the difficulty of the problem, we are witnessing an 'explosion' in the design of algorithms for GO. As a consequence it is not easy for the user to choose a suitable algorithm for his/her application, particularly if the algorithm is to be embedded in a general purpose package. Despite many reviews of GO techniques, the experimental record is patchy but points to the relative success of stochastic algorithms. However, there are many algorithms in this class.

In the following we shall be concerned with representatives of two popular stochastic algorithms for global optimisation, namely clustering and simulated

annealing. Issues related to their implementation and use to solve practical problems involving general constraints in addition to simple bounds on the variables will be discussed. Computational results will be given.

2 Stochastic Algorithms

Stochastic algorithms generally employ both heuristic and probabilistic methods to provide an approximation to the global optimum. The major components of such algorithms are:

- sampling - generate a random sample of points in A
- local optimisation - apply a local optimisation algorithm to f starting from each of a subset of the sampled points
- stopping rules - decide whether to draw another sample or whether to accept the smallest observed value of f in A as an approximation to the global minimum.

The quality of the approximation usually can be traded-off against the runtime of the algorithm by adjusting one or more parameters which control the above components. However there is little or no systematic knowledge on which to base such tradeoffs. In the case of constrained global optimisation, A is often relaxed to a hypercube in order to ease the task of sample generation. We then rely on the local optimiser to find feasible points.

2.1 Clustering

Cluster analysis is usually associated with pattern recognition where objects belonging to an initial set are separated into subsets or *clusters* according to some similarity characteristic, say shape or colour etc... In GO it was introduced by Becker and Lego [2] in the early 70's in order to reduce computing time in the multistart algorithm [15]. The basic idea is to try to identify clusters of points which will lead to the same local minimum. By starting a local optimisation from a representative of each cluster each distinct local minimum hopefully will be discovered only once. The practical implementation of this idea, however, depends on the choice of many parameters which make up

the *Threshold Distance* on which clustering depends. Negotiating the difficult choice of these parameters and adequate stopping rules remains the key to the efficient implementation of clustering algorithms. Recent attempts have been made to improve clustering algorithms by adequate choice of sample size [12] and use of topographical information of the search space [13]. However, no experimental results yet provide strong support for these improvements.

The multi-level single linkage algorithm of Rinnooy Kan and Timmer [10] appears to be one of the most efficient amongst stochastic algorithms for GO. Tests, however, appear to be confined to problems of small dimension and with only simple bound constraints. An implementation of this algorithm will be considered here. A somewhat simpler clustering method due to Torn and Viitanen [14] is also considered since it has potential advantage for the application which motivated this research.

2.2 Simulated Annealing

The concepts of annealing in optimisation were introduced in the early 80's [9], [5]. Initially interest was in their application to solving combinatorial optimisation problems [1]. The simulated annealing based algorithm (SA) has been reported [9], [8] to perform well on such problems in high dimensions with a large number of local minima. Based on this success, variants of SA for continuous global optimisation have been developed [16], [4], [6], [7]. SA is a stochastic method by means of which the global minimum of a function f, regardless of its continuity or differentiability, can be approached as close as one desires. The main feature of the algorithm is its ability to 'distinguish' between the fine behaviour and the gross behaviour of the objective function. Assuming that fine behaviour leads to poor local minima with small regions of attraction, the algorithm detects it and avoids getting trapped by taking uphill steps, thus allowing the function value to increase momentarily. This strategy is based on what is called the acceptance/rejection rule. To draw a parallel, note that the strategy of the multistart algorithm relies on starting a local search from different points.

Accepting or rejecting uphill steps in SA is determined by a sequence of random points with a controlled probability $min(1, e^{\frac{-(f(y)-f(x))}{T}})$, [7], where T is the control parameter. This parameter is crucial as it slows down the algorithm

if it is too high and it removes the global aspect of the algorithm, i.e. uphill moves, if it is too small [1], [8], [11]. Besides parameter T which requires an initial value, a decrement function for decreasing it and a final value to use in the stopping condition, another parameter is also required to be set for any practical implementation of the algorithm. This parameter is the length L of each Markov chain corresponding to each sequence of decreasing T. The concept of finite Markov chains is used to derive a mathematical model of SA, given that in SA the outcome of a trial depends only on the outcome of the previous trial [1]. This set of parameters is usually referred to as a *cooling schedule* [7], [1].

The simulated annealing algorithm implemented in the present work was based on the Dekker and Aarts variant [7]. The principal difference is in the generation of new points from a given point. While in [7] this was accomplished using a local search procedure, we adopt a Hit-and-Run algorithm for detecting non-redundant constraints [3]. This is mainly because the search spaces of the problems to be solved are not defined just by simple bound constraints.

3 Experimental Work

The test problems used in the experiments are practical problems arising in decision analysis. They are by no means extremely difficult, but they present nondifferentiability, nonlinearity in the objective function or the constraints or both. Above all, they have many minima and nontrivial constraints and are of higher dimension than is often encountered in the literature. Comparative results obtained with Fortran77 codes for the three algorithms cited above will be reported.

References

[1] E. Aarts and Korst J. *Simulated Annealing and Boltzmann Machines.* Wiley-Interscience Series in Discrete Mathematics and Optimization. John Wiley and Sons, 1989.

[2] R.W. Becker and G.V. Lego. A global optimization algorithm. In *Proceedings of the 8th Allerton Conference on Circuits and Systems Theory*, pages 3–12, Monticello, Illinois, 1970.

[3] H.C.P Berbee, C.G.E. Boender, A.H.G. Rinnoy Kan, C.L. Scheffer, R.L. Smith, and J. Telgen. Hit-and-run algorithms for the identification of nonredundant linear inequalities. *Mathematical Programming*, 37:184–207, 1987.

[4] I.O. Bohachevsky, M.E. Johnson, and M.L. Stein. Generalized simulated annealing for function optimization. *Technometrics*, 28:209–217, 1986.

[5] V. Cerny. Thermodynamical approach to the traveling salesman problem: an efficient simulation algorithm. *Journal of Optimization Theory and Application*, 45:41–51, 1985.

[6] A. Corana, M. Marchesi, C. Martini, and S. Ridella. Minimizing multimodal functions of continuous variables with the simulated annealing algorithm. *ACM Transactions on Mathematical Software*, 13(3):262–280, 1987.

[7] A. Dekker and E. Aarts. Global optimization and simulated annealing. *Mathematical Programming*, 50:367–393, 1991.

[8] R.W. Eglese. Simulated annealing: A tool for operational research. *European Journal of Operational Research*, 46:271–281, 1990.

[9] S. Kirkpatrick, C.D. Gelatt Jr., and M.P. Vecchi. Optimization by simulated annealing. Technical Report RC 9355, IBM, 1982.

[10] A.H.G. Rinnooy Kan and G. Timmer. Global optimisation. In G.L. Nemhauser, A.H.G. Rinnooy Kan, and M.J. Todd, editors, *Optimization*, chapter IX, pages 631–662. North Holland, Amsterdam, 1989. Handbooks in Operations Research and Management Science Volume 1.

[11] F. Schoen. Stochastic techniques for global optimization: A survey of recent advances. *Journal of Global Optimization*, 1:207–228, 1991.

[12] F. Schoen. On a new stochastic global optimization algorithm based on censored observations. Working Paper, July 1992.

[13] A. Torn and S. Viitanen. Topographical global optimization. In C.A. Floudas and P.M. Pardalos, editors, *Recent Advances in Global Optimization*, pages 384–398. Princeton University Press, 1992.

[14] A. Torn and S. Viitanen. Topographical global optimization using presampled points. Working Paper, September 1992.

[15] A.A. Torn and A. Zilinkas. *Global Optimization.* Springer-Verlag, 1989.

[16] D. Vanderbilt and S.G. Louie. A monte carlo simulated annealing approach to optimization over continuous variables. *Journal of Computational Physics*, 56:259–271, 1984.

A General Class of Greedily Solvable Linear Programs

Maurice Queyranne
University of British Columbia, Vancouver, Canada

Frits Spieksma *
University of Limburg, Department of Mathematics, Maastricht, The Netherlands

and Fabio Tardella
Istituto d'Elaborazione dell' Informazione, Pisa, Italy

September 1992

1 Introduction.

An important class of simple and efficient algorithms for optimizing a function f on a set S is the class of *greedy* (or myopic) algorithms. Since the work of Edmonds [14], [15] on matroids and of Hoffman [23] on transportation problems, numerous authors have studied conditions on f and S which guarantee the convergence of greedy algorithms to optimal solutions. In the case where f is linear and S is a polyhedron, two broad and well-known classes of linear programs have been shown to be optimally solvable by a greedy algorithm. Following the work of Edmonds, the first class includes optimization problems on polymatroids and related submodular polyhedra, see Frank and Tardos [17] and Fujishige's monograph [19] for in-depth studies. On the other hand, following the work of Hoffman, the second class includes transportation problems, both ordinary and multi-index, with cost coefficients satisfying some form of a so-called *Monge condition* (see Hoffman [25] and Bein et al. [11] for details).

In this paper, we present a dual pair of linear programs, in which the variables are associated with the elements in a sublattice of a discrete product lattice. We show that a greedy algorithm solves both primal and dual programs when the cost coefficients in the primal problem (or, equivalently, the right hand sides in the dual problem) are given by a submodular function on the sublattice. The primal problem generalizes the multi-index transportation problem of Bein et al. [11] to the case of forbidden arcs with a sublattice structure. The dual problem generalizes the linear optimization problems on submodular polyhedra by Lovász [29] and Fujishige and Tomizawa [20] to a distributive sublattice of a finite product space.

In addition to enlarging the class of linear programs solvable by a greedy algorithm, our work also unifies heretofore separate streams of research and highlights the duality relationship between (multi-index) transportation problems and linear optimization problems on submodular polyhedra. In particular, we observe that submodularity and the Monge condition are the same concept expressed in different forms. Indeed, known results on lattices and submodular functions

*University of Limburg, Department of Mathematics, P.O. Box 616, 6200 MD Maastricht, The Netherlands

are independently re-discovered in the context of the Monge condition for multi-dimensional arrays. Conversely, new results for Monge matrices also apply to submodular functions.

The contents of this paper are as follows. In Section 2, we define the dual pair of linear programs which is the object of this paper. We show how they generalize multi-index transportation problems and linear optimization problems on submodular polyhedra. In Section 3, we present the greedy algorithm and prove that it produces optimal solutions to the primal and the dual problems. Finally, in Section 4, we discuss relations between submodularity, the Monge condition for a matrix and the existence of Monge sequences.

2 Lattices, submodular functions, and a dual pair of linear programs.

Let the integer $k \geq 2$ denote the dimension of the product lattice defined below, and $K \doteq \{1,\ldots,k\}$. For $i \in K$, let A_i be a totally ordered set (or *chain*) with $m(i)+1$ elements. For simplicity, we let $A_i \doteq \{0,1,\ldots,m(i)\}$, with the usual total order $0 < 1 < \cdots < m(i)$, for all $i \in K$. The *product space* $A \doteq A_1 \times A_2 \times \cdots \times A_k$ is a distributive lattice with *join* and *meet* operations defined by

$$a \vee b \doteq \Big(\max\{a(1),b(1)\},\ \ldots\ ,\max\{a(k),b(k)\}\Big)$$

and

$$a \wedge b \doteq \Big(\min\{a(1),b(1)\},\ \ldots\ ,\min\{a(k),b(k)\}\Big),$$

respectively, where a and b are any elements of A. As is well-known in lattice theory (see references below), these operations induce the usual *partial order* "$\leq$" on this lattice A by

$$a \leq b \iff a = a \wedge b \quad (\iff b = a \vee b).$$

Let $0 \doteq (0,\ldots,0)$ and $m \doteq (m(1),\ldots,m(k))$ denote the smallest and largest element of A, respectively.

Let B denote any subset of A. For any $i \in K$ and $j \in A_i$ we define the *section* $B(i,j)$ of B at (i,j) as $B(i,j) \doteq \{a \in B\ :\ a(i) = j\}$. For any $a \in A$ we define the *(lower) truncation* B_a of B at a as $B_a \doteq \{b \in B\ :\ b \leq a\}$.

A subset B of A is a *sublattice* if for every $a,b \in B$ we have $a \vee b \in B$ and $a \wedge b \in B$. If B is a sublattice then the sections $B(i,j)$ and the truncations B_a are also sublattices, for all $i \in K$, $j \in A_i$ and $a \in A$.

A real-valued function $f : B \mapsto \mathcal{R}$ on a sublattice B is *submodular* if the following *submodular inequality*

$$f(a \vee b) + f(a \wedge b) \leq f(a) + f(b)$$

holds for all $a,b \in B$. It is *strictly submodular* if this inequality is strict whenever $a \vee b \notin \{a,b\}$. See, for example, Birkhoff [12] for a general exposition of lattice theory, and Topkis [37], Veinott [38] and Granot and Veinott [21], and the references therein, about product lattices, their sublattices, and submodular functions (called "subadditive" functions in the latter reference).

Let B be any subset of A and let $B^\bullet \doteq B \setminus \{0\}$. We associate a *cost* $w(a)$ with every element $a \in B^\bullet$, and a nonnegative *demand* d_{ij} with every section $B(i,j)$ where $i \in K$ and $j \in A_i^\bullet \doteq A_i \setminus \{0\}$.

We now formulate a dual pair of linear programs (P) and (D):

$$(P) \quad \min \quad \sum_{a \in B^*} w(a) x_a$$

$$\text{s.t.} \quad \sum_{a \in B(i,j)} x_a = d_{ij} \qquad \text{for } i \in K \text{ and } j \in A_i^*;$$

$$x_a \geq 0 \qquad \text{for all } a \in B^*;$$

and

$$(D) \quad \max \quad \sum_{i \in K} \sum_{j \in A_i^*} d_{ij} y_{ij}$$

$$\text{s.t.} \quad \sum_{\substack{i \in K \\ a(i) \neq 0}} y_{i,a(i)} \leq w(a) \qquad \text{for all } a \in B^*.$$

Multi-index transportation problems. The primal problem (P) just defined contains as a special case the following *axial k-index transportation problem with forbidden arcs*. The k sets $A_1^*, \ldots, A_k^*$ may be interpreted as sets of sources, destinations, types of goods, and various related resources. Let B denote the subset of $A' \doteq A_1^* \times \ldots \times A_k^*$ consisting of non-forbidden (permissible) combinations $a \in A'$. With each section $B(i,j)$ we associate a nonnegative "demand" (which may be interpreted as a supply when A_i^* is a set of sources, and as a capacity when A_i^* is a set of resources). It is assumed that $\sum_{j \in A_i^*} d_{ij} = D$, a constant for all $i \in K$. With each element $a \in B$, often called an "arc", we associate a cost rate $w(a)$ and a nonnegative decision variable x_a representing the amount of "demand" which is satisfied by the corresponding combination. The (axial) k-index transportation problem is to determine the amount associated with each permissible arc $a \in B$ so as to satisfy exactly the demand of each section $B(i,j)$ (for all $i \in K$ and $j \in A_i^*$) at minimum total cost. This problem may be formulated precisely as an instance of problem (P).

The case where $B = A'$ (no forbidden arcs) is the axial multi-index transportation problem defined by Haley [22] (see also Chapter 8 in [41], and [11]). The *axial k-index assignment problem* arises when all $m(i)$ are equal to a constant m, all demands are equal to 1, and all decision variables are restricted to be either 0 or 1, see, e.g., Pierskalla [32] and Bandelt et al. [9]. When, in addition, $k = 3$, we have the much studied (axial) *three-dimensional assignment problem*, see Frieze and Yadegar [18], Balas and Saltzman [8], and Crama and Spieksma [13]. The above references describe several practical applications of these different models in such areas as logistics, automated production, statistics, and course scheduling.

Note that, in addition to offering a compact notation (compared to the above references), our formulation of problem (P) allows us quite naturally to exclude forbidden arcs. Ordinary (i.e., 2-index) transportation problems with forbidden arcs were considered by Shamir and Dietrich [36] in connection with the existence of Monge sequences; see Section 4 for details. Note also that we do not need to require in problem (P) that the total demand $\sum_{j \in A_i^*} d_{ij}$ be constant for all $i \in K$.

Submodular polyhedra. When $m(i) = 1$ for all $i \in K$, the lattice A may be identified with the lattice 2^K of all subsets of K. Sublattices B are then (distributive) set lattices, and submodular functions coincide with those now well-known in combinatorial optimization (see, e.g., Nemhauser and Wolsey [30]). The constraints of problem (D) are then precisely those which define a *submodular polyhedron* as in Fujishige [19] (see also Frank and Tardos [17]).

When $B = A = 2^K$ we have the problem which Lovász [29] shows to be solvable by a greedy algorithm. These polyhedra are closely related to the *polymatroids* introduced by Edmonds [15], see the preceding references for details.

Problem (D) properly generalizes these submodular polyhedra by allowing each chain A_i in the lattice to contain any number of elements, giving rise to arbitrary (finite) product lattices. This is akin to extending attention, in integer programming, from binary variables to general integer-valued variables. We refer to the work of Topkis, Veinott, and Granot and Veinott cited above for a description of some of the problems amenable to this broader framework.

The Greedy Algorithm in the next section generalizes on one hand those of Hoffman and Bein et al. for (ordinary and multi-index) transportation problems, and on the other hand those in Lovász and in Fujishige and Tomizawa for submodular polyhedra (the latter two being themselves generalizations of that of Edmonds for polymatroids).

3 A greedy algorithm.

We first describe the input and output of our algorithm for problems (P) and (D). Throughout this Section, we assume that B is any sublattice of A.

Input:

integer k	the dimension of A;
integers $m(1), \ldots, m(k)$	defining the range of each coordinate of A;
reals $d_{ij} \geq 0$	demands, for all $i \in K$, $j \in A_i^*$;
oracle MAXLE	describing sublattice B (see explanations below);
oracle w	returning the value $w(b)$ for any $b \in B$.

Output:

variable *Status*	indicating the status, *Feasible* or *Infeasible*, of problem (P);

and if *Status* = *Feasible*:

list $((b^1, x_{b^1}), \ldots, (b^T, x_{b^T}))$	describing a primal solution (see below);
reals $y_{ij} \geq 0$	describing a dual solution, for all $i \in K$ and $j \in A_i^*$.

The sublattice B might be presented in different ways, such as: a list of all elements in B (the *permissible* elements or cells); a list of all elements in $A \setminus B$ (the *forbidden* elements); collections of conditions (for example, *monotone linear inequalities*, see [38]) characterizing permissible or forbidden elements; and so forth. However, to achieve sufficient generality and to exploit the intrinsic simplicity of the Greedy Algorithm, we use the following oracle, which we call MAXLE. The input to MAXLE consists of any element $a \in A$. Oracle MAXLE then returns "NULL" if the truncation B_a of B generated by a is empty; else it returns $\bigvee B_a$, the largest element of $b \in B$ such that $b \leq a$. We leave it to the interested reader which data structures can be used to efficiently implement this oracle for a given representation of the sublattice B, such as one of those outlined above.

The output to the Greedy Algorithm exploits the sparseness of the basic solutions to problem (P). Although the primal solution vector x has one component x_b for every element $b \in B^*$ (and hence potentially up to $(\prod_i (m(i)+1)) - 1$ variables), at most $\sum_i m(i)$ of these will assume

a positive value. Thus the Greedy Algorithm, which will be shown to produce a dual pair of basic solutions to problems (P) and (D), returns a list of T pairs (b, x_b) with $b \in B^*$ and x_b is the value of the corresponding variable in the solution. The number T of such pairs is determined by the algorithm, but will be shown not to exceed $\sum_i m(i)$. It is understood that $x_b \doteq 0$ for all $b \in B^*$ which do not appear in this list.

The Greedy Algorithm detailed below consists of two phases. The Primal Phase repeats the following step: identify (using the MAXLE oracle) the largest available element $b \in B^*$ and assign the largest possible value to its variable x_b. This step is repeated until either infeasibility is detected, in which case the algorithm halts, or all demands are satisfied. In the latter case, the list of (b, x_b) pairs output by the algorithm defines a feasible primal solution. The Dual Phase then traces back the sequence of elements $b \in B$ recorded in the Primal Phase to construct (using the w oracle) a dual solution y.

GREEDY ALGORITHM:

Primal Phase.

0. (Initialize:)
   Let $\delta_{ij} := d_{ij}$ for all i, j;
   $\Lambda := K$; $a := \text{MAXLE}(m)$; t := 0;
1. (Main step):
   Repeat
     if ($a \neq$ NULL) then {
       if (there exists (i, j) with $j > a(i)$ and $\delta_{ij} > 0$) then
         return(*Status* := *Infeasible*)
       else {
       $t := t + 1$;
       let $i = \text{argmin}\{\delta_{\ell,a(\ell)} : \ell \in \Lambda\}$;
       $x_a := \delta_{i,a(i)}$; $\eta(t) := i$;
       add $(a_t, x_{a_t}) := (a, x_a)$ to the output list;
       let $\delta_{\ell,a(\ell)} := \delta_{\ell,a(\ell)} - x_a$ for all $\ell \in \Lambda$;
       $a(i) := a(i) - 1$; if $a(i) = 0$ then let $\Lambda := \Lambda \setminus \{i\}$;
       let $a := \text{MAXLE}(a)$;
       }
     }
   until ($a = 0$ or $a =$ NULL);
2. (Final test for infeasibility):
   If ($a =$ NULL and there exists (i, j) with $\delta_{ij} > 0$) then
     return(*Status* := *Infeasible*);
   else { let $T := t$; output the list $(a_1, x_{a_1}), \ldots, (a_T, x_{a_T})$; }

Dual Phase.

For $t := T$ down to 1 do
  output $y_{\eta(t),a_t(\eta(t))} :=$
    $w(a_t) - \sum\{w(a_u) : \text{all } u > t \text{ with } a_u(\eta(u)) = a(\eta(u))\}$;
Return(*Status* = *Feasible*).

Note that, when applied to an ordinary (two-dimensional) transportation problem, the Primal Phase reduces to the well-known North-West corner rule (with an appropriate geographic orientation of the transportation array). More generally, the Primal Phase reduces to the greedy

algorithm of Bein et al. [11] for multi-index transportation problems without forbidden arcs. In the case where problem (D) is a linear optimization problem over a submodular polyhedron, (that is, $A_i = \{0,1\}$ for all $i \in K$), the Primal Phase amounts to sorting the d_{i1} values in a nondecreasing order, *consistent* with the sublattice B in the following sense: if $b(i) = 1$ for all $b \in B$ with $b(h) = 1$, and if $\eta(t) = i$ and $\eta(u) = h$, then $t > u$ (for all $h, i \in K$). Then, in the Dual Phase, the y-variables are sequentially maximized according to this sequence, with $y_{\eta(t),a_t(\eta(t))} := w(a_t) - \sum_{u>t} w(a_u)$. Thus the Greedy Algorithm just presented reduces to that of Lovász [29] and of Fujishige and Tomizawa [20] in the case of submodular polyhedra.

Theorem 3.1 *Let B be a sublattice of a finite product space.*

(1) The Greedy Algorithm returns Status = Infeasible *if and only if problem (P) is infeasible.*

(2) If problem (P) is feasible, then the Greedy Algorithm outputs an optimal solution to problem (P) for all nonnegative demands d if and only if w is submodular.

(3) If problem (P) is feasible and w is submodular, then the Greedy Algorithm outputs an optimal solution to problem (D) in the Dual Phase.

Note that, for a given demand vector d such that problem (P) is feasible, the primal solution constructed by the Greedy Algorithm does not depend on the cost function w, provided it is submodular. See [1] and [2] for a study of a similar property in the context of ordinary transportation and minimum cost network flow problems.

A consequence of Theorem 1 is that, when w is submodular, the inequalities of problem (D) form a *totally dual integral* (TDI) linear inequality system (see Edmonds and Giles [16], Hoffman [24], and Nemhauser and Wolsey [30] for a definition and properties of TDI systems).

4 Submodular costs and Monge properties.

The main purpose of this section is to point out and exploit the equivalence between submodularity of a function defined on a product of k chains, and the Monge condition of a k-dimensional array. We also introduce the concept of submodular sequences, and discuss its relationship with that of Monge sequences in two-dimensional arrays. In particular, we show that, for any *strictly* submodular two-dimensional array, the class of submodular sequences coincides with that of Monge sequences.

The concept of Monge sequences was introduced for two-dimensional arrays (matrices) by Hoffman in 1963 [23] in order to describe classes of transportation problems that are greedily solvable. A *Monge sequence* for a two-dimensional $n \times m$ array $C = (c[i,j])$ is a total ordering of the nm pairs (i,j) such that, whenever pair (i,j) precedes both pairs (i,ℓ) and (k,j),

$$c[i,j] + c[k,\ell] \leq c[i,\ell] + c[k,j].$$

The inequality in this definition has been independently observed and exploited in algorithms for a wide variety of problems, under various conditions about the indices i, j, k and ℓ. The most common condition is that $i < k$ and $j < \ell$: a two-dimensional array C satisfies the *Monge condition* if

$$c[i,j] + c[k,\ell] \leq c[i,\ell] + c[k,j]$$

holds for all i, j, k, ℓ with $i \leq k$ and $j \leq \ell$. Note that this definition is precisely that of the submodularity of the function defined by C on the product lattice $\{1,\ldots,n\} \times \{1,\ldots,m\}$.

When the only defined entries in the matrix are above the diagonal (thus forming a sublattice of the lattice A of matrix cells), this condition is also called the (concave) quadrangle inequality

(e.g., [40]), and (as if to add to the confusion) functions defined by such an array are sometimes called concave functions (e.g., [28]). The computer science community has seen a flurry of activity on these and closely related concepts during the past few years. This activity was motivated in part by the seminal paper of Aggarwal et al. [4] on matrix searching, and also by a wide variety of applications to problems in such areas as computational geometry (e.g., [4], [3]), VLSI channel routing (e.g., [4], [5]), signal quantization [39], molecular biology (e.g., [35], [28]), dynamic lot sizing (the Wagner-Whitin problem) [7], and the travelling salesman problem [31]. Because the field is now so vast, we have only given here a few indicative references. Further references can be found therein.

The concept of Monge arrays has recently been extended to k-dimensional arrays by Aggarwal and Park [5], [6]: a k-dimensional array C satisfies the *Monge condition* if every two-dimensional plane of C corresponding to fixed values of $k-2$ coordinates satisfies the Monge condition.

Actually, the Monge condition just defined is equivalent to the submodularity of the function $c : A \mapsto \mathcal{R}$ defined by the array C. Indeed, the following result (compare Proposition 2.4 in [5]) is a direct rephrasing of Theorems 3.1 and 3.2 in Topkis [37] for the case where A is a product of a finite number of chains, as is assumed throughout the present paper:

Theorem 4.1 *A function $c : A \mapsto \mathcal{R}$ is submodular if and only if it is submodular on every two-dimensional sublattice (plane) corresponding to fixed values of $k-2$ coordinates.*

As a consequence, many results on k-dimensional arrays satisfying the Monge condition (such as in Section 2 of [5] and in Section 2 in [11]) can be directly derived from known results on submodular functions. In addition, the deep theory of parametric lattice programming developed in Topkis [37] also applies to problems involving k-dimensional arrays satisfying the Monge condition. Conversely, the rich computer science literature on Monge and related arrays, in particular the fast algorithms developed for a variety of such problems, may also be exploited to study submodular functions on discrete product lattices. (A case in point is the dynamic lot-sizing problem considered in detail in Topkis, and for which fast algorithms were derived in [7] using the Monge condition.) The problem of recognizing the Monge condition, and hence submodularity, is considered by Ruediger [34] for k-dimensional arrays.

We now go back to Monge sequences. This concept was originally introduced by Hoffman for two-dimensional arrays, as seen above. It was further investigated, among others, in [36], [1], [2], and is closely related to the notion of *greedoids* (see [27]).

Some authors (e.g., Bein et al. [10], and Ruediger [33]) have recently investigated the possibility of extending this concept to higher dimensions. However, at present no approach seems to clearly dominate. Hence we restrict our discussion of relations between submodularity and the existence of Monge sequences to the two-dimensional case. The following notion arises naturally in the lattice framework: a *submodular sequence* for a two-dimensional $n \times m$ array $C = (c[i,j])$ is a total ordering of the nm pairs (i,j) such that, for any i, j, k and ℓ, at least one of the pairs $(i,j) \wedge (k,\ell)$ or $(i,j) \vee (k,\ell)$ precedes both pairs (i,j) and (k,ℓ).

Proposition 4.1 *A two-dimensional array is submodular (or, equivalently, satisfies the Monge condition) if and only if every submodular sequence is a Monge sequence.*

It is not true that, in a submodular array, every Monge sequence is a submodular sequence. For example, consider a constant array C where all entries $c[i,j]$ have the same value. Then every sequence of the pairs (i,j) is a Monge sequence. However, the next result shows that Monge sequences coincide with submodular sequences when C satisfies a *strict* version of the Monge condition:

Proposition 4.2 *If a two-dimensional array C defines a* strictly *submodular function then a total ordering of the pairs (i,j) is a submodular sequence if and only if it is a Monge sequence.*

References

[1] Adler, I., A.J. Hoffman, and R. Shamir (1990), "Monge and feasibility sequences in general flow problems," Rutcor Research Report 70–90, Rutgers University.

[2] Adler, I. and R. Shamir (1990), "Greedily solvable transportation networks and edge-guided vertex elimination," Rutcor Research Report 39–91, Rutgers University.

[3] Aggarwal A. and M.M. Klawe (1990), "Applications of generalized matrix searching to geometric algorithms," *Discrete Applied Mathematics* **27**, 3–23.

[4] Aggarwal A., M.M.Klawe, S. Moran, P. Shor, and R. Wilber (1987) "Geometric applications of a matrix-searching algorithm," *Algorithmica* **2**, 195–208.

[5] Aggarwal A. and J.K. Park (1989) "Sequential searching in multi-dimensional monotone arrays," Research Report RC 15128, IBM T.J. Watson Research Center, Yorktown Heights, NJ.

[6] Aggarwal A. and J.K. Park (1989) "Parallel searching in multi-dimensional monotone arrays," Research Report RC 14826, IBM T.J. Watson Research Center, Yorktown Heights, NJ.

[7] Aggarwal A. and J.K. Park (1992) "Improved algorithms for economic lot-size problems," to appear in *Operations Research.*

[8] Balas, E. and M.J. Saltzman (1989), "Facets of the three-index assignment polytope," *Discrete Applied Mathematics* **23**, 201–229.

[9] Bandelt, H.J., Y. Crama and F.C.R. Spieksma (1992), "Approximation algorithms for multidimensional assignment problems with decomposable costs," to appear in *Discrete Applied Mathematics.*

[10] Bein, W.W., P. Brucker and P.K. Pathak (1991), "Monge properties in higher dimensions," Preprint 134, Universität Osnabrück.

[11] Bein, W.W., P. Brucker, J.K. Park and P.K. Pathak (1992), "A Monge property for the d-dimensional transportation problem," to appear in *Discrete Applied Mathematics.*

[12] Birkhoff, G., *Lattice Theory,* (3rd ed.) American Mathematical Colloquium Publications, Vol 25, 1967.

[13] Crama, Y. and F.C.R. Spieksma (1992), "Approximation algorithms for three-dimensional assignment problems with triangle inequalities", *European Journal of Operational Research* **60**, 273–279.

[14] Edmonds, J. (1970), "Submodular functions, matroids and certain polyhedra," in R. Guy et al., edrs, *Combinatorial Structures and Their Applications,* Gordon and Breach, 69–87.

[15] Edmonds, J. (1971), "Matroids and the greedy algorithm," *Mathematical Programming* **1**, 127–136.

[16] Edmonds, J. and R. Giles (1977), "A min-max relation for submodular functions on graphs", *Annals of Discrete Mathematics* **1**, 185–204.

[17] Frank, A. and É. Tardos (1988), "Generalized polymatroids and submodular flows," *Mathematical Programming* **42**, 489–563.

[18] Frieze, A.M. and J. Yadegar (1981), "An algorithm for solving 3-dimensional assignment problems with applications to scheduling a teaching practice," *Journal of the Operational Research Society* **32**, 989–995.

[19] Fujishige, S., *Submodular Functions and Optimization*, North Holland, 1991.

[20] Fujishige, S. and N. Tomizawa (1983), "A note on submodular functions on distributive lattices," *Journal of the Operations Research Society of Japan* **26**, 309–318.

[21] Granot, F. and A.F. Veinott, Jr. (1985), "Substitutes, complements and ripples in network flows," *Mathematics of Operations Research* **10**, 471–497.

[22] Haley, K.B. (1962) "The solid transportation problem," *Operations Research* **10**, 448–463.

[23] Hoffman, A.J. (1963), "On simple linear programming problems," in V. Klee, edr., *Convexity: Proceedings of the Seventh Symposium in Pure Mathematics*, Proceedings of Symposia in Pure Mathematics, Vol. 7, American Mathematical Society, 317–327.

[24] Hoffman, A.J. (1974), "A generalization of max-flow min-cut," *Mathematical Programming* **6**, 352–359.

[25] Hoffman, A.J. (1985), "On greedy algorithms that succeed," in: I. Anderson, edr, *Surveys in Combinatorics*, Cambridge University Press, 97–112.

[26] Klawe, M.M. (1992), "Superlinear bounds for matrix searching problems," *J. of Algorithms* **13**, 55–78.

[27] Korte, B., L. Lovász and R. Schrader (1991), *Greedoids*. North Holland.

[28] Larmore, L.L. and B. Schieber (1991), "On-line dynamic programming with applications to the prediction of RNA secondary structure," *J. of Algorithms* **12**, 490–515.

[29] Lovász, L. (1983), "Submodular functions and convexity," in A. Bachem et al., edrs, *Mathematical Programming — The State of The Art*, Springer, 235–257.

[30] Nemhauser, G.L., and L.A. Wolsey (1988), *Integer and Combinatorial Optimization*, Wiley.

[31] Park, J.K. (1991), "A special case of the n-vertex traveling salesman problem that can be solved in $O(n)$ time," *Information Proc. Let.* **40**, 247–254.

[32] Pierskalla, W.P. (1968), "The multidimensional assignment problem," *Operations Research* **16**, 422–431.

[33] Ruediger, R. (1992), personal communication.

[34] Ruediger, R. (1992), "Recognition of d-dimensional Monge arrays," Report Nr. 230, Institute für Mathematik, Technische Universität Graz.

[35] Sankoff, D and J.B. Kruskal, edrs, (1983), *Time Warps, String Edits, and Macromolecules: The Theory and Practice of Sequence Comparison*, Addison-Wesley.

[36] Shamir, R. and B.L. Dietrich (1990), "Characterization and algorithms for greedily solvable transportation problems," *Proc. 1st ACM/SIAM Symposium on Discrete Algorithms*, 358–366.

[37] Topkis, D.M. (1978) "Minimizing a submodular function on a lattice," *Operations research* **26**, 305–321.

[38] Veinott, A.F., Jr. (1989), "Representation of general and polyhedral subsemilattices and sublattices of product spaces," *Linear Algebra and its Applications* **114/115**, 681–704.

[39] Wu, X. (1991) "Optimal quantization by matrix searching," *J. of Algorithms* **12**, 663–673.

[40] Yao, F.F. (1980) "Efficient dynamic programming using quadrangle inequalities," in *Proc. 12th ACM Symp. on Theory of Computing*, 429–435.

[41] Yemelichev, V.A., M.M. Kovalev, and M.K. Kratsov (1984), *Polytopes, Graphs and Optimisation*. Cambridge University Press.

Eigenvalues and Graph Partitioning bounds

Franz Rendl
Technische Universität Graz
Institut für Mathematik
Steyrergasse 30, A-8010 Graz, Austria

August 1992

Let $G = (N, E)$ be an edge-weighted undirected graph with node set $N = \{1, \ldots, n\}$, edge set E and weights w_{ij}, $ij \in E$. We consider the problem of partitioning the node set N into k disjoint subsets $S_1, \ldots, S_k$ of specified sizes $m_1 \geq m_2 \geq \ldots \geq m_k$, $\sum_{j=1}^{k} m_j = n$, so as to minimize the total weight of the edges connecting nodes in distinct subsets of the partition. This problem is well known to be NP-hard and therefore finding an optimal solution is likely to be a difficult task. We focus on bounding the optimal partitioning. A survey on related problems can be found in e.g. [4].

An instance of a graph partitioning problem is described by the (symmetric) adjacency matrix A of size n and an integer vector $m = (m_1, \ldots, m_k)$, defining the specified sizes for the subsets of the partition. The vector u is a vector consisting of ones. Finally we denote by $w(E)$ the sum of all edge weights of G, i.e. $w(E) = u^t Au/2$, and by $w(E_{cut})$ the total weight of the edges cut by an *optimal* partition. Moreover let $w(E_{uncut}) := w(E) - w(E_{cut})$. The following nonlinear optimization problem solves the graph partitioning problem, see e.g. [5].

$$\textbf{(GP)} \quad w(E_{uncut}) = \max \frac{1}{2} \mathrm{tr} X^t A X$$

such that

$$X^t X = \mathrm{diag}(m) \tag{1}$$

$$X u_k = u_n; \quad X^t u_n = m \tag{2}$$

$$X \geq 0. \tag{3}$$

The constraints guarantee that all entries of the $n \times k$ matrix X are either 0 or 1. The nonzero entries of column j of X represent the nodes contained in S_j.

We will use the model (GP) to obtain tractable relaxations for graph partitioning and hence at least bounds on $w(E_{uncut})$.

Dropping the constraints (2) and (3) leads to one of the first relaxations for graph partitioning. It was proposed by Donath and Hoffman in the 1970s [1]. $\lambda_j(M)$ denotes the j-largest eigenvalue

of a (symmetric) matrix M.

$$w(E_{uncut}) \leq \max\{\frac{1}{2}\mathrm{tr}X^tAX : X \text{ satisfies (1)}\} = \frac{1}{2}\sum_{j=1}^{k} m_j\lambda_j(A). \tag{4}$$

Any X containing pairwise orthogonal eigenvectors x_j corresponding to $\lambda_j(A)$ and having the correct length $\|x_j\|^2 = m_j$ constitutes a maximand in (4).

The Donath-Hoffman bound can be further strengthened by dropping only the nonnegativity conditions from (GP), see [5]. In the case where the m_j are all equal (to n/k), the linear term in the bound is constant. Assumption: $m_1 = \ldots = m_k = n/k$. We denote by V_n an arbitrary matrix satisfying $V^tu = 0$, $V^tV = I_{n-1}$. In other words the columns of V_n are orthonormal and orthogonal to u.

$$w(E_{uncut}) \leq \max\{\frac{1}{2}\mathrm{tr}X^tAX : X \text{ satisfies (1),(2)}\} = \frac{n}{2k}\sum_{j=1}^{k-1}\lambda_j(V_n^tAV_n) + \frac{1}{2k}s(A). \tag{5}$$

This upper bound is attained for

$$X = \frac{1}{k}u_nu_k^t + \sqrt{\frac{n}{k}}V_nZV_k^t, \tag{6}$$

where Z contains a set of $k-1$ orthonormal eigenvectors corresponding to the largest $\lambda_j(V_n^tAV_n)$.

A further improvement can be achieved along the following lines [1, 5]. Let $d \in \Re^n$ and X be an arbitrary feasible partition, i.e. X satisfies (1,2,3). Then it can readily be seen that

$$\mathrm{tr}\ X^t(\mathrm{diag}(d) - \frac{d^tu}{n}I)X = 0.$$

Therefore, see [5], it can be shown that

$$w(E_{uncut}) \leq \frac{1}{2k}s(A) + \frac{n}{2k}\sum_{j=1}^{k-1}\lambda_j(V_n^t(A + \mathrm{diag}(d) - \frac{d^tu}{n}I)V_n) =: \frac{1}{2k}s(A) + \frac{n}{2k}f(d).$$

This leads to the following bound for partitioning the nodes into subsets of equal size, see [5]:

$$w(E_{uncut}) \leq \frac{s(A)}{2k} + \min\{\frac{n}{2k}f(d) : d \in \Re^n\}. \tag{7}$$

The corresponding maximizer X can be used to generate "good" partitions, by either rounding or solving transportations problems, see [5], [2] for further details.

We conclude with some numerical results on graphs used by Johnson et al [3] to test graph bisection heuristics. In the following table n denotes the size of the (unweighted) graph of cardinality $|E|$. We further provide the density in %. The column labeled "Upper Bound" contains the upper bound (7), using tools from matrix analysis to calculate the largest eigenvalue, and tools from nonlinear optimization to carry out the minimization. We consider bisecting the graph, i.e. $k = 2$. The column labeled "Lower Bound" contains a bisection obtained by rounding

and limited local improvement. We also provide the relative gap (in %) between lower and upper bound, and include the best solutions described in [3]. It is interesting to see that the present approach yields not only tight bounds, but also produces at low computational cost feasible solutions that are highly competitive with solutions obtained after extensive experiments using simulated annealing and the Kernighan-Lin heuristic.

n	$\|E\|$	Density (%)	Upper bd	Lower bd	% gap	Johnson
124	149	2	141	136	3.7	136
124	318	4	271	254	6.7	255
124	620	8	467	442	5.6	442
124	1271	17	853	822	3.8	822
250	331	1	316	301	5.0	302
250	612	2	531	495	7.3	498
250	1283	4	981	925	6.1	926
250	2421	8	1675	1588	5.5	1593
500	625	0.5	600	573	4.7	573
500	1223	1	1071	1001	7.0	1004
500	2355	2	1844	1713	7.6	1727
500	5120	4	3564	3358	6.1	3376
1000	1272	0.25	1228	1172	4.8	1170
1000	2496	0.5	2193	2030	8.0	2045
1000	5064	1	3958	3676	7.7	3697

References

[1] W.E. DONATH and A.J. HOFFMAN. Lower bounds for the partitioning of graphs. *IBM J. of Research and Developement*, 17:420–425, 1973.

[2] J. FALKNER, F. RENDL and H. WOLKOWICZ. A computational study of graph partitioning. Technical Report, TU Graz, 1992.

[3] D.S. JOHNSON, C.R.ARAGON, L.A. MCGEOCH, and C. SCHEVON. Optimization by simulated annealing: an experimental evaluation; part 1, graph partitioning. *Operations Research*, 37:865–892, 1989.

[4] T. LENGAUER. *Combinatorial algorithms for integrated circuit layout.* John Wiley and Sons, Chicester, 1990.

[5] F. RENDL and H. WOLKOWICZ. A projection technique for partitioning the nodes of a graph. Technical Report CORR 90-20, University of Waterloo, Waterloo, Canada, 1990.

New heuristics for decomposing traffic matrices in TDMA satellite communication (extended abstract)

Günter Rote, Andreas Vogel
Technische Universität Graz, Institut für Mathematik
Steyrergasse 30, A-8010 Graz, Austria.
electronic mail: rote@ftug.dnet.tu-graz.ac.at

1. Introduction

Background and description of the problem. In satellite communication, one satellite can serve several radio stations on earth. In order to allow signals to be sent from each radio station to each other radio station, the TDMA (time division multiple access) technique is used. At any instant, the satellite is set to a fixed *switching mode*: All radio stations transmit and receive data simultaneously, and the switching mode determines for each radio station the radio station which receives the data which the former transmits. In mathematical terms, a switching mode is a one-to-one mapping on the set of radio stations, i. e., a permutation. The satellite time-multiplexes regularly between different switching modes in short intervals, according to a fixed cyclic schedule.

The communication needs between the radio stations are given by a matrix $T = (t_{ij})$, the *traffic matrix*. t_{ij} is the amount of information per time unit that has to be transmitted from the i-th to the j-th radio station. More information on the technical background can for example be found in Burkard [1985]. We consider the problem of setting up a schedule for the satellite, i. e., a sequence of switching modes and a duration for each switching mode. Formally, the *matrix decomposition problem* can be stated as follows:

> Given a nonnegative $n \times n$ matrix $T = (t_{ij})$, find a *decomposition* of T, i. e., a sequence of permutation matrices $P^1, P^2, ..., P^q$ and a sequence of nonnegative weights $l_1, l_2, ..., l_q$ such that
>
> $$T \le \sum_{k=1}^{q} l_k P^k \quad \text{(elementwise)} . \tag{1}$$

The total duration d of the decomposition is given by $d = \sum_{k=1}^{q} l_k$.

The first goal in setting up a switching schedule is of course to keep the total duration as small as possible. On the other hand, every change of the switching mode incurs a certain overhead and loss of time. Therefore, the number of matrices, q, should not be too large. There is clearly a trade-off between the two objectives, d and q.

Related results and results of the present paper. Inukai [1979] and Burkard [1985] have shown that the optimal total duration is equal to t^*, the maximum row or column sum of the traffic matrix, but in general, a time-optimal decomposition may require up to $n^2 - 2n + 2$ matrices, which is too large for practical purposes. Burkard [1985] has also given algorithm which constructs such a decomposition in $O(n^4)$ steps.

This work was supported by the Fonds zur Förderung der wissenschaftlichen Forschung, Project S32/01.

It is clear that any decomposition must consist of at least n matrices, unless some entries in the traffic matrix are zero. Gopal and Wong [1985] and Rendl [1985] have shown that the problem of constructing a shortest decomposition into at most n matrices is NP-complete. Thus, it makes sense to look for heuristics. The currently best heuristic for decomposing into n matrices is due to Balas and Landweer [1983]. A more extensive review of results concerning the matrix decomposition problem can be found in Burkard [1991].

We propose a simple and fast "scaling" heuristic for constructing a short schedule with a given upper bound Q on the number q of switching modes. We can prove a relative error guarantee for the total duration d of the decomposition. The method is not applicable if $Q = n$ or Q exceeds n only slightly. When Q is somewhat larger than n (of the order $2n$ or $3n$), the error bound is still very crude, but it improves as the ratio of Q and n increases.

As a subproblem, we address the problem of decomposing a matrix under the constraint that the lower bound t^* on the total duration has to be achieved; the number q of matrices remains as the objective function to be minimized. The traffic matrices that we have in mind for this problem are matrices with small integer entries. Here we use two heuristics: one based on a bottleneck assignment problem and on matching techniques, and a more powerful one which solves maximum flow problems.

As a side issue, we mention that one other subproblem that we have to solve has some connections with voting systems.

Finally we present the results of numerical experiments measuring the actual behavior of our heuristics for some randomly generated problems. We compare our algorithm to the heuristic of Balas and Landweer [1983] for decomposing into only n matrices. The experiments show that the heuristics performs very well, and that the method might be of practical interest.

A full version of this abstract is available as a technical report, Rote and Vogel [1990].

2. The heuristics

Our algorithm is based on the simple idea of scaling the entries of the given traffic matrix and rounding them to small integers. A matrix with small integers will require a small number q of matrices for decomposition; theoretically, we will utilize the trivial upper bound t^* on the number q of required permutation matrices. (Recall that t^* is the maximum row or column sum of the traffic matrix.)

Globally, the algorithm runs as follows:

Input: A non-negative real $n \times n$ matrix T.

(a) *Choose* some "unit" $F > 0$.

(b) Round the entries of the matrix upwards to the next multiple of F: $\overline{t}_{ij} := F \cdot \left\lceil \frac{t_{ij}}{F} \right\rceil$.

(c) Solve the matrix decomposition problem for the resulting matrix $\overline{T}$ (or equivalently, for the *integer* matrix $(u_{ij}) := (\lceil t_{ij}/F \rceil)$ obtained by dividing through F).

(d) The resulting decomposition can be adjusted downwards to compensate for the rounding up in step (b).

The quality of the solution produced depends first of all on the choice of F in step (a). The idea is to choose F so large that the matrix (u_{ij}) consists of small integers and only few permutation matrices are needed for its duration-optimal decomposition in step (c), and so small that the error incurred in the rounding in step (b) is not too large. By choosing F appropriately, we will be able to give a performance guarantee for the quality of the solution produced by the heuristic.

Step (c) is the heart of the algorithm. The principal goal of this step, a duration-optimal decomposition, is relatively easy to achieve, but we also want *few* matrices, since their number will be the number of matrices that the solution will have. We shall discuss three methods for carrying out this step.

Method I — simple and fast: edge coloring

Lemma 1. *An integer $n \times n$ matrix U with maximum row and column sum u^* can be decomposed into $q = u^*$ permutation matrices.*

Proof: If we restrict the problem to decomposition into "unit" permutation matrices, i. e., we allow only weights $l_k = 1$ in (1), then we get essentially an edge coloring problem for a bipartite multigraph with vertices r_i and c_j $(i = 1, \ldots, n)$, with u_{ij} parallel edges between r_i and c_j. In a bipartite (multi-)graph, the number of colors required (the chromatic index) equals the maximum degree, which is equal to u^* in our case.

The currently fastest edge coloring algorithm is the one of Cole and Hopcroft [1982], which leads to a time complexity of $O(u^* n \log n)$ for carrying out the decomposition. The bound $q \le u^*$ of lemma 1 is tight if and only if $u^* \le (n+1)^2/4$ (see the appendix of Rote and Vogel [1990]).

The next lemma, which relates F and the quality of the solution, can be proved by quite straightforward calculations.

Lemma 2. *If F is chosen as the smallest value such that $u^* \le Q$, for some given value $Q \ge n$, then the following relation holds between the maximum row and column sum Fu^* of the rounded-up matrix and the corresponding value t^* of the original matrix:*

$$Fu^* \le \frac{Q}{Q-n+1} \cdot t^*.$$

By simply choosing F in our algorithm as the smallest value which gives $u^* \le Q$, the preceding two lemmas yield the following theorem:

Theorem. *An $n \times n$ matrix T can be decomposed into a weighted sum of no more than Q permutation matrices $(Q \ge n)$ with a total duration that is within a factor of $Q/(Q-n+1)$ of the value t^* that is obtainable without restriction on the number of matrices in the decomposition.*

For a *positive* matrix U the analysis of lemma 1 can be refined by the maximum flow techniques discussed below to yield a bound of $q \le \lceil (u^* + n)/2 \rceil$. This allows us to improve the bound $Q/(Q-n+1)$ in the above theorem to $(Q - n/2)/(Q - n + 1/2)$ (for arbitrary nonnegative matrices).

To actually determine F, we look at each row and column individually, find the smallest F such that the sum of $\lceil t_{ij}/F \rceil$ in this row or column is at most Q, and take the

maximum of all those F's. The problem of computing for the, say, first row of T, the smallest F such that $\sum_{j=1}^{n} \lceil t_{1j}/F \rceil \le Q$ occurs also in proportional voting systems and the theory of apportionment (cf. e. g. Woodall [1982] or Balinski and Young [1983]). In this case, the number F is the quota. Algorithmically, it can be determined as the $(Q-n+1)$-largest element in the (infinite) array $(t_{1i}/l)_{1 \le i \le n, l \ge 1}$. This array has sorted rows. We remove from each row j of the array the first $\lfloor (Q-n)t_{1j} / \sum_{k=1}^{n} t_{1k} \rfloor$ elements. In this way we remove between $Q-2n$ and $Q-n$ elements, and F is the k-largest element in the remaining array, where k is some number is between 1 and n. This element can be found in $O(n)$ time by the method of Frederickson and Johnson [1982]. There are also simple and practical methods which use priority queues and take $O(n \log n)$ time. In any case the complexity of the algorithm is dominated by step (c), and it is $O(Qn \log n)$.

Method II — greedy: bottleneck assignment

There is no way to use the edge coloring for a decomposition into fewer than u^* matrices. In practice, we would like to have an algorithm which uses as few matrices as possible because this would allow us to choose F smaller, and we would lose less in the rounding-up of step (b).

We try to reduce q by the following greedy strategy: We select the *first* weighted permutation matrix $l_1 P^1$ in such a way that the maximum row and column sum of the remaining traffic matrix $\max(U - l_1 P^1, 0)$ is reduced by as much as possible. This will reduce the bound u^* on the number (and, hopefully, also the actual number) of further matrices which will be needed in the decomposition. We continue this strategy with the remaining matrix until we are done.

By analyzing the condition that the maximum row and column sum of U must be reduced by l_1 when the matrix $\min(l_1 P^1, U)$ is subtracted, we get the following condition on P^1:

$$\text{If} \quad P^1_{ij} = 1 \quad \text{then} \quad l_1 \le u_{ij} + (u^* - \max\{r_i, c_j\}). \tag{2}$$

Let us interpret this formula. For *critical* rows (and columns), i. e., rows with $r_i = u^*$, l_1 must be $\le u_{ij}$. Non-critical rows and columns have some *slack* $u^* - r_i$ or $u^* - c_j$, respectively, which allows to weaken this inequality for their elements: The smaller of the row slack and the column slack for each element can be added to the bound u_{ij}.

Finding the maximum value of l_1 for which such a permutation matrix P^1 exists is a bottleneck assignment problem whose cost matrix is given by the right side of (2). After determining P_1 and l_1, $\min(l_1 P^1, U)$ is subtracted from U. P^2 and l_2 are determined by the same procedure for the remaining matrix, and so on.

The sequence $(l_1, l_2, \ldots, l_q)$ is weakly decreasing and consists of small numbers. Therefore we solve the bottleneck assignment problem by testing for successive values of l_k in decreasing order. Each test amounts to finding a complete bipartite matching. For this purpose, we can use the procedure of Hopcroft and Karp [1973], which requires $O(n^{5/2})$ steps, and thus method II can be carried out in $O((q + l_1)n^{5/2})$ time.

In our computational experiments we have additionally reduced l_k in each step, if necessary, in order to ensure that the graph defined by (2) has no isolated vertices. This was sufficient to eliminate most of the unsuccessful tests.

Method III — even greedier: maximum network flow

By solving a maximum flow problem on a suitably defined graph, we can try to determine *all* permutation matrices P^k which belong to a group of equal l_k's *simultaneously*. Our maximum flow problem is an extension of the matching problem of method II.

We set up a network which has two nodes for each row i: A regular source node r_i and a "slack" node $\bar{r}_i$. Similarly, there is a sink node c_j and a column slack node $\bar{c}_j$ for each column j. For each entry u_{ij} we have now two arcs: There is a "direct" arc from r_i to c_j of capacity $\lfloor u_{ij}/l_k \rfloor$. This capacity counts how often a permutation may use the entry u_{ij} by reducing row and column sums r_i and c_j. In addition, there is a "pseudo-arc" from $\bar{r}_i$ to $\bar{c}_j$ of infinite capacity. Using this arc corresponds to reducing the slacks $u^* - r_i$ and $u^* - c_j$ by l_j. This usage is restricted by the capacities of the "slack" arcs from r_i to $\bar{r}_i$ of capacity $\lfloor (u^* - r_i)/l_k \rfloor$ and from $\bar{c}_j$ to c_j of capacity $\lfloor (u^* - c_j)/l_k \rfloor$, which precede and follow the pseudo-arc. However, when $u^* - r_i < l_k$, we set the capacity of the slack arc $(r_i, \bar{r}_i)$ to 1 instead of 0, but at the same time we eliminate all arcs out of $\bar{r}_i$ for which $(u_{ij} \bmod l_k) + u^* - r_i < l_k$. We do the same for all columns.

This last modification ensures that, if an entry u_{ij} should be usable by criterion (2), then there is a way to send at least one unit of flow from r_i to c_j. The remainder $(u_{ij} \bmod l_k)$, which cannot be "used up" by the flow on the direct edge (r_i, c_j) of capacity $\lfloor u_{ij}/l_k \rfloor$, is put together with the slack $u^* - r_i$ to see whether a total of l_k can be reached.

We place a constant supply and demand of value g at each source vertex r_i and at each sink vertex c_j, respectively. The largest value of g such that a flow satisfying all these supplies and demands exists is the number of permutation matrices of weight l_k into which we can decompose the matrix. To find those g matrices, we decompose the flow into "permutation flows". This can be done by the coloring algorithms of Cole and Hopcroft [1982], which we used in method I. If $g > 0$, we have to try the same value of l_k again, since we may not have exhausted all possible matrices with weight l_k. If we got $g = 0$, we decrease l_k by one and try again.

3. Computational results

We have programmed methods II and III and applied them to randomly generated test problems of various sizes in the range from $n = 10$ to $n = 100$. For the various subprocedures, we used the simplest possible implementations that did not take too much time.

We have tried to get approximately $q \approx 2n$ matrices. Some target value M for u^*, which was determined experimentally, was an input parameter of the program, and we computed F by the formula $F = 1/2 \cdot (t^*/(M - n + 1) + t^*/M)$. We computed u^* with this value of F, and then we reduced F as much as possible while still keeping the same u^*. With this procedure, the desired number of $q = 2n$ matrices was achieved reasonably precisely. For step (d), we successively lowered each l_k, for k from 1 to q, as much as possible while still maintaining the relation $\sum_{k=1}^{q} l_k P^k \geq T$.

The results are shown in the following table. The numbers are averages of 100 matrices each, with random integer entries in the range 1–100 for method II (and also in the last column), and in the range 1–1000 for method III. CPU-times are given in seconds on a DEC VAX 11/785 computer. The measure of the solution quality, the total duration, is normalized in terms of the relative excess over the lower bound t^*, in order to make the

results comparable for different types of matrices. For contrast, the last column contains the results of the heuristic of Balas and Landweer [1983] for decomposing into only $q = n$ matrices. Of course, this is not a fair comparison because their heuristic solves a more restricted problem. Still, one can see how much may be gained in total duration by allowing more than n matrices, in particular for smaller n.

n	Method II, $q \approx 2n$, entries 1–100			Method III, $q \approx 2n$, entries 1–1000			B&L, $q = n$
	M	CPU-time	$(d - t^*)/t^*$	M	CPU-time	$(d - t^*)/t^*$	$(d - t^*)/t^*$
10	220	0.20	1.007%	350	0.87	0.598%	7.84%
20	450	1.16	1.480%	720	4.49	0.993%	5.13%
30	700	3.40	1.648%	1200	12.68	1.014%	5.47%
40	1050	7.57	1.596%	1800	27.49	0.962%	4.09%
50	1400	14.07	1.504%	2500	51.31	0.884%	2.58%
60	1750	23.51	1.460%	3200	84.62	0.847%	2.36%
80	2600	53.09	1.375%	5000	193.25	0.743%	2.62%
100	3900	100.57	1.121%	7000	361.22	0.683%	1.78%

References

Balas, E., and P. R. Landweer [1983], Traffic assignment in communication satellites. *Operations Research Letters* **2**, 141–147.

Balinski, M. L., and H. P. Young [1982], *Fair Representation — Meeting the Ideal of One Man, One Vote*. Yale University Press, New Haven and London.

Burkard, R. E. [1985], Time-slot assignment for TDMA-systems. *Computing* **35**, 99–112.

Burkard, R. E. [1991], Time division multiple access systems and matrix decomposition. *Proceedings of the Fourth European Conference on Mathematics in Industry (ECMI 4)*, (H. Wacker and W. Zulehner, eds.), B. G. Teubner, Stuttgart, and Kluwer Academic Publishers, Dordrecht, pp. 35–46.

Cole, R., and J. Hopcroft [1982], On edge coloring bipartite graphs. *SIAM J. Computing* **11**, 540–546.

Frederickson, G. N., and D. B. Johnson [1982], The complexity of selection and ranking in $X + Y$ and matrices with sorted columns. *J. Computer System Sciences* **24**, 197–208.

Gopal, I. S., and C. K. Wong [1985] Minimizing the number of switchings in an SS/TDMA system. *IEEE Trans. Comm.* **COM-33**, 497–501.

Hopcroft, J. E., and R. M. Karp [1973], An $n^{5/2}$ algorithm for maximum matching in bipartite graphs. *SIAM J. Computing* **2**, 225–231.

Inukai, T. [1979], An efficient SS/TDMA time slot assignment algorithm. *IEEE Transactions on Communication* **COM-27**, 1449–1455.

Rendl, F. [1985], On the complexity of decomposing matrices arising in satellite communication. *Operations Research Letters* **4**, 5–8.

Rote, G., and A. Vogel [1990], A heuristic for decomposing traffic matrices in TDMA satellite communication. Report 1990-73, Technische Universität Graz, Inst. f. Mathematik, Feb. 1990, 28 pp.; submitted.

Woodall, D. R. [1986], How proportional is proportional representation? *Math. Intelligencer* **8** (4), 36–46.

Parallel Simulated Annealing and its Relation to Evolutionary Algorithms

Günter Rudolph

rudolph@ls11.informatik.uni-dortmund.de

University of Dortmund

Department of Computer Science

P.O. Box 50 05 00

W-4600 Dortmund 50 / FRG

Extended Abstract

1 Introduction

Simulated annealing (SA) [17] is a widely used method for combinatorial optimization problems. If this method is designed for optimization over continuous variables, i.e. $\min\{f(x) \mid x \in M \subseteq \mathbb{R}^n\}$, a close relationship between Simulated Annealing and so-called Evolution Strategies (ES) [21][24][25] can be noticed. This is why several results for ES algorithms can be used to design efficient parallel SA algorithms.

The study of convergence to the global minimum of SA has mainly concentrated on the case of a finite or countable state space (see e.g. the review of Romeo and Sangiovelli-Vincentelli [22]). For continuous state spaces there are results in form of stochastic differential equations [1][10][11], whereas a proof of original SA is given by Haario and Saksman [13] for general state spaces.

In this paper the differences and similarities of SA and ES algorithms and their implications for convergence results are investigated. It turns out that it is necessary to adapt the sampling distributions over time to achieve a reasonable convergence rate and it is shown by example that the gain of multiple trials from a single point is low. Therefore the algorithm is modified so that it can be executed on a SIMD parallel com-

puter. This algorithm is much more reliable than multiple trial SA which is supported by some preliminary test results.

2 Markovian Optimization Algorithms

Sequential variants of SA and ES algorithms can be studied in the general framework of markovian processes. The general algorithmic frame can be formulated as:

choose $X_0 \in M \subseteq \mathbb{R}^n$ and set $t = 0$
repeat
$Y_{t+1} = X_t + Z_t$
$X_{t+1} = Y_{t+1} \cdot a(X_t, Y_{t+1}; .) + X_t \cdot (1 - a(X_t, Y_{t+1}; .))$
increment t
until termination criterion applies

where $a(x, y; .)$ denotes the acceptance function which may depend on additional parameters. The distribution of random vector Z_t is chosen to be symmetric, i.e. $z \stackrel{d}{=} Bz$ for every orthogonal matrix B. In this case z may be expressed in its stochastic representation $z \stackrel{d}{=} r\,u$, where r is a nonnegetive random variable and u a random vector uniformly distributed on a hypersphere surface of dimension n (see e.g. [9]). This reveals that the trial point generation mechanism of the above algorithm is equivalent to that of a random direction method with step size distribution r (see [19][20]).
Depending on the choice of the acceptance function $a(x, y; .)$ and of the generating distribution of z one gets a family of markovian optimization algorithms which can be identified by a sequence of transition probabilities $(P_t)_{t \in \mathbb{N}}$:

$$P_t(x, A) = \int_A Q_t(x, d\omega)\, q_t(x, \omega)\, d\omega + 1_A(x) \int_M Q_t(x, d\omega)\,(1 - q_t(x, \omega))\, d\omega \qquad (1)$$

with $A \subseteq M$, $x \in M$ and where Q denotes the generating distribution, 1_A the characteristic function of set A and q_t the acceptance probability function which is related to the acceptance function a_t via

$$a_t(x, y; \xi) = 1_{[0, q_t(x, y; .)]}(\xi) \ , \qquad (2)$$

where ξ is a random variable uniformly distributed on $[0, 1]$. Typical examples are:

$$q(x, y; .) \ = \ 1_{\mathbb{R}_0^+}(f(x) - f(y) + T) \qquad (3)$$

$$q(x,y;.) = 1_{\mathbf{R}^+}(f(x)-f(y)) \quad (4)$$

$$q(x,y;.) = 1_{\mathbf{R}_0^+}(f(x)-f(y)) + 1_{\mathbf{R}^-}(f(x)-f(y)) \cdot \exp\left(-\frac{f(x)-f(y)}{T}\right) \quad (5)$$

where (3) is used by *threshold accepting* methods proposed by Dueck and Scheuer [8] for combinatorial problems and tested by Bertocchi and Di Ordoardo [2] for continuous variables, whereas (4) is applied by *evolution strategies* and (5) by *simulated annealing* .

Usually, the sampling distribution is chosen to be a uniform distribution on *bounded* regions, e.g. fixed [31][16][30] or adapted hypercubes [29][14][6] and fixed [4] or adapted hypersphere surfaces [3]. Since it is not possible with those distributions to reach each state in M when being trapped in a local minimum one has to provide the algorithm with the chance to perform some steps with worse objective function values to escape from local minima. This is realized by using (5) in (2). However, in order to establish convergence at all the probability of accepting a worse point has to be decreased to zero over time. The result of Haario and Saksman [13] indicates that the rate of decrease has to be logarithmic as in the finite case.

Using this cooling schedule the rate of convergence is rather slow. Hence, other schedules are used in practical applications (see table 1) which serve with faster but possibly nonglobal convergence. This problem can be circumvented by an appropiate choice of the generating distribution. Indeed, if M is bounded one might use the uniform distribution over M and global convergence for continuous functions follows from standard arguments (see e.g. [7]) with $T_t \equiv 0$. Szu and Hartley [27] claim that global convergence can be established by employing a multidimensional Cauchy distribution with density $g(x) = K_n T_t^{-1} (T_t^2 + ||x||^2)^{(n+1)/2}$, which concentrates around 0 caused by the schedule $T_t = T_0/(t+1)$. The advantage opposed to sampling distributions with bounded support is due to the fact that for each trial there exists a (small) probability to reach any state. Actually, under some conditions no cooling is necessary at all such that (5) becomes equivalent to (4) and global convergence can be guaranteed.

THEOREM 1 (see e.g. [26][18])

Let $f^* > -\infty$ and for the Lebesgue measure of the level sets $\mu(L_{f^*+\epsilon}) > 0$ for all $\epsilon > 0$. If

$$\sum_{t=0}^{\infty} Q(X_t, L_{f^*+\epsilon}) = \infty \quad \forall \epsilon > 0 \quad (6)$$

then $f(X_t) \to f^*$ with probability one. □

cooling type	schedule	references
geometrical	$T_t = c^t T_0$ with $c \in (0,1)$	[29][31][30][6][3]
subtractive	$T_t = \max\{0, T_0 - t\,\Delta T\}$	[14]
linear	$T_t = T_0/(t+1)$	[27][28]
function value	$T_t = \alpha f(X_t) + \beta$	[4]

Table 1: Typical cooling schedules used in practical applications

For instance, let $Q(Y_{t+1} - X_t)$ be multinormally distributed with zero mean and covariance matrix $C_t = \sigma_t^2 I$, where $\min\{\sigma_t \mid t > 0\} \geq \sigma > 0$ and $f(x) \to \infty$ for $||x|| \to \infty$. Then, the lower level sets are bounded and there exists a minimum positive probability to hit the level set $L_{f^*+\epsilon}$. Thus, $\liminf_{t\to\infty} Q(X_t, L_{f^*+\epsilon}) > 0$ and the sum in (6) diverges. Although global convergence of the above type should be the minimum requirement of a probabilistic algorithm it is more interesting to know something about the finite time behavior, i.e. the rate of convergence. For finite state space it is known that the convergence rate of the *probability* to reach the optimal state is of order $1 - O(t^{-a})$ with $a > 0$ depending on the problem (see Chiang and Chow [5]). This is slow convergence since the rate of convergence of pure random search is of order $1 - O(\beta^t)$ with $\beta \in (0,1)$. Another measure of convergence rate is the expected error defined by $\delta_t := \mathrm{E}[f(X_t) - f^*]$. It can be shown that $\delta_t = O(t^{-\alpha})$ for fixed sampling distributions even for strongly convex functions [19]. However, if the distribution of $Z_t \stackrel{d}{=} r_t u$ is adapted via $r_t = ||\nabla f(x_t)||\, r$, where r has nonvoid support on $(0,s)$, then one gets $\delta_t = O(\beta^t)$ with $\beta \in (0,1)$ for objective functions with "sufficiently spherical" level sets close to the minimizer, e.g. strongly convex functions [19][20]. It can be shown that a success/failure control as proposed in [21] can be used, too. If it would be possible to adapt and concentrate the sampling distribution to the lower level sets geometrical convergence of δ_t can be shown even for lipschitz–continuous function with several local minima [32].

3 Parallel Simulated Annealing

Markovian algorithms considered so far are not well suited for parallelization. For finite state space variants some proposals are surveyed in [12]. A straightforward way to take advantage of parallel hardware is to perform, say p, trials in parallel on p processors and to select the best move. This is the idea of so–called $(1,p)$-Evolution

Strategies [25] and it can be used for SA as well [3]. However, a simple example reveals that the speedup is poor even for convex functions: Let $f(x) = ||x||^2$ and $M = \mathbb{R}^n$. Then the convergence rate of the sequential version using criterion (4) can be estimated to be $\delta_t = (1 - 0.405/n)^t (f(x_0) - f^*)$ for large n, whereas for the $(1,p)$-ES holds $\delta_t \approx (1 - 2\log(p)/n)^t (f(x_0) - f^*)$. It follows that the expected speedup is $\mathrm{E}[S_p] = O(\log p)$.

Another straightforward parallelization scheme is to run the sequential algorithm on p processors independently [3] as a parallel version of the well-known multistart technique. This is well suited for SIMD parallel computers which perform the same instruction on p processors in parallel but on different data streams as well as for MIMD parallel computers which can simulate SIMD programs [23].

The general idea of Evolutionary Algorithms is to view the trial vector as the *genome* of an *individual* that is *mutated* by the sampling distribution. Choosing the better trial point for the next iteration/generation can be regarded as *selection* . It can be shown that simply placing one individual on each processor (the processors are arranged in a torus topology) and performing selection among the nearest neighbors is less reliable than parallel multistart. However, introducing a *recombination* mechanism provides the parallel algorithm with a new quality: Before reproducing a new trial point, another individual is selected from the neighborhood and the genomes are merged. With this mechanism it is possible to push individuals out of local minima. In biological terms, the population becomes or keeps more *diversity* increasing the chance to find the global minimum. Preliminarily test results support this hypothesis.

4 Acknowledgment

The author is supported under project 107 004 91 by the Research Initiative *Parallel Computing* of Nordrhein-Westfalen, Land of the Federal Republic of Germany.

References

[1] F. Aluffi-Pentini, V. Parisi, and F. Zirilli. Global optimization and stochastic differential equations. *Journal of Optimization Theory and Apllications*, 47(1):1–16, 1985.

[2] M. Bertocchi and C. Di Odoardo. A stochastic algorithm for global optimization based on threshold accepting technique. In Phua, Wang, Yeong, Leong, Loh, Tan, and Chou, editors, *Optimization Techniques and Applications*, volume 1. World Scientific, 1992.

[3] M. Bertocchi and P. Sergi. Parallel global optimization over continuous domain by simulated annealing. In P. Messina and A. Murli, editors, *Proceedings of Parallel Computing: Problems, Methods and Applications*, pages 87–97, Amsterdam, 1992. Elsevier.

[4] I.O. Bohachevsky, M.E. Johnson, and M.L. Stein. Generalized Simulated Annealing for function optimization. *Technometrics*, 28(3):209–217, 1986.

[5] T.-S. Chiang and Y. Chow. On the convergence rate of annealing processes. *SIAM Journal on Control and Optimization*, 26(6):1455–1470, 1988.

[6] A. Corana, M. Marchesi, C. Martini, and S. Ridella. Minimizing multimodal functions of continuous variables with the "Simulated Annealing" algorithm. *ACM Transactions on Mathematical Software*, 13(3):262–280, 1987.

[7] L. Devroye. Progressive global random search of continuous functions. *Mathematical Programming*, 15:330–342, 1978.

[8] G. Dueck and T. Scheuer. Threshold accepting: A general purpose optimization algorithm appearing superior to simulated annealing. Technical Report TR 88.10.011, IBM Scientific Center, Heidelberg, Germany, 1988.

[9] K.-T. Fang, S. Kotz, and K.-W. Ng. *Symmetric Multivariate and Related Distributions*, volume 36 of *Monographs on Statistics and Applied Probability*. Chapman and Hall, London and New York, 1990.

[10] S.B. Gelfand and S.K. Mitter. Simulated annealing type algorithms for multivariate optimization. *Algorithmica*, 6:419–436, 1991.

[11] S.B. Gelfand and S.K. Mitter. Weak convergence of markov chain sampling methods and annealing algorithms to diffusions. *Journal of Optimization Theory and Applications*, 68(3):483–498, 1991.

[12] D.R. Greening. Parallel simulated annealing techniques. *Physica D*, 42:293–306, 1990.

[13] H. Haario and E. Saksman. Simulated annealing process in general state space. *Adv. Appl. Prob.*, 23:866–893, 1991.

[14] L.M. Haines. The application of the annealing algorithm to the construction of exact optimal design for linear regression models. *Technometrics*, 29(4):439–448, 1987.

[15] W. Joosen and E. Milgrom, editors. *Parallel Computing: From Theory to Sound Practice*, Amsterdam, 1992. IOS Press.

[16] A. Khachaturyan. Statistical mechanics approach in minimizing a multivariable function. *Journal of Mathematical Physics*, 27(7):1834–1838, 1986.

[17] S. Kirkpatrick, C.D. Gelatt, and M.P. Vecchi. Optimization by simulated annealing. *Science*, 220:671–680, 1982.

[18] J. Pinter. Convergence properties of stochastic optimization procedures. *Math. Operat. Stat. , Ser. Optimization*, 15:405–427, 1984.

[19] G. Rappl. *Konvergenzraten von Random Search Verfahren zur globalen Optimierung.* Dissertation, HSBw München, Germany, 1984.

[20] G. Rappl. On linear convergence of a class of random search algorithms. *Zeitschrift f. angew. Math. Mech. (ZAMM)*, 69(1):37–45, 1989.

[21] Ingo Rechenberg. *Evolutionsstrategie: Optimierung technischer Systeme nach Prinzipien der biologischen Evolution.* Frommann–Holzboog Verlag, Stuttgart, 1973.

[22] F. Romeo and A. Sangiovelli-Vincentelli. A theoretical framework for simulated annealing. *Algorithmica*, 6:302–345, 1991.

[23] G. Rudolph. Parallel approaches to stochastic global optimization. In Joosen and Milgrom [15], pages 256–267.

[24] Hans-Paul Schwefel. *Numerische Optimierung von Computer-Modellen mittels der Evolutionsstrategie.* Interdisciplinary systems research; 26. Birkhäuser, Basel, 1977.

[25] Hans-Paul Schwefel. *Numerical Optimization of Computer Models.* Wiley, Chichester, 1981.

[26] F.J. Solis and R.J.-B. Wets. Minimization by random search techniques. *Math. Operations Research*, 6:19–30, 1981.

[27] H. Szu and R. Hartley. Fast simulated annealing. *Physics Letters A*. 122(3/4):157–162, 1987.

[28] H. Szu and R. Hartley. Nonconvex optimization by fast simulated annealing. *Proceedings of the IEEE*, 75(11):1538–1540, 1987.

[29] D. Vanderbilt and S.G. Louie. A Monte Carlo Simulated Annealing approach to optimization over continuous variables. *Journal of Computational Physics*. 56:259–271, 1984.

[30] L.T. Wille. Minimum energy configurations of atomic clusters: New results obtained by simulated annealing. *Chemical Physics Letters*. 133:405–410, 1987.

[31] L.T. Wille and J. Vennik. Electrostatic energy minimization by simulated annealing. *Journal of Physics A*, 18:L1113–1117, 1985. (Corrigendum, 19:1983, 1986).

[32] Z.B. Zabinsky and R.L. Smith. Pure adaptive search in global optimization. *Mathematical Programming*, 53:323–338, 1992.

The Role of Duality in the Efficient Use of a Variant of Karmarkar's Algorithm

A.Salhi[+], G.R.Lindfield[*] and L.G.Proll[+]

[+] Division of Operational Research and Information Systems,
School of Computer Studies, University of Leeds, UK

[*] Dpt. of Computer Science and Applied Mathematics,
Aston University, Birmingham B4 7ET, UK

1 Introduction

Duality is routinely used in conjunction with simplex-based algorithms for linear programming. In conjunction with interior-point methods, the same can hardly be said. Although it has been used in the design of many interior-point algorithms[1],[5],[6], its role in postoptimality analysis and structure exploitation when these algorithms are used, is not fully investigated.

The problem of postoptimality analysis is central to optimisation in general and linear programming in particular. It relies heavily on the duality concept without which linear programming would not be such a powerful decision making tool. It is concerned with the stability of the solution set of a given LP problem when perturbations occur in the input data. Here, emphasis will be on discrete changes only.

Structure is also an important aspect of linear programming. A lot of effort has gone in designing algorithms which take advantage of structure in general linear programmes but without much success [2]. These algorithms are almost exclusively simplex-based. Looking into ways of exploiting structure when interior-point methods are used is therefore an interesting challenge. The concern here will be with common structures such as block-angular and staircase.

To deal with these questions a variant of Karmarkar's algorithm which generates dual variables will be used.

The question of implementing this variant will be addressed in Section 2. In Sections 3 and 4 ways of extending it to deal with postoptimality analysis and structure respectively will be investigated. Section 5 will discuss computational results and conclusion.

2 A dual variant of Karmarkar's Algorithm

The dual variant of Karmarkar's algorithm considered in the present work is based on an algorithm that can be found in one form or another in [1],[5],[6]. It handles LP problems in standard form with bounded and non-empty feasible regions. No assumption is made about degeneracy and the algorithm has polynomial complexity. An important feature of the algorithm is the way in which improved lower bounds on the objective function value are found. The set from which these bounds can be chosen will be explicitly given. Some comparative results concerning robustness between this algorithm and a standard simplex routine on Hilbert type LP problems will be reported.

3 Postoptimality Analysis

Postoptimality analysis considers variations in the coefficients of the problem with a view to assessing how sensitive to these variations is the optimum solution which has been obtained. When simplex-based algorithms are used the process of postoptimality analysis can be conducted without difficulty. The crucial question is whether this process in the case of Karmarkar-type algorithms is of similar difficulty and can be done with similar efficiency. It can already be said that given the way in which optimality is checked in interior-point methods, postoptimality is never going to be easy and cost effective. In fact our limited investigation points to the use of standard results such as the complementary slackness conditions, which are independant of the method used to solve the LP problem, to conduct postoptimality analysis.

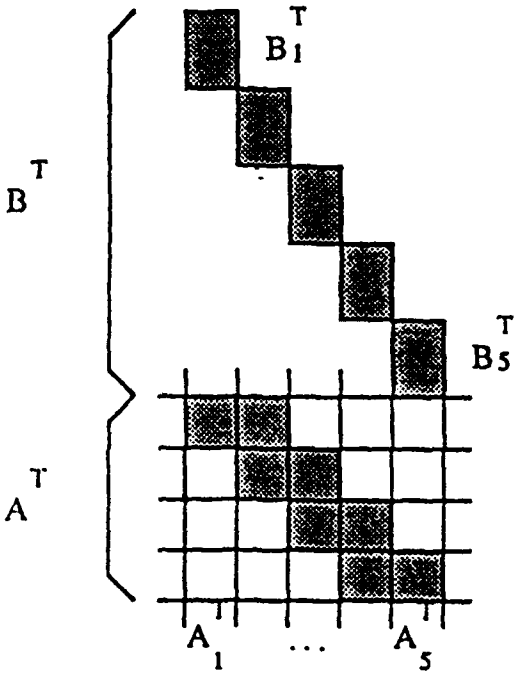
4 Structure Exploitation

In [4] an extension of the algorithm mentioned earlier to structured LP (block-angular and staircase) has been given. In that extension, structure exploitation was at the level of the computation of the dual solutions, which is the most expensive step of the algorithm. It relies on the updating algorithm for least squares of Heath [3].

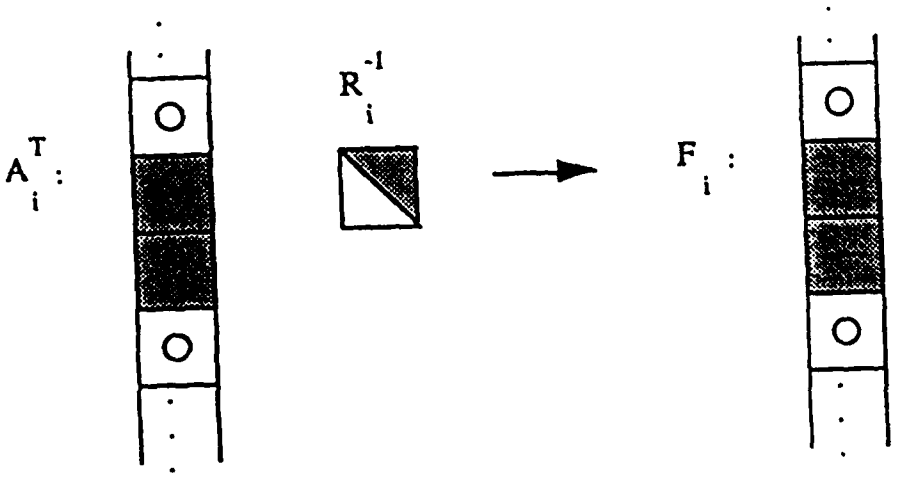
The way staircase structured problems are handled is by means of problem manipulation so that a block-angular structure is arrived at. However, the resulting block-angular problem has a linking block with many columns. The updating algorithm, which includes the solution of a square system of linear equations of the form $(I + FF^T)u = r$ where $F = (F_1 F_2 ... F_l)$ and $F_i = A_i^T R_i^{-1}$, A being the linking block, would, therefore, not be effective. However, the linking block is also structured and the structure is carried through to the system of linear equations, (see diagrams below); partitioning it to take advantage of its structure appears to be an attractive way of improving the updating step.

5 Computational Results and Conclusion

Experiments with well known difficult LP problems show that the dual algorithm considered here is robust. The extended version has been tested on block-angular problems. It leads to good speed-ups as the results show. Postoptimality analysis is made possible as dual solutions are available. However, it does not seem to be easily conducted via the interior-point method considered. Much work needs to done before any conclusion can be reached.



Partitioning of the Linking Block of the Matrix
Derived from a 5-Stage Staircase LP Problem



Structures of A_i^T and F_i.

References

[1] D.M. Gay. A variant of Karmarkar's linear programming for problems in standard form. *Mathematical Programming*, 37(1):81–90, 1987.

[2] C.W. Gear. Numerical computation: Its nature and research directions. *SIGNUM Newsletter*, 1979.

[3] M.T. Heath. Numerical methods for large sparse linear least squares problems. *SIAM J.Sci.Stat.Comp.*, 4(3):497–513, 1984.

[4] A. Salhi and G.R. Lindfield. Standard form karmarkar's algorithm for structured linear programming. In E.F. Deprettere and A.-J. van der Veen, editors, *Algorithms and Parallel VLSI Architecture*, pages 502–510. Elsevier Science Publisher, 1991.

[5] M.J. Todd and M.P. Burrell. An extension of Karmarkar's algorithm for linear programming using dual variables. *Algorithmica*, 1:409–424, 1986.

[6] Y. Ye and Kojima M. Recovering optimal dual solutions in Karmarkar's polynomial algorithm for linear programming. *Mathematical Programming*, 39(3):305–317, 1987.

MARKOV DECISION PROCESSES WITH RESTRICTED OBSERVATIONS: FINITE HORIZON MODEL

b y

Yasemin Serin and Zeynep Müge Avşar

Department of Industrial Engineering

Middle East Technical University

In this report, we develop an algorithm to compute optimal policies for Markov Decision Processes subject to the constraints that result from some observability restrictions on the process. We assume that the state of the Markov process under consideration is unobservable, but there is an observable process related to the original one. So, we want to find a decision rule depending on this observable process only. The objective is to minimize the total expected discounted cost over a finite horizon.

Restricting the policies, as explained above, results in a nonlinear programming model. The solution procedure for this nonlinear problem is a method of feasible directions (Bazaraa and Shetty(1979), Luenberger(1973)) that uses special structure of the problem. On the other hand, it is a policy iteration method that iterates between feasible policies.

1. PROBLEM DEFINITION: Consider a Markov Decision Process (MDP) $\{(X_t, A_t): t=1, ..., T\}$ where X_t is the state of the system and A_t is the action chosen in period t, t=1, ..., T. We use period t or epoch t to mean there are t periods to go until the end of the planning horizon. Let $S=\{1, 2, ..., N\}$ be the

state set and $A=\{1, 2, ..., M\}$ be the action set. The transition probability law of the MDP is $P_{ij}(a)=P\{X_{t-1}=j| X_t=i, A_t=a\}$ for all $i, j \in S$, $a \in A$. Let $P(a)$ be the transition matrix under the action $a \in A$ and $C(X_t,A_t)$ be the cost incurred at time t with expected value $c_{ia}=E(C(X_t,A_t)| X_t=i, A_t=a)$. We assume that c_{ia} for all $i \in S$, $a \in A$, and $P(a)$ for all $a \in A$ are known. A nonstationary Markovian policy can be described by $\alpha_{iat}=P(A_t=a| X_t=i)$ for all $i \in S$, $a \in A$, $t=1, ..., T$: $\alpha \in R^{NMT}$.

Let $\mathbf{S} = \{S_1, S_2, ..., S_K\}$ be a given partition of the state space. Suppose, at decision epoch t, the state of the process X_t can not be observed, but only the subset, say S_k, that X_t belongs to is known. So, a practical decision rule is defined depending on only S_k at period t, rather than X_t, and the same decision is used for every state in S_k. We define a random variable Z_t as $Z_t=k$ if and only if $X_t \in S_k$ and call Z_t **the observation variable**. The process $\{Z_t, t=1, ..., T\}$ is called **the observation process**. So, the observation variable Z_t takes values from **the observation set** O, $O=\{1, 2, ..., K\}$. A policy α is called **a policy with respect to the partition S**, if $\alpha_{iat}=\alpha_{kat}$ for all $i \in S_k$, $a \in A$, $k \in O$, $t=1,...,T$. and satisfies

$$\sum_{a=1}^{M} \alpha_{kat}=1 \text{ for all } k \in O, t=1, ..., T \qquad (1.1a)$$

$$\alpha_{kat} \geq 0 \text{ for all } k \in O, a \in A, t=1, ..., T \qquad (1.1b)$$

In the remaining part of this report, we assume that we are given a fixed partition **S** of the state space. Also, we mean completely observable MDP by an unrestricted MDP and the

MDP under observability constraints by the restricted MDP with respect to partition $\mathbb{S}$. Clearly, we must have

$$\alpha_{iat}=P(A_t=a|\ X_t=i) \text{ for } a\in A,\ i\in S_k,\ k\in O,\ t=1,\ ...,\ T \qquad (1.2)$$

$$=P(A_t=a|\ X_t\in S_k)$$

$$=P(A_t=a|\ Z_t=k\} \text{ for } a\in A,\ k\in O,\ t=1,\ ...,\ T$$

$$=\alpha_{kat}$$

The aim is to find a policy α^* with respect to partition $\mathbb{S}$ that minimizes expected total discounted cost over T-period horizon. The same problem under infinite horizon is studied in Serin(1989) and Kulkarni and Serin(1990). This problem can also be considered as a partially observable MDP problem (Monahan, (1982)). The methodology used here is different from partially observable Markov Decision Process methodology.

2. MODEL AND THE SOLUTION METHOD: The optimal policy for the unrestricted MDP problem can be found by the probabilistic dynamic programming moving backward period by period. The optimal expected total discounted cost of a policy α over a t-period planning horizon starting with an initial state i, v^*_{it}, satisfies

$$v^*_{it}=\underset{a\in A}{\text{minimum}}\left\{c_{ia}+\gamma\sum_{j=1}^{N}P_{ij}(a)\,v^*_{j(t-1)}\right\} \qquad (2.1)$$

for all $i\in S$ and t=1, ..., T, where γ is the discount factor. A constant value is assigned to v^*_{j0}'s (Hillier and Lieberman, (1974)), e.g., $v^*_{j0}=0$.

We can state the expected total discounted cost minimization problem for unrestricted MDP by the following linear programming problem :

$$\text{Minimize} \sum_{i=1}^{N} \sum_{a=1}^{M} c_{ia} \left\{ \sum_{t=1}^{T} y_{iat} \right\} \qquad (2.2a)$$

subject to

$$\sum_{a=1}^{M} y_{iaT} = \bar{p}_i \quad \text{for all } i \epsilon S \qquad (2.2b)$$

$$\sum_{a=1}^{M} \left\{ y_{iat} - \gamma \sum_{j=1}^{N} y_{ja(t+1)} P_{ji}(a) \right\} = 0 \text{ for all } i \epsilon S,\ t=1, \ldots, T-1 \qquad (2.2c)$$

$$y_{iat} \geq 0 \quad \text{for all } i \epsilon S,\ a \epsilon A,\ t=1, \ldots, T \qquad (2.2d)$$

where the decision variable y_{iat} can be interpreted as the discounted probability of being in state i and taking action a in period t, p_i is the probability of being at state i at the beginning of the planning horizon. Then, the optimal solution:

$y^*_{iat} = \gamma^{(T-t)} P_{\alpha^*}(X_t=i,\ A_t=a)$ for all $i \epsilon S$, $a \epsilon A$, $t=1, \ldots, T$ (2.3)

The optimal policy α^* is given by

$$\alpha^*_{iat} = \frac{y^*_{iat}}{\sum_{a=1}^{M} y^*_{iat}} \quad \text{for all } i \epsilon S,\ a \epsilon A,\ t=1, \ldots, T \qquad (2.4)$$

$$= P_{\alpha^*}(A_t=a \mid X_t=i)$$

and α^* satisfies (1.1a) and (1.1b).

At a basic optimal solution, y^*_{iat} can take a positive value for at most one action, which is in accordance with the implication of recursion (2.1), i.e., the optimal policy is deterministic. If it is not possible to be in state i at some period, some arbitrary action is assigned to that state (Ross, (1989)).

Now, we may define w_{it} as the discounted probability of being in state i at period t under a given policy α, i.e.,

$$w_{it} = \gamma^{(T-t)} P_{\alpha}(X_t=i) \quad \text{for all } i\epsilon S,\ t=1, ..., T \qquad (2.5)$$

$$y_{iat} = w_{it}\, \alpha_{iat} \quad \text{for all } i\epsilon S,\ a\epsilon A,\ t=1, ..., T \qquad (2.6)$$

Now, we are ready to consider the above MDP under observability constraints. Suppose that $\{Z_t: t=1, ..., T\}$ is the observation process defined over $O=\{1,..., K\}$ characterized by a partition $\mathbf{S}=\{S_1, ..., S_K\}$. If α is a policy with respect to partition $\mathbf{S}$, then the probability of taking action a at some period t is the same for all the states in the same subset. Then, observability constraints in (1.2) with respect to partition $\mathbf{S}$ are introduced by imposing

$$\alpha_{iat}=\alpha_{jat} \quad \text{for all i, j pair in the same subset}$$
$$=\alpha_{kat} \quad \text{for all } i, j\epsilon S_k$$

to the feasible policy space of unrestricted MDP problem.

$P_{ij}(\alpha, t)$ is the probability of being in state j when there are (t-1) periods to go until the end of the planning horizon, given that the system is in state i when there are t periods to go until the end of the planning horizon and when the policy α is employed,

$$P_{ij}(\alpha, t) = P_{\alpha}(X_{t-1}=j|\ X_t=i)$$
$$= \sum_{a=1}^{M} \alpha_{k(i)at} P_{ij}(a) \quad \text{for all } t=2, ..., T \text{ and } i, j\epsilon S \qquad (2.7)$$

and $c_{ia}(\alpha)$ is the expected immediate cost incurred under policy α, given that the system is in state i at the beginning of the period t

$$c_{it}(\alpha) = \sum_{a=1}^{M} \alpha_{k(i)at} c_{ia} \quad \text{for all } i\epsilon S,\ t=1, ..., T \qquad (2.8)$$

and the cost vector in period t is

$$c_t(\alpha) = (c_{1t}(\alpha), c_{2t}(\alpha), ..., c_{Nt}(\alpha))' \qquad (2.9)$$

The optimal policy α^* with respect to partition S for a MDP is given by the solution of the following problem.

Problem D:

Minimize

$$\Phi(\alpha)= p'\left(c_T(\alpha) + \sum_{t=1}^{T-1} \gamma^{(T-t)} P(\alpha, T)...P(\alpha, t+1)c_t(\alpha) \right) \quad (2.10a)$$

subject to

$$\sum_{a=1}^{M} \alpha_{kat} = 1 \quad \text{for all } k\epsilon O,\ t=1, ..., T \quad (2.10b)$$

$$\alpha_{kat} \geq 0 \quad \text{for all } k\epsilon O,\ a\epsilon A,\ t=1, ..., T \quad (2.10c)$$

Note that there is a deterministic global optimal policy to Problem D.

In order to obtain a solution to this problem, we use the method of feasible directions (Bazaraa and Shetty, (1979)). The algorithm we develop iterates between deterministic policies, using the fact that there exists a deterministic global optimal policy to Problem D. In order to guarantee improvement at each iteration from one deterministic policy to another, a descent direction is selected in such a way that the policy improvement is achieved through changes in the partial policy of only one period, although there may be other periods implying improvement, i.e., contributing the directional derivative with a negative value. Proceeding along such a direction causes improvement at a constant rate. Then, if the search procedure starts with a deterministic policy, iterations occur between deterministic policies by taking a step of size one at each iteration.

As in the case of policy iteration algorithm of Howard(1971) for unrestricted MDP, the algorithm proceeds along the steepest descent directions satisfying above conditions.

3.CONCLUSION: Using feasible descent directions that change the partial policy of only one period at an iteration may cause the algorithm to terminate after a large number of iterations. Another disadvantage of the algorithm is the risk of termination with a deterministic local optimum or saddle point in spite of the fact that there may exist a randomized local optimum or a saddle point with a lower expected cost. The reason is that, algorithm does not take randomized policies into account. Along the line between two deterministic policies of two successive iterations of the algorithm, there can not be any point satisfying necessary Kuhn-Tucker conditions, because the expected cost function decreases linearly. However, there can be randomized policies which do not lie on any such line.

We propose another algorithm for solving Problem D which allows changes in partial policy of every period in an iteration and proceeds along the steepest descent directions. Directions making changes in more than one period cause the expected cost function to be a nonlinear function of step size. Then, for minimizing the cost function along such directions, the policy improvement step must include a line search, which is the computational burden of this algorithm and may slow the algorithm in terms of the computation time required until termination. On the other hand, it may

decrease the number of iterations. In that case, randomized policies can also be encountered.

REFERENCES

Bazaraa, M.S., and **Shetty** C.M.,1979. Nonlinear Programming, John Wiley and Sons, Inc., U.S.A..

Hillier, F.S., and **Lieberman** G.J., 1974. Operations Research, Holden-Day, Inc., San Francisco.

Howard, R.A., 1971. Dynamic Probabilistic Systems : Semimarkov and Decision Processes, Academic Press, U.S.A., Inc., Vol. II.

Kulkarni, V. G., and **Serin**, Y., 1990. Optimal Implementable Policies: Average Cost and Minimax Criteria, Technical Report 90/9, Dept. of Operations Research, University of North Carolina, Chapell Hill.

Luenberger, D. G., 1973. Introduction to Linear and Nonlinear Programming. Addison-Wesley, Massachusettes.

Monahan, G.E., 1982. "A Survey of Partially Observable Markov Decision Processes: Theory, Models and Algorithms", Naval Research Logistics Quarterly, Vol.29, No. 1, pp. 1-16.

Ross, K.W., 1989. "Randomized and Past-Dependent Policies for Markov Decision Processes with Multiple Constraints", Operations Research, Vol.37, No. 3, pp. 474-477.

Serin, Y.Y., 1989. "Implementable Policies for Markov Decision Processes", Ph.D dissertation, The University of North Carolina at Chapel Hill.

ON THE CORE OF THE MINIMUM COST STEINER TREE GAMES

Darko Skorin-Kapov
Harriman School for Management and Policy
State University of New York at Stony Brook
Stony Brook, NY 11794-3775
e-mail dskorin@ccmail.sunysb.edu

Key words: Steiner Tree, Integer Programming, Cost Allocation, Game Theory

1. Introduction

Consider a connected undirected network $G = (N \cup \{O\}, E)$ with a set of nodes $N \cup \{O\}$ and a set of arcs E. The subset D, $D \subseteq N$, is the set of communities (users). A common supplier O, provides service which is required by the communities, and any community receiving the service can in turn deliver it to adjacent communities. Each community in D is required to be connected, perhaps through other communities, to a common supplier. There is a cost, $w((i,j)) = w_{ij} \geq 0$, $(i,j) \in E$, if arc (i,j) is used to deliver service. The objective is to provide service to the communities in D at a minimum cost. We will refer to the above optimization problem as the minimum cost Steiner Tree (ST) problem.

We provide in this paper a computational analysis of a game theoretic approach to a cost allocation problem arising in a minimum cost ST-problem. The cost allocation is concerned with the fair distribution of the cost of providing the service among customers. We formulate this cost allocation problem as a cost cooperative game in characteristic function form, referred to as the ST-game. In general, the ST-game generalizes several cooperative games studied in the literature which were used to analyze a variety of cost allocation problems. For example, the class of ST-games properly generalizes the class of minimum cost spanning tree games (Bird (1976), Granot and Huberman (1981, 1984)), tree games (Megiddo (1978)) and airport games (Littlechild (1974)). ST-game is equivalent to the Fixed Cost Spanning

Forest (FCSF) game which was studied by D. Granot and F. Granot (1990) for a special case when the underlying network G has a tree structure. They show that in this very special case the core of a *FCSF* game is not empty. We extend the analysis to general networks. Work in this paper is also related to Sharkey's (1990) study of the shared facility game. Therein, he defines a simple game, and shows that the core of a simple game is nonempty if and only if the optimal values of the respective objective functions of associated IP (Integer Program) and LP (Linear Program) are equal. Here, we analyze the relationship between certain IP and LP associated with the ST-game (note that ST-game is not a simple game).

It is shown that in general the core of a ST-game may be empty. Our main result provides a sufficient (and in some cases necessary) condition for the nonemptiness of the core of the ST-game. It turns out that the core is not empty if the incidence vector of an optimal ST coincides with an optimal solution to a certain linear programming problem. We also show that the reverse is not necessarily true. Further, given an optimal ST, we construct an $O(n^3)$ algorithm (where n is the number of nodes) which verifies whether the above sufficient condition is satisfied. Moreover, if the answer to the above algorithm is positive it generates a cost allocation vector in the core.

This extended abstract is organized as follows. In Section 2 we review some standard definitions and introduce some notation. In Section 3 we provide sufficient condition for the nonemptiness of the core of the ST- game. In Section 4 we present an efficient algorithm to check whether the sufficient condition for the existence of the core allocation is satisfied and in case of favorable answer we provide a point in the core of the ST-game. Finally, in Section 5 we give some concluding remarks.

2 Definitions and Preliminaries

The minimum cost ST problem can also be formulated for directed graphs. The minimum cost Directed Steiner Tree (DST) problem is defined with respect to a

directed weighted graph $G=(N\cup\{O\},E)$ with a weight function $w: E \rightarrow R^{+}$. Nameiy, given a subset of nodes $D \subseteq N$; find a directed tree $T=(N_T\cup\{O\},E_T)$ in G, rooted away from node O and whose node set contains D, such that the total edge-weight of T is minimum. It is clear that any minimum cost ST problem can be solved by considering an appropriate minimum cost DST problem, obtained by replacing each edge of the given network by two arcs of opposite directions.

In order to analyze the cost allocation problem associated with the minimum cost ST problem, we formulate this cost allocation problem as a **cooperative game.** Consider ST problem on a network $G=(N\cup\{O\},\ E)$, with a set of users $D \subseteq N$. Denote by ST_Q, for $Q \subseteq D$, the ST problem obtained from the original problem by simply replacing D by Q. Then, the pair (D,c), where $c: 2^{|D|} \rightarrow R$ is such that $c(\emptyset)=0$ and for each $Q \subseteq D$ $c(Q)$ is the minimum objective function value of ST_Q, is a game to be referred to as the ST-game. For $x \in R^{|D|}$ and $Q \subseteq D$, let $x(Q) \equiv \sum_{j \in Q} x_j$. We can interpret $x(Q)$ as the part of the total cost paid by the coalition Q. A **cost allocation** vector x in a game $(D;c)$ satisfies $x(D)=c(D)$, and the solution theory of cooperative games is concerned with the selection of a reasonable subset of cost allocation vectors.

Central to the solution theory of cooperative games is the concept of solution referred to as the core of a game. The **core** of a game $(D;c)$ consists of all vectors $x \in R^{|D|}$ such that $x(Q) \leq c(Q)$ for all $Q \subseteq D$, and $x(D)=c(D)$. Observe that the core consists of all allocation vectors x which provide no incentive for any coalition to secede.

3 The Core of the ST-Game

It was shown by D. Granot and G. Huberman (1981) that the core of the ST-game is not empty when all nodes are communities i.e $D=N$. Unfortunately, this result cannot be extended to cases when $D \neq N$. Indeed, a simple example below (shown to me by A. Tamir) demonstrates that the core of a ST-game may be empty.

Explicitly, consider the directed network $G=(N\cup\{O\},E)$ shown in Fig. 3.1 below. Therein, $N=\{1,2,3,4,5,6\}$, and $D=\{1,2,3\}$ is the set of communities. Further, we assume that $w(i,j)=1$ for all $(i,j)\in E$.

The core, $\mathbb{C}(D,c)$, of the ST-game associated with network G is given by: $\mathbb{C}(D,c)=\{x\in R^3:\quad x_1\le 2,\quad x_2\le 2,\quad x_3\le 2,\quad x_1+x_2\le 3,\quad x_1+x_3\le 3,\quad x_2+x_3\le 3,\quad x_1+x_2+x_3=5\}$

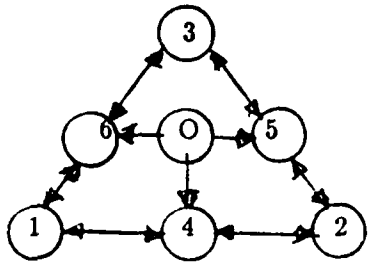


Figure 3.1 $G=(N,E)$

Now, one can easily verify that the core constraints induced by the three two-members coalitions imply that $x_1+x_2+x_3\le 4\frac{1}{2}$ for any core allocation. Thus, since any core allocation x must distribute the entire cost, i.e., $x_1+x_2+x_3=5$, we conclude that the core of the ST game associated with G , displayed in Fig. 3.1, is empty.

Below, we provide a sufficient condition for the nonemptiness of the core of the ST-game. It is based on integer programming formulation of the minimum cost DST problem used in Prodon et. al. (1985). To describe their formulation, as applied to our minimum cost DST problem, we need the following notation. Let $G=(N\cup\{O\})$ be a directed graph and D, $D\subseteq N$ the set of communities. For a directed edge $l=(i,j)$ we refer to i as the **tail** and j as a **head** of l , and for a subset of vertices S, $S\subseteq N$, we denote by $\delta(S)$ the set of all directed edges having their heads, but not their tails, in S. A subset S, $S\subseteq N$, is said to be an **admissible cut-set** of G, if, $S\cap D\neq\emptyset$ (D is the set of communities) and both subgraphs $G(S)$ and $G(N\cup\{O\}\,|\,S)$ of G induced by S and $N\cup\{O\}\,|\,S$, respectively, are connected. We denote by Λ the set of all admissible cut-sets of G. Now the DST problem can be formulated as the following integer

programming problem: $IP(D)$: $min\left\{cx : x(\delta(S)) \geq 1,\ S \in \Lambda,\ S \cap D \neq \emptyset,\ x \in \{0,1\}\right\}$.

Then, our ST-game based on the above formulation of the DST problem is the pair (D,c), where $c: 2^{|D|} \rightarrow R$ is such that $c(\emptyset) = 0$ and for each $Q \subseteq D$, $c(Q)$ is the minimum objective function value of $IP(Q)$.

Clearly, the exponential number of core constraints, coupled with the fact that $IP(Q)$ is NP-hard whenever $2 < |Q| < |N|$, makes the core computations hard. We provide some shortcuts that enable efficient computation in certain cases.

Consider the linear programming relaxation $LP(D)$ of $IP(D)$ defined as follows:

$$LP(D) : min\left\{cx : x(\delta(S)) \geq 1,\ S \in \Lambda,\ S \cap D \neq \emptyset,\ x \geq 0\right\}$$

In, view of favorable computational results obtained by Prodon et al. (1985) and Chopra et. al. (1992) with instances of the DST problem on general graphs, they were led to conjecture that the inequalities describing the feasible region of $LP(D)$), while not sufficient to describe the polyhedron for the DST problem, are nevertheless important in the sense that they often produce optimal integer extreme points. However, notwithstanding this favorable computational experience, $LP(D)$ would fail to produce optimal directed Steiner trees even for the very simple case like network G in Fig. 3.1. Indeed, consider the directed network G given in Fig. 3.1. All the edge weights in G are assumed to be 1 . Then, it is easy to check that the minimum cost DST in G, with root O and whose vertex set contains vertices O, 1, 2, 3, has a weight of 5 . However $x^* \in R_+^E$ defined as follows: $x^*_{(0,4)}=\frac{1}{2}$, $x^*_{(0,5)}=\frac{1}{2}$, $x^*_{(0,6)}=\frac{1}{2}$, $x^*_{(4,1)}=\frac{1}{2}$, $x^*_{(4,2)}=\frac{1}{2}$, $x^*_{(5,2)}=\frac{1}{2}$, $x^*_{(5,3)}=\frac{1}{2}$, $x^*_{(6,1)}=\frac{1}{2}$, $x^*_{(6,3)}=\frac{1}{2}$, and $x^*_{(i,j)}=0$ otherwise, is feasible to $LP(D)$ associated with G and has a lower objective function value of 4.5 .

Nevertheless, $LP(D)$ is still useful for the analysis of the ST-game. Indeed, we show that sufficient condition for nonemptiness of the core of the ST-game is that linear programming relaxation of $LP(D)$ has integral optimal solution.

THEOREM 3.1 If the incidence vector of a minimum cost DST in G, rooted away from O and whose vertex set contains D, is an optimal solution to $LP(D)$, then the

core of the associated ST-game is not empty. □

Next we show that the sufficient condition given in T.3.1, in general, is not necessary for the nonemptiness of the core.

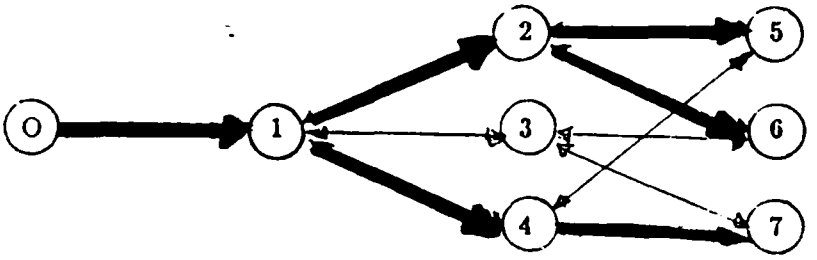


Figure 3.1 $G_1 = (N_1 \cup \{O\},\ N_1)$

Indeed, consider the network $G_1 = (N_1 \cup \{O\},\ E_1)$ in Fig. 3.2. Assume that all edge weights in G_1 are 1 and let $D_1 = \{5,6,7\}$ be the set of users. An optimal solution to $IP(D_1)$ is indicated by bold arcs and has total weight $c(D_1) = 6$. It is easy to check that vector $(x_5,x_6,x_7) = (2,2,2)$ is in the core $C(D_1,c)$ of the associated ST-game. On the other hand one can verify that $x^* \in R_+^{E_1}$ defined as follows: $x^*_{(0,1)}=1$, $x^*_{(1,2)}=\frac{1}{2}$, $x^*_{(1,3)}=\frac{1}{2}$, $x^*_{(1,4)}=\frac{1}{2}$, $x^*_{(2,5)}=\frac{1}{2}$, $x^*_{(2,6)}=\frac{1}{2}$, $x^*_{(3,6)}=\frac{1}{2}$, $x^*_{(3,7)}=\frac{1}{2}$, $x^*_{(4,5)}=\frac{1}{2}$, $x^*_{(4,7)}=\frac{1}{2}$, and $x^*_{(i,j)}=0$ otherwise, is feasible to $LP(D_1)$ associated with G_1 and has the objective function value of 5.5.

4 The Core Algorithm

Whenever we can find the optimal minimum cost ST, we can efficiently test whether the sufficient condition for the nonemptiness of the core of the ST-game, given by T.3.1 is satisfied. We construct the algorithm which is a modification of Prodon's et. al. (1985) T-guided heuristic for finding a minimum cost DST. We prove that our algorithm will terminate with a dual feasible solution, whose corresponding objective function value is equal to the weight of T if and only if the incidence vector of T is an optimal solution to $LP(D)$. In addition to that, if the algorithm delivers optimal dual solution, it also immediately generates point(s) in the core. The algorithm uses only

quadratic number of simple constraints and overall takes $O(n^3)$ time.

Let $T=(N_T \cup \{O\}, E_T)$ be an optimal DST rooted in O. Number nodes in T starting with assigning zero to root O and $\{1,2,..,|N_T|\}$ to nodes in N_T in such a way that for every $k \in N_T$, all successors of k have numbers greater then k. For $k \in N_T$, let $T_k=(N_k \cup \{p(k)\},\ E_k)$ be the subarborescence of T rooted in the unique immediate predecessor $p(k)$ of k. The following algorithm often produces point in the core $C(D,c)$.

THE CORE ALGORITHM

For $k = |N_T|,\ldots,1$, scan k as follows:

begin

while $w(p(k),k) > 0$ and there exist $S \in \Lambda$ such that
$S \cap (N_T \cup \{O\} \setminus N_k) = \emptyset$ and $w(e) > 0$, for all $e \in \delta(S)$ **do**

find minimal such set S,
$y_S = min\{w(e) : e \in \delta(S)\}$,
$w(e) = w(e) - y_S$ for all $e \in \delta(S)$,
pick up an arbitrary node $l \in D \cap S$ and update $x_l = x_l + y_S$,

end

end

THEOREM 4.1 Let $x \in R^D$ be the vector obtained by the core algorithm. Then for all $S \subseteq D$, $x(S) \leq c(S)$. □

Let λ represent the maximum fraction of total cost that can be distributed while satisfying the core constraints. That is :

$$\lambda = max\ \{x(D)\ s.t.\ x(S) \leq c(S) \text{ for all } S \subseteq D\}/c(D).$$

(Bounds on λ for some special cases of the underlying network G are presented in Sharkey (1992)).

COROLLARY 4.2 Let $x \in R^D$ be the vector obtained by the core algorithm. Then $x(D)/c(D)$ is a lower bound on value of λ. □

THEOREM 4.3 The vector $x \in R^D$ obtained by the core algorithm is in the $C(D,c)$ if and only if the incidence vector of optimal DST is optimal solution to $LP(D)$. □

5 Summary

In this paper we provided sufficient condition for the nonemptiness of the core of a ST-game. We also developed an efficient algorithm that gives us a lower bound of the maximal total cost that can be distributed while satisfying core constraints. We prove that this algorithm will generate point in the core if and only if the optimal objective function values of associated $IP(D)$ and $LP(D)$ are equal. Computational experiments with DST on general networks performed by Chopra et. al. (1992) and Prodon et. al. (1985) indirectly confirm that our algorithm will often produce core points or will offer a good approximation for a core point.

References:

C. Bird, "On Cost Allocation for a Spanning Tree: A Game Theoretic Approach," *Networks* 6 (1976), 335-350.

S. Chopra, E.R. Gorres and M.R. Rao, "Solving The Steiner Tree Problem on a Graph Using Branch and Cut," ORSA *Journal of Computing,* Vol.4, No.3, Summer 1992

D. Granot, "A Generalized Linear Production Model: A Unifying Model," *Mathematical Programming* 34 (1986), 212-223.

D. Granot and F. Granot, "A Fixed-Cost Spanning-Forest Problem," University of British Columbia, Working Paper No. 1235, March 1987, forthcoming in *Math. of OR*

D. Granot and G. Huberman, "Minimum Cost Spanning Tree Games," *Mathematical Programming* 21 (1981) 1-18.

S.C. Littlechild, "A Simple Expression for the Nucleolus in a Special Case," *International Journal on Game Theory* 3 (1974), 21-29.

N. Megiddo, "Computational Complexity and the Game Theory Approach to Cost Allocation for a Tree," *Mathematics of O.R.* 3 (1978), 189-196.

A. Prodon, M. Liebling and H. Gröflin, "Steiner's Problem on Two-Trees," RO 850315, Department of Mathematics, EPF Lausanne, Switzerland, March 1985.

W.W. Sharkey, "Cores of Games With Fixed Costs and Shared Facilities," *International Economic Review,* Vol. 30. No. 2, May 1990

W.W. Sharkey, "Cost Allocation Theory Applied to the Telecommunications Network," Bellcore , May 1992

Buffer Allocation for a Class of Nonlinear Stochastic Knapsack Problems

[Extended Abstract]

J. MacGregor Smith
Nikhil Chikhale *
email: jmsmith@umaecs.edu

November 18, 1992

ABSTRACT

In this paper we examine a class of nonlinear, stochastic knapsack problems which occur in manufacturing, facility or other network design applications. Series, merge & split topologies of series-parallel $M/M/1/K$ and $M/M/C/K$ queueing networks with an overall buffer constraint bound are examined. Bounds on the objective function are proposed and a sensitivity analysis is utilized to quantify the effects of buffer variations on network performance measures.

Keywords — Buffer Allocation, Stochastic, Nonlinear Knapsack

1 INTRODUCTION

Stochastic networks of service centers with variable service rates and finite waiting capacities (buffers) occur in many network design applications such as manufacturing facilities, communication networks and vehicular traffic systems,. One of the most challenging tasks of the network designer is to allocate buffers at each service center while keeping in mind the total capacity of the network.

2 CONSTRAINED NETWORK DESIGN (CND) PROBLEM

This section discusses the research problem, the inherent complexities of the problem, various optimisation techniques, then proposes a methodology to solve the problem.

The analysis of a queueing network is highly dependent on the topology of the nodes and arcs of the network. We can assume a Graph $G(V, A, \Gamma)$ where

V = A finite set of nodes(vertices) which represent machine centers, $i = (1, 2, \ldots, N)$

A = A finite set of arcs which may represent the material handling transfer systems.

Γ = An incidence function which regulates the flow or routing of entities within G.

Any queueing network optimisation problem can be decomposed into a set of three inter-related optimisation problems [24].

1. *Topological Network Design Problem (TND)* The topology of the network can be identified as a combination of one or all of the three classes of network topologies, vis. i)series, ii)splitting, and iii)merging. This problem deals with finding the best topology of the nodes and the arcs.

2. *Routing Network Design Problem (RND)* This deals with routing of the flow of entities along the arcs in the given topology. The research problem studied in this paper also assumes that the topology and routing of entities in the network has already been identified.

3. *Constrained Network Design Problem (CND)* The general research problem is concerned with the allocation of resources, such as the number of servers, buffers at each of the servers, assuming that the TND and RND problems have been solved. This paper deals only with allocation of buffers at each node in the network given the constraint that the sum of the number of the buffers for the entire network does not exceed a maximum limit set for the network. An extension to the simultaneous problems of routing and buffer allocation is also discussed.

2.1 Assumptions

The assumptions for the analysis are as follows

1. *The Graph $G(V, A, \Gamma)$ has been identified i.e. the type of topology, number of nodes, arcs connecting*

*Department of Industrial Engineering and Operations Research, University of Massachusetts Amherst Massachusetts 01003

the nodes and the routing of the entities within G are given.

2. *There are N nodes in the layout.*

3. *Once the service on a product is completed at the i^{th} node, it is moved to the j^{th} node with routing probability r_{ij}. If there is no room at the j^{th} node, the item is blocked.*

4. *Arrivals to the network are Poisson with mean arrival rate Λ.*

5. *Service times at the i^{th} node are exponentially distributed with the mean rate μ_i.*

6. *The first node in the network is never starved and the final node in the network is never blocked.*

2.2 Mathematical Model

The CND problem has the following basic objective function for general network topologies:

$$Maximize\ Z = \Theta(P - V) - HL$$

$$st. \quad \sum_{i=1}^{N} X_i \leq B$$

$$X_i \geq 0 \quad \text{and integer}$$

where:

Θ = *Average throughput of the topology*

P = *Average revenue/item*

V = *Average variable production cost*

H = *Average holding cost/item*

L_i = *Number of units at node i at steady state*

$L = \sum_{i=1}^{N} L_i$ = *Average total number of units in the production line at steady state*

B = *Total capacity allocated to the network*

X_i = *Capacity (buffer) allocated at each node*

C_i = *Cost of each buffer allocated to node i in the network*

$C = C_i, i = 1, \ldots, N$, *Cost of each buffer in the network, all C_i being equal.*

μ_i = *service rate of node i*

Λ_i = *arrival rate to queue i*

The CND problem mentioned above is a nonlinear stochastic knapsack problem. One of the features of the CND problem which makes the problem very challenging to solve is that, no known closed-form expression for estimating the throughputs in arbitrarily configured finite open queueing networks exists. This feature makes it very hard to control the design variables as a function of the variation in the objective function.

2.3 Proposed Methodology

Since there is no closed-form expression for estimating the objective function of a finite queueing network, the only way of finding the sub-optimal buffer allocation is to employ an iterative method, which estimates the objective function for a set of buffers at each iteration, finds the direction of optimally and changes the buffers at nodes in a manner such that the objective function is increased until a set of convergence standards are satisfied.

One of the keys to our study here is to develop performance bounds on the objective function value so that when the optimal search procedure is carried out, we can have a robust and stable technique for searching for the optimal values of the design variables.

We will first present the bound for the methodology for $M/M/1/K$ queues, where the customer is lost if blocking occurs in the topology, then the bound for a delay system and, finally, the bound for $M/M/C/K$ queues.

3 Design Methodology

3.1 Introduction

The search method employed for arriving at the sub-optimal decision variables is the Complex Method of BOX [4], where the independent variables are the buffer values at the nodes and the objective function is the earlier mentioned objective function of the CND.

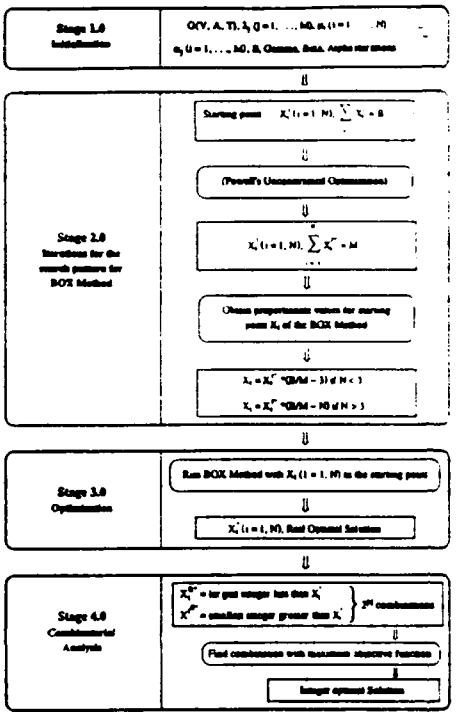
The BOX method is a derivative-free sequential search technique which conducts an iterative search for the optimum value for an objective function while there are linear or non-linear constraints on values of the independent variables.

3.2 Starting Solution

One of the features of the BOX method is that the user supplies a starting feasible point (buffer values) and this starting solution sets the search pattern for rest of the iterations. Since the pattern of allocation of buffers in the sub-optimal solution is highly dependent on the arrival rate, service rates and utilisation at the nodes and the total capacity allocated for the network, it is very important that the starting point supplied to the search method have the pattern of allocation of buffers similar to that in the final sub-optimal solution.

It is very difficult for the user to supply the right starting point since not much is not known about the dynmamic behavior of the queues and no simple knapsack heuristic starting solution was found to be appropriate. At this stage, the authors decided to use the procedure proposed in [9] to arrive at a refined final starting point for the constrained search method. This procedure, using the previous bounding methods to estimate the throughput and the objective function of the network and Powell's procedure [21] for solving unconstrained nonlinear programming problems, calculates the allocation of buffer at the nodes based on the arrival rate,

service rates of the nodes, routing of the entities and the assumption that there is no constraint on the total capacity of the network.



3.3 General Design Methodology

The CND problem is solved in four stages. A flowchart representation is given in figure 1.

Stage 1.0 *(Initialization)*

- **1.1** *Identify the Graph $G(V, A, \Gamma)$, arrival rate/s Λ_j, $j = 1, \ldots, M$, where (M= number of queues in a merging topology) , service rates μ_i, $i = 1, \ldots, N$, routing probabilities α_j, $j = 1, \ldots, K$, (K = number of queues in a splitting topology) and the total capacity B in the network .*
- **1.2** *Select the values of γ, ζ, beta and δ.*
- **1.3** *Set the limit on the number of iterations for the search method.*

Stage 2.0 *(Iterations for the search pattern for BOX method)*

- **2.1** *Select a starting point X_i^1 $i = 1, \ldots, N$ such that*
  $X_i^1 \geq 0$ and $\sum_{i=1}^{N} X_i^1 < B$ for the procedure in step 2.2.
- **2.2** *Obtain $X_i^{1\circ}$ $i = 1, \ldots, N$, the sub-optimal buffer allocation at the nodes using the procedure in [9] assuming infinite capacity for the network.*
- **2.3** *Obtain proportionate values X_i $i = 1, \ldots, N$, at the nodes such that they meet the capacity constraint set for the network.*
  Let
  $X_i^{1\circ}$ $(i = 1, \ldots, N)$ = buffers allocated to the nodes in the optimal solution in step 2.2
  $M = \sum_{i=1}^{N} X_i^{1\circ}$
  B = total capacity for the network
  X_i $i = 1, \ldots, N$ = starting solution for the BOX search method
  then
  $X_i = X_i^{1\circ} * (B/(M-3))$ if $N \leq 3$
  and $X_i = X_i^{1\circ} * (B/(M-N))$ if $N \geq 3$

Stage 3.0 *(Obtain a continuous sub-optimal solution)*

Run the BOX search method to obtain the solution X_i° $i = 1, \ldots, N$.

Stage 4.0 *(Obtain an integer sub-optimal solution) The sub-optimal solution obtained in step 3 will be, but for an exception, a non-integer solution.*

- **4.1** *To arrive at an integer solution, form 2^N combinations using the two integers closest to the value of the buffers for each node.*
  Let
  $X_i^{1\circ}$ be the largest integer less than X_i° and
  $X_i^{2\circ}$ be the lowest integer greater than X_i°
  So we have 2^N combinations of integer solutions.
- **4.2** *Of the above 2^N combinations, only those whose sum is less than or equal to B are selected. Thus there are $N * (N+1)/2$ combinations. Obtain the objective function for each of the combinations. The objective function for each combination is calculated using the expansion method.*
- **Stage 4.3** *The combination which returns the maximum value of the objective function is the sub-optimal integer solution.*

The authors conducted approximately 200 experiments with various types of topologies and their combinations and found the above described methodology to work successfully each and every time. The results of experiments conducted for series, splitting and merging topologies are described and validated in the next section.

4 Summary Of Results

In this paper, a methodology was proposed for finding the optimal buffer allocation in a constrained network

design problem. The results in the preceding sections show that this methodology is very effective in providing local optimal or sub-optimal buffer allocations to the nodes for series, merge and split topologies. A sensitivity analysis was also conducted to provide insights regarding the behavior of constrained queueing networks.

References

[1] Altiok, T., "Approximate analysis of exponential tandem queues with blocking," *European Journal of Operations Research*, 11, 1982, pages 390-398.

[2] Altiok, T. and Perros, H. G., "Open networks with blocking: split and merge configurations," *AIIE Transactions*, 1986, pages 251-261.

[3] Bell, B. C., "The effect of restricted access to intermediate product inventory in two-stage production lines," *International Journal of Production Research*, Vol. 21, No. 1, 1983, pages 75-85.

[4] Box, M. J., *Computer Journal*, 8:42(1965).

[5] Busacott, J. A., "Automatic transfer lines with buffer stocks," *International Journal of Production Research*, 5, 1967, pages 183-200.

[6] Busacott, J. A., "The role of inventory banks in flow-line production systems," *International Journal of Production Research*, 9, 4, 1971, 425-436.

[7] Conway, R., Maxwell, W., McClain J. O., "The role of work-in-process inventory in serial production lines," *Operations Research*, Vol. 36, No. 2., March-April 1988.

[8] Daskalaki. S. and Smith, J. M., "The static routing problem in open finite queueing networks," *ORSA/TIMS Miami Meeting*, Dec. 1986.

[9] Daskalaki, S. and Smith, J. M., "Optimal routing and buffer space allocation in series-parallel queueing networks," *in review*, February 89.

[10] Freeman, M. C., "The effects of breakdown and interstage storage on production line capacity," *The Journal of Industrial Engineering*, Vol. 15, No. 4, July-August 1964, pages 194-200.

[11] Gershwin, S.B., "An efficient decomposition method for the approximate evaluation of tandem queues with finite storage space and blocking," *Operations Research*, Vol. 35, Dec. 1986, pages 291-305.

[12] Hatcher, J. M., "The effect of internal storage on the production rate of a series of stages having exponential service times," *AIIE Transactions*, Vol. 1, June 1969, pages 150-156.

[13] Himmelblau M., *Applied Nonlinear programming*, McGraw-Hill Book Company, New York 1987, pages 177-180.

[14] Jafari, M., and Shanthikumar, J., G., " Determination of optimal buffer storage capacities and optimal allocation in multistage automatic transfer lines," *To appear in IIE Transactions.*

[15] Kerbache, L., " Analysis of Open Finite Queueing Networks "*Ph.D. Thesis*, I.E.O.R., University of Massachusetts, 1984.

[16] Kerbache, L., and Smith, J., M., " The generalised expansion method for open finite queueing networks," *European Journal of Operations Research*, Vol. 32, 1987, pages 448-461.

[17] Kubat, P. and Sumita, U.," Buffers and backup machines in automatic transfer line," *International Journal of Production Research*, Vol. 23, No. 6, 1985, pages 1259-1270.

[18] Labetoulle, J. and Pujjole G., "Isolation method in a network of queues," *IEEE Transactions on Software Engineering*, Vol. SE-6, No. 4, July 1980, pages 373-381.

[19] Okamura, K., and Yamashina, H.,"Analysis of the effect of buffer storage capacity in transfer line systems," *AIIE Transactions*, June 1977.

[20] Okamura, K., and Yamashina, H.,"Justification for installing buffer stock in unbalanced two stage automatic lines," *AIIE Transactions*, Vol. 11. No. 4, 1979.

[21] Powell, M. J. D., "Unconstrained Minimisation procedures without using derivatives," *Computer Journal*, 7:155(1964), 7:303(1965).

[22] Shanthikumar, J. G., "On the production capacity of automatic transfer lines with unlimited buffer space," *AIIE Transactions*, September 1983, pages 273-274.

[23] Smith, J. M., "Traypack Engineering Study: Prototype Plant Design," *Technical Report*, TR-87/017, U.S. Army Natick RD&E Center, Natick, Mass.

[24] Smith, J. M., Graves, R. J. and Kerbache, L., "QNET: An open queueing network model for material handling system analysis," *Material Flow*, 3 ,1986, pages 225-242.

[25] Van Dijk, N. and B. Lamond, "Bounds for the Call Congestion of Finite Single-server Exponential Tandem Queues." *Oper. Res. 36*, 470–477, 1988.

[26] Van Dijk, N. and J. van der Wal, "Simple Bounds and Monotonicity Results for Finite Multi-Server Exponential Tandem Queues," *Queueing Systems 4* 1–16, 1989.

CONSTRUCTION AND IMPLEMENTATION OF A CENTRAL PATH FOLLOWING ALGORITHM FOR SEMIFINITE CONVEX PROGRAMS IN MATLAB

György Sonnevend
Eötvös University, Budapest, Inst. of Math.,
Dept. of Numerical Analysis

Extended Abxtract

Interior point methods more or less close method of following the central path have been succesfully applied for the solution of large linear programs.

Generalizing the notion of the analytic center of a finite system of linear (convex, analytic) inequalities — which proved to be of central importance for the theory of interior point methods in linear (convex) programming - – we define an analytic center for convex sets K in R^n defined as feasible sets, corresponding to a smooth, p parameter family of convex, (e.g. quadratic, or linear) inequalities $1 \leq p \leq n-1$.Connections to the theory of (central solutions of) the classical moment and related operator extension problems as well as to relevant notions of affine differential and integral geometry are briefly discussed.One of the most reassuring fact is that the "maximum entropy" solutions, in the classical moment problems, see [1], can be interpreted as a special application of the general principle, used by us to define a path of nice feasible solutions leading to the set of the optimal ones, see below.

For the solution or semiinfinite linear programs (wich arise when the finite index set is replaced with a continuum on wich one is interested to solve say an optimal approximation problem or a moment problem the commonly used methods were based on (adaptive) discretization of A and solution of the arising finite but large linear programs. In this way the smoothness, analiticity of the data functions (on A) has *not* been used i.e. exploited at all, and the dimension grow drastically when accuracy requirements are increased. In [1], [2], [3] we outlined methods using analytic centers and central path for solving semiinfinite convex programs.

In order to explain our approach we remind, that an optimization problem is easily reduced to a one parameter family of feasibility problems. Therefore we think, that a basic problem of numerical convex analysis is to find a nice solution concept for (important classes of) feasibility problems with feasible sets, say of the following type:

$$P_p^{A,B} := \{x = L\xi | f(\alpha, \xi, D_1\xi, p) \geq 0, \alpha \in A, e(\beta, \xi, D_2\xi, p) = 0, \beta \in B\}, \qquad (1.1)$$

where L is a linear operator, ξ is the state of the underlying system, D_1 and D_2 are linear, constant differential operators (applicable to the functions $\xi(\cdot)$), $f(\cdot)$ being concave quadratic in $(\xi, D_1\xi)$, $e(\cdot)$ being linear in $(\xi, D_2\xi)$, p is a parameter, A and B are "index sets", the elements of which have often the interpretation as points in space and (or) time. "Nice" means, that this solution must be a low complexity function of the "data", i.e. the parameter p defining the system (of inequalities and equalities), which can easy be updated, when this system, i.e. its parameters are changed (usually by a one parameter *homotopy*). Section 2 is devoted to such a concept of nice central solution.

The second point is here that reasonably (i.e. not too) complex feasibility problems are those, in which the elementary inequalities and equalities are simple, i.e. the first given by the positivity of a linear or quadratic function on the unknowns, while the equalities being linear in x, moreover the dependence of an individual inequality (equality) on its defining parameter is also "simple"; this will be qualified further below, e.g. in the case $D_1 = D_2 = 0$ mainly by requiring, that certain integrals of the arising, algebraically simple functions over the given

sets, A, B can be easily computed by simple quadrature (i.e. "cubature",...) formulae (within appropriate accuracy). By requiring (in fact often : "exploiting") this, we are able to avoid the blowing up of the dimension of the linear or "quadratic" programming problems arising by brute force discretization of the parameter space A). Similarly we have to assume, that the "structure" of the equality constraints imposed on the unknowns is also "simple". We shall see, that in this approach the basic problems belong to the realm of classical analysis, algebraically simple analytic functions and their integrals and approximations — say by rational functions, or by other simple, constructive classes of rather smooth functions — playing an important role: the effectivity of the proposed method depends on how quickly we are able to follow, i.e. continue by extrapolating (i.e. predicting) the homotopy path of "nice", interior solutions leading to an optimal solution.

It turns out, that the latter problem is closely connected to an other problem: how to find "nice", relatively tight two sided *ellipsoidal approximations* (around the previously defined "nice", central solutions) for the corresponding feasible sets, in fact nicety of these centres should be defined as to include the existence of low complexity algorithms for constructing and updating these ellipsoids. There are several reasons for imposing these requirements. First of all: the existence of such approximations turns out to be responsible for the effectivity of the corrector phase — via Newton's method — of the (homotopy) path following, predictor-corrector method,see [5].

2. Basic properties of analytic centers

The (analytic) center $x(f_p^{A,B})$ of the convex inequality system (1.1) — with a bounded feasible set $P_p^{A,B}$ having a nonempty interior in R^n — is defined as the (in general) unique solution of the supremum problem

$$\sup_{x \in P_p^{A,B}} \Phi(x), \Phi(x) = \sup\{\int_A \log f(\alpha, \xi, D_1\xi, p) d\alpha | e(\beta, \xi, D_2\xi, p) = 0, x = L\xi, \forall \beta \in B\}, \quad (2.1)$$

where $d\alpha$ is a measure, which is independent on ξ, but may depend on the set $\{f^A\}$; we assume here — just for simplicity — ,that $d\alpha$ depends only on A. Notice, that (2.1) is a classical Euler-Lagrange type variational problem, which in general has a unique solution, depending analytically on the parameter p. In this section we discard the dependence of f on the parameter p and first restrict the attention to the case, where B is the empty set. In general, e.g. if A is a finite set and if all $f(\alpha, \cdot)$ are linear, or if all $f(\alpha, \cdot)$ are concave and (at most) quadratic at least one being negative definite, the function

$$\Phi(x) = \int \log f(\alpha, x) d\alpha \quad (2.2)$$

is strongly concave over P^A. To assure the existence of the integrals in (2.1) and those appearing later below it would be enough to assume that $f(\cdot, x)$ and its derivatives (up to order two) are continous and uniformly (in $x \in P^A$) bounded over A. More important is that in the proposed methodology (using analytic homotopies along centers) we need (at least we should like to get) a high degree of smoothness and algebraic simplicity, therefore a nonsmooth constraint of the type, say

$$\max_i g_i(y) \leq 1, \; g_i(y) := \sum_{1 \leq j \leq k} | f_{ij}(y) |, i = 1, \ldots, n \quad (2.3)$$

will be replaced by a set of smooth contraints

$$\sum_j \eta_{ij} < 1, \quad -\eta_{ij} \leq f_{ij}(y) \leq \eta_j, \quad 1 \leq j \leq k, \quad i = 1, \ldots, n$$

i. e. we set $x = (\eta_1, \ldots, \eta_k, y)$. For simplicity we shall assume first that in (2.1) $f(\cdot) = f(\alpha, x), e \equiv 0, \ L = id, \ D_1 \equiv 0$. The following invariance properties of this "solution" concept are important:

(1) *affine invariance*: if we replace $f(\alpha, \cdot)$ by $\hat{f}(\alpha, \cdot), \hat{f}(\alpha, y) := f(\alpha, Ty + t)$, where $t \in R^n, T : R^n \to R^n, \det T \neq 0$ then

$$t + Tx(\hat{f}^A) = x(f^A) \tag{2.4},$$

(2) invariance under *scaling*: if we replace $f(\alpha, \cdot)$ by $\check{f}(\alpha, \cdot) := k(\alpha) f(\alpha, \cdot)$, for an arbitrary function $k(\alpha)$ then

$$x(\check{f}^A) = x(f^A) \tag{2.5}.$$

The proof of (1) is easy obtained from the characterization of the optimal solutions for (2.1)

$$\int_A \frac{\nabla_x f(\alpha, x)}{f(\alpha, x)} d\alpha = 0 \tag{2.6}$$

The proof of (2) is immediate from the "additivity" of the log function.

A further, rather useful property of this solution concept is that it often allows to find good *ellipsoidal inner or (and) outher approximations for the set* P^A. The idea is simple: consider the function

$$\Psi^A(x) := \exp(\phi(x)) \tag{2.7}$$

and the set —where $H = D_2 \Psi^A(x(f^A))$ is the Hessian of Ψ^A at $x(f^A)$ —

$$E(f^A) := \{z \mid -\frac{1}{2} z^* H z \leq \Psi^A(x(f^A))\}. \tag{2.8}$$

It is natural to expect that $E = E(f^A)$ is a good ellipsoidal approximation of P^A (wich of course, shares the invariance properties (1) and (2) of $x(f^A)$.

Theorem 1

Suppose that A is a finite set of cardinality m and that the functions $f(\alpha, \cdot), \ \alpha \in A$ are concave and quadratic, then

$$x(f^A) + \frac{1}{2\sqrt{m}} E(f^A) \leq P^A \leq x(f^A) + \sqrt{2m} E(f^A) \tag{2.9}$$

This is proved in [14]. The suprising point is here the independence of the quality of this approximation on the specific data, i. e. the form of the functions $f_\alpha, \ \alpha \in A$.

An important property of the function Ψ is that it is *concave* on P^A,if

$$\int_A d\alpha = 1$$

, a condition (normalization) which we shall assume below (obviously without loss of generality).This can be proved by computing the Hessian of Ψ

$$H = \int_A (\frac{D_2 f(\alpha, x)}{f(\alpha, x)} - a(\alpha)a^*(\alpha))d\alpha + \int_A a(\alpha)d\alpha (\int_A a(\alpha)d\alpha)^*$$

—where $a(\alpha) = \frac{\nabla_x f(\alpha, x)}{f(\alpha, x)}$— using the inequality

$$(\int_A d\alpha)^2 \int_A a(\alpha)a(\alpha)^* d\alpha \geq \int_A a(\alpha)d\alpha (\int_A a(\alpha)d\alpha)^*,$$

which is obtained from the Cauchy-Schwartz inequality (multiplying—from both sides—with a vector).

Many convex feasibility problems are written in "dual" form, known as a finite ,or restricted moment problem: given $c^T = \{c(t) \mid t \in T\}$, the "set of feasible solutions" is formed by the nonnegative mass distributions (densities over a set S

$$P_K(c^T) = \{\mu \mid c(t) = \int_S K(t, s)d\mu(s), \ t \in T \ d\mu \geq 0\}. \tag{2.10}$$

The central feasible solution is defined as that element (if it exists and is unique) which solves the supremeum problem

$$\sup\{\int_S \log \mu'(s)ds \mid \mu \in intP^K(c(\cdot))\} \tag{2.11}$$

here *int* stands (intuitively) for "interior of" and means (precisely) that only those mass distributions are regarded which have a log integrable density.

If A is a finite set, $A = \{1, \ldots, m\}$ and $f(\alpha, \cdot)$ are linear in x, say,

$$f(\alpha, x) = b_\alpha - a_\alpha^T x, \ \alpha = 1, \ldots, m$$

then introducing

$$\mu_i := b_i - a_i^T x, \ i = 1, \ldots, m$$

we can find vectors $k_1, \ldots, k_{m-n} \in R^m$ and scalars $c_1, \ldots, c_{m-n}$ such that

$$P^K(c^N) = \{\mu \mid < k_j, \mu >= c_j, \ \mu \in R_+^m\} \tag{2.12}$$

is identical with the set of vektors $\{f^m(x) \mid x \in P(f^A)\}$ (note that $\mid T \mid = N = m - n$).Obviously $f^m(x(f^A))$ yields then the solution of problem (2.11).

A further strong motivation for the importance of the solution concept (2.11)—thus for (2.1)—is that for some rather interesting special cases (of the kernel function $K(\cdot, \cdot)$), e. g. in the class of the Nevanlinna Pick type moment problems, the solution (2.11) — known as the maximum entropy solution — can be *exactly*(!) computed in a very simple way in $O(N^2)$ (in fact even in $O(N \log N)$ operations and recursively in N). In fact this classical example is at the root of recent more general results about the role (existence, applications) of central or "maximum entropy" solutions to the basic H^∞ optimization problems, see [1],[6] for further references.

An important application of centers is for solving optimization problems of the type

$$\inf\{f_0(x) \mid x \in P^A\} =: f_0^* =? \tag{2.13}$$

Using the observation that $f_0(x(\lambda)) \searrow f_0^*$ for $\lambda \searrow f_0^*$ where $x(\lambda)$ is the center of the extended system

$$\sup\{\log(\lambda - f_0(x)) + \int_A \log f(\alpha, x)d\alpha \mid x \in P^A, \lambda \geq f_0(x)\}, \tag{2.14}$$

the idea is to follow the homotopy path $x(\cdot)$ from λ_0 to f_0^* by a predictor-corrector method, where in the corrector steps Newton's method is applied to the system

$$\frac{\nabla f_0(x)}{r} + \int_A \frac{\nabla f(\alpha, x)}{f(\alpha, x)}d\alpha = 0, \quad r > 0. \tag{2.15}$$

Here we introduced a new parametrization of the central path by $r = \lambda - f_0(x(\lambda))$ such that $r = 0$ corresponds to the optimum.

3. Definition of an analytic center for convex sets defined as the intersections of a k parameter family of halfspaces

Let K be a compact, convex set in R^n, $cgr(K)$ its center of gravity is the solution $\bar{x} = x(K)$ of the following opyimization problem:

$$\sup_{x \in K} \Phi(x), \Phi(x) := \sup_{\mu}\{\int_K \log \mu(z)dz | x = \int_K z\mu(z)dz, \int \mu(z)dz = 1, \mu \geq 0\}, \tag{3.1}$$

where — for simplicity — it is assumed, that the sup is extended over all mass distributions, which have a log-integrable density. In fact the solution of the inner optimization problem iseasily obtained (notie, that (3.1) is a moment problem like (2.10)–(2.11)): for given x the optimal density has the form $\mu(z) = (\sum_{i=1}^n \alpha_i z_i + \alpha_0)^{-1}$, where the Lagrange multipliers $\alpha_i = \alpha_i(x)$, $i = 1, \ldots, n$ are uniquely determined through the optimality (and positivity) conditions (see below) — by the strong convexity of the problem —

$$\Phi^K(x) = \int_K \log(\alpha^T(x)z)dz, \quad x = \int_K z(\alpha^T(x)z)^{-1}dz, \tag{3.2}$$

where we used the notations $z \mapsto (1, z_1, \ldots, z_n)$, $\alpha^T z = \sum_{i=1}^n \alpha_i x_i + \alpha_0$. Without loss of generality and following our earlier normalization we can assume, that $vol(K) = 1$ and $cgr(K) = 0$, this implies, that $\alpha_i = 0$, $i = 1, \ldots, n$, $\alpha_0 = 1$ and $\Phi(0) = 1$.

Pursuing the analogy with the construction (2.7)–(2.8) we are led to consider the ellipsoid

$$E := \{z | -\frac{1}{2} < D^2\Phi(0)z, z > \leq 1\}. \tag{3.3}$$

This ellipsoid is the complete analogon of (2.8),there the total mass is m (the number of summands in the analogon of the potential function (2.2)).

Theorem 2. *For an arbitrary convex, compact domain K in R^n the above ellipsoid — centered at the origine, the centre of gravity of K — provides an (optimal order) two sided approximation of K:*

$$\frac{1}{\sqrt{2}}E \subseteq K \subseteq \text{const}\, nE \tag{3.4}$$

ProofSee [3],where also the case of a oneparameter family of linear inequalities is dealt with in more detail.

Now we look to the dual of the above approximation (construction).

Definition. Let P be a convex compact set, its analytic center, $c(P)$ is defined as that (uniquely determined) point, which becomes the center of gravity of the dual set P^a, when it is taken as the center of duality:

$$c(P) = cgr(P^d_{c(P)}). \tag{3.5}$$

(Here and below we indicate — by a lower index — the point, which is taken as the center of duality, unless it is the origine.)

Lemma. *The above point $c(P)$ is uniquelly determined, since it is the solution of a (strongly) convex minimization problem: it minimizes the volumen of the dual set P^d_x*

$$\arg\inf_x vol(P^d_x) = c(P), \;\; vol(P^d_x) = \int\limits_{\|\phi\|=1} \frac{d\phi}{(m(\phi,P) - \phi^T x)^n} \tag{3.6}$$

Proof. Noting that $P^d_x = \{\phi | m(\phi,P) - \phi^T x \le 1\}$, where $m(\phi,P) = \sup\{\phi^T x | x \in P\}$, the validity of the second formula being immediate differenciating with respect to x we get the condition

$$0 = \int\limits_{\|\phi=1\|} \frac{\phi d\phi}{(m(\phi,P) - \phi^T x)^{n+1}} = \text{const} \int\limits_{P^d_x} \phi d\phi, \tag{3.7}$$

i.e. that x is the center of gravity of P^d_x. The unicity of the fixpoint of (3.5) follows from the strong convexity of the function $\int (m(\phi,P) - \phi^T x)^{-1} d\phi$ in x over P, note that instead of the Lebesque measure we could take here any measure $d\phi$, which has at least n positive weights in linearly independent "points" ϕ. We were led to the point (3.5) by analogy with (2.11)–(2.12), having in mind this, latter case (of a discrete measure), where the potential function is proportional to the volume of an ellipsoid containing the dual polyhedron, see the references to earlier papers of the author in [2] and below. If we describe the set P by its dual with respect to the point $c(P)$

$$P = \{x | m(\phi,P) \ge \phi^T x, \;\; \forall \phi \in P^d_{c(P)}\}, \tag{3.8}$$

then $c(P)$ yields the maximum of the potential function:

$$\hat{\Phi}(z) = \int\limits_{P^d} \log(m(\phi,P) - \phi^T x) d\phi;$$

in fact the above lemma shows, that $c(P)$ is the unique fixed point of the map $x \to z(x)$

$$z(x) = \arg\max_z \hat{\Phi}_x(z), \;\; \hat{\Phi}^P_x(z) = \int\limits_{P^d_x} \log(m(\phi,P) - \phi^T z) d\phi$$

indeed the equations

$$0 = \int\limits_{P^d_x} \frac{\phi d\phi}{m(\phi,P) - \phi^T x}; \;\; \int\limits_{P^d_x} \phi d\phi = 0$$

are equivalent (by the zero, resp. first order) "homogenity" of the integrands). Moreover the second derivative of this "optimal" potential function, which is expected to yield the ellipsoidal approximation for this inequality system according to the construction (2,7)-(2.8), is just the matrix M, i.e. the inverse of the similar (but dual) one associated to $c(P)$ as to a center of gravity. This observation (showing the primal-dual coherence of our constructions) is a generalization of an analogous connection observed earlier for finite systems of linear or convex (quadratic) inequalities.

$$D^2\hat{\Phi}_x^P(x) = [D^2\Phi^{P^d}(x)]^{-1} \text{ for } x = c(P) = cgr(P^d)$$

In the latter (finite) case the connection (equality) between the volume of an ellipsoid E_x^d containing the dual polyhedron K_x^d and the value of the potential function $\Phi(x)$ has been known earlier.

3.On the chosen implementation

Here we present experiences with a class of methods, whose distinctive features of implementation can be summarized as follows

1.)Path following using the differential equation of the central and parallel paths (particularly its simplest first order implementation: the affine scaling direction)

2.) Exploiting the analyticity of the elementary bounds and their dependence on the semiinfinite parameter α by using high order quadrature (and path extrapolation).

3.) Dynamic, adaptiv selection of the approximate finite set of nodes to be used for checking feasibility of the extrapolated points, this is used for the selecting the stepsizes. These nodes are provided by the use of adaptive quadrature algoritms, see e.g. [5], to select (i.e. concentrate asymptotically) the integration nodes in the subdomain, where the constraints are "active".

4.) Regulation of the step size either with adopting a continuous recentering strategy or (in the first order case) by advancing with a constant portion of the distance to the boundary (along the extrapolation line).

It should be noted, that for seminfinite problems especially over a higher dimensional set A, the computation of

$$\max_{\alpha \in A} f(\alpha, x) = d(x)$$

which would be needed for an exact "estimation" of the "feasibility" (distance to the boundary) and which — by the way — is also needed to accomplish a usual pivot step in the simplex method may be rather difficult.

Since in α this function may be neither concave nor convex, the computation of the above value $d(x)$ is to be avoided. We can solve circumrent this problem in two ways; either by adopting a close following of the central path, measuring the "distance" from it by a specific quantity (which comes out from the well known analysis of the convergence domain of Newton's method for computing the centers) for monitoring the stepsize selections, or we can use an adaptive discretization of the set A, which is automatically generated by the adaptive quadrature method (by adding all the dyadic in $A = [a, b]$ nodes, which have been generated for the various components of the functions to be integrated. We did not try to implement a primal-dual procedure, since the dual variable here is a function.

In the linear case, when $f(\alpha, x) = b(\alpha) - a^T(\alpha)x$ we have the dual problem

$$\max\{-\int b^T(\alpha)d\mu(\alpha) | c_i = \int a_i(\alpha)d\mu(\alpha), \quad i = 1, \ldots, n, \quad d\mu(\cdot) \geq 0\}$$

known as a moment problem, and along the central path $x = x(r),\ r > 0$ we generate feasible solutions $\mu(\cdot)$ of the form

$$d\mu(\alpha) = \frac{r}{b(\alpha) - a^T(\alpha)x} d\alpha$$

which aproach δ-type (atomic) measures, as we approach to a (nondegenerate) optimum. The nodes generated by the adaptive quadrature method will be concentrate more and more around these points.

The problem of the optimal selection of the underlying measure $d\alpha$ A will be discussed in connection to affine invariance. We shall present test results also for the case when all constraints $f(\alpha, x)$ are quadratic in x (arising from optional control problems with "pointwise" state and control bounds). In the MATLAB code we tried to use as much parallelity as possible (in evaluating the functions, integrals, directions as vectors over the current index set, the adaptive quadrature algorithm is hard to be integrated in such a "parallel" enviroment, and for moderate accuracy requirements can be replaced by a fixed composite Gaussian quadrature.

References

[1] G. Sonnevend, Applications of Analytic Centers, NATO ASI ser., F vol. 70, "Numerical Linear Algebra and Digital Signal Processing" (P. van Dooren, and G.Golub eds.), Reidel 1988, 8p,

[2] G. Sonnevend, Application of analytic centers to feedback design for systems with uncertainties, in "Control on Uncertain Systems", eds. D. Hinrichsen, B. Martenson, Birkhäuser, 1989, 13p.,

[3] G. Sonnevend, Construction of ellipsoidal observers in linear games with bounded controls and measurement noise, in Lect. Notes in Contr. and Inf. Sci., vol 143 (1989), 413-423,in extended form this is the DFG report Nr 170/1989, Inst, fúer Angewandte Mathematik, Univ,. Würzburg,

[4] G. Sonnevend, J. Stoer, Global ellipsoidal approximations and homotopy methods for smooth, convex, analytic programs, in Applied Math. and Appl., 21(1980) pp. 139-165.

[5] G. Sonnevend, Sequential algorithms of optimal order global error for the uniform recovery of functions with monotone (r-1)-th derivatives, Analysis Mathematica, 10(1984) pp. 311-335.

[6] G. Sonnevend, J. Stoer, G. Zhao, On the complexity of following the central path by linear extrapolation in linear programs, Mathematical Programming, 1991,series B., 31p.

[7] G. Sonnevend, Constructing feedback control in differential games by the use of central trajectories; DFG report,Nr. 385/1992, Inst, fúer Angewandte Mathematik, Univ,. Würzburg, to appear in ZAMM.

Bounding network reliability via surface duality (Extended Abstract)

Heidi J. Strayer and Charles J. Colbourn
Department of Computer Science
University of Waterloo
Waterloo, Ontario, CANADA

Abstract

A probabilistic network is a network in which the vertices are assumed to be perfectly reliable and the edges have independent operational probabilities. The k-terminal reliability of a probabilistic network is the probability that all nodes are connected in a selected set of k nodes. We investigate the effectiveness of bounding all- and two- terminal reliability using surface dualization techniques. Dualization heuristics are discussed, and some computational results are given.

1 Introduction

A *network* is a probabilistic graph. In our network model all the nodes are perfect (i.e. have operational probability 1) while all edges have a fixed operation probability in the range $[0, 1]$. We assume that the operational state of each edge is independent of the operational states of all other edges. A subset of two or more nodes is selected to be the *terminal set* of the network. It is the connectedness of these terminals that reliability measures.

All-terminal reliability is the probability that the network is connected at any instant of time. *Two-terminal reliability* is the probability that two terminal nodes s and t are always connected. Computing all- or two- terminal reliability is #P-complete even for planar graphs [18]. Thus many polynomial time approximation techniques have been developed. These techniques can be divided into two categories: Monte Carlo methods and bounding methods. Monte Carlo methods result in an approximation with confidence intervals while bounding methods produce an absolute lower or upper bound.

Bounding methods are of two main types: iterative methods and static methods. Static methods produce a single bounding value, with no obvious way of improving the produced bound. Iterative methods, however, allow for incremental improvement of produced bounds. An important class of iterative methods are those that generate the most probable network states. It has been shown that static methods based on subgraph extraction and network transformation can greatly increase the efficiency of both Monte Carlo and most probable states methods [7, 9]. Hence improvements on static methods are desirable, both to obtain better bounds and to accelerate iterative techniques.

There are several existing all- and two-terminal static lower bounds. For all-terminal bounds of networks with equal edge probabilities there are Kruskal-Katona [14] and Ball-Provan [2]. Existing two-terminal static lower bounds include Kruskal-Katona [14], Chari-Provan [4], and Series-Parallel [1]. Kruskal-Katona, Ball-Provan and Chari-Provan can only be applied to networks in which all the edge probabilities are equal. Computational results support the observation that existing upper bounds are tighter than existing lower bounds. Hence better lower bounds are very desirable.

For planar graphs there is a one-to-one mapping between cutsets of a graph (primal) and the cyclic subgraphs of its dual [19]. This translates to a linear relationship between two-terminal reliability of a graph and the two-terminal reliability of its $s-t$ dual [22]. Our strategy is to generalize these equalities that hold for planar graphs to inequalities which hold for non-planar graphs and their surface duals.

2 Definitions

A *k-terminal network* $G = (V, E, \rho, T)$ consists of a set of nodes V, a set of undirected edges E, a set of edge operational probabilities ρ, and a set of k terminals T. A *two-terminal network* has two terminals s and t. For all-terminal networks we typically omit the terminal set specification. We refer to $G = (V, E)$ as the *underlying graph* of the network. The value $p_{i,j} \in \rho$ represents the probability that edge $\{i, j\}$ is operational at any instant of time. It is assumed that the edge operational probabilities are independent. Then $q_{i,j} = 1 - p_{i,j}$ represents the probability that edge $\{i, j\}$ is non-operational at any instant of time, and $\overline{\rho} = \{q_{i,j} : \{i, j\} \in E\}$. We assume that the network is simple and 2-edge connected, since reliability of a 1-edge connected multigraph can be linearly reduced to the calculation of the reliability of at most n simple 2-connected subgraphs. In this paper the number of nodes in a network is denoted by n and the number of edges by m.

In a *directed k-terminal network* $\vec{G} = (V, E, \rho, T)$, E contains directed edges (arcs). For directed networks one terminal is denoted as the source s while all remaining terminals are

destination terminals. A directed all-terminal network is specified as $\vec{G} = (V, E, \rho, (s))$ and a directed two-terminal network as $\vec{G} = (V, E, \rho, (s, t))$.

A *pathset* is a subgraph which connects all terminals (or in a directed network, connects the source to all destination terminals). A *minimal pathset* is a pathset which does not properly contain any other pathset. Hence an $s - t$ *minimal pathset* is a path from s to t and an *all-terminal minimal pathset* is a spanning tree. A *cutset* is a subset of edges that separates one or more terminals from each other (or in the case of a directed network, separates the source terminal from at least one of the destination terminals). A *minimal cutset* does not properly contain another cutset.

The sets of all pathsets, cutsets and complements of pathsets of size i are denoted by ν_i, κ_i and χ_i respectively, with $\nu = \cup\nu_i$, $\kappa = \cup\kappa_i$ and $\chi = \cup\chi_i$. The numbers of pathsets, cutsets, and complements of pathsets containing i edges are denoted by N_i, C_i, and F_i respectively. It is easily verified that:

$$N_i = \binom{m}{m-i} - C_{m-i}$$

$$N_i = F_{m-i}$$

$$F_i = \binom{m}{i} - C_i$$

If all of the edge probabilities in the network are equal to p, then $Rel[G]$ can be expressed in terms of the N_i's, C_i's or F_i's as follows:

$$Rel[G] = \sum_{i=0}^{m} N_i p^i q^{m-i} = \sum_{i=0}^{m} F_i p^{m-i} q^i = 1 - \sum_{i=0}^{m} C_i q^i p^{m-i}$$

Sometimes to emphasize the fact that we are interested in all- or two-terminal reliability we use $Rel_A[G]$ and $Rel_2[G]$ in place of $Rel[G]$.

The following topological definitions are needed (see Gross and Tucker [10]).

An *imbedding* of a network on a surface S is the drawing of the network on the surface such that no edges of the network cross. A *2-cell imbedding* is an imbedding in which all the regions are open disks or cells. All imbeddings in this paper are assumed to be 2-cell imbeddings. An *orientable 2-cell imbedding* is a 2-cell imbedding on an orientable surface. Spheres and planes (which are just spheres with a point removed) are orientable surfaces of *genus* 0. For $i \geq 1$, an

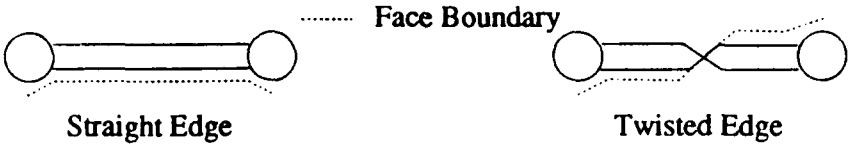


Figure 1: Face Tracing Along Straight and Twisted Edges

orientable surface of genus i can be represented by a string of i adjacent tori, or equivalently as a sphere with i loops called *handles* added to it. A *nonorientable 2-cell imbedding* is a 2-cell imbedding on a nonorientable surface. Spheres, even though they are orientable, are also taken to be nonorientable surfaces of *crosscap* 0. For $i \geq 1$, a nonorientable surface of crosscap i can be represented by a sphere with i holes, each hole closed by attaching to it a Mobius band (created by taking a strip, twisting it once, and attaching its ends together) at its boundary. Orientable imbeddings contain only *straight* edges while nonorientable imbeddings contain both straight and *twisted* edges. (See Figure 1).

Both nonorientable and orientable imbeddings can be *locally oriented* in that a clockwise or counterclockwise direction can be associated with each node of the graph. Straight edges having like oriented endpoints and twisted edges having opposing oriented endpoints are *type-0* edges. Similarly, straight edges having opposing oriented endpoints and twisted edges having like oriented endpoints are *type-1* edges. In this paper we will assume that every vertex is locally oriented in a counterclockwise direction in all imbeddings. Thus straight edges will always be type-0 and twisted edges will always be type-1.

A *surface (topological) dual* G_I^D of a graph G with respect to an imbedding I is defined in the same way as a planar dual. Each region of I is a vertex in G_I^D and edge e of the graph is added between nodes i and j in G_I^D if e is common to the boundaries of regions i and j in G. The surface dual of an all-terminal network is the surface dual of its underlying graph with edge operational probabilities of the dual is $\overline{p}$.

A generalization of an $s-t$ dual [22] $G_{I_{st}}^D$ can be defined for any 2-connected simple two-terminal network $G = (V, E, \rho, \{s,t\})$ as follows. Find some circuit containing both s and t, $C = (s, v_2, v_3, \ldots v_{i-1}, t, v_{i+1} \ldots v_k)$ (such a circuit must exist since G is 2-connected). Add edge $e_{st} = \{s,t\}$ to G. Adding e_{st} to C creates two circuits $C_1 = (s, v_2, v_3, \ldots, v_{i-1}, t)$ and $C_2 = (t, v_{i+1}, \ldots, v_k, s)$. One can then find an imbedding for the underlying graph of network $G \cup e_{st}$ such that these two cycles form region boundaries [23]. The edge probabilities in $G_{I_{st}}^D$ are $\overline{p}$ and the terminals s' and t' of $G_{I_{st}}^D$ are the dual nodes corresponding to the regions whose boundaries are C_1 and C_2 respectively. Thus $G_{I_{st}}^D = (V^D, E^D, \overline{p}, \{s', t'\})$.

For a directed network $\vec{G}$ we assign directions to the dual arcs as follows: if arc e is directed

clockwise around region $R1$ and thus counterclockwise around region $R2$ then in the dual arc e^D is directed from node $R1$ to node $R2$. Furthermore for all arcs $e^D = (i,j) \in E^D, i \neq j$, add arc $e'^D = (j,i)$ to E^D with $p_{e'} = 1$.

If $\vec{G}$ is a two-terminal directed network, we direct e_{st} from s to t and ensure that e_{st} is directed counterclockwise around C_1 and clockwise around C_2. We also delete all arcs directed into s' and all arcs out of t'. The $s-t$ dual of a directed network $\vec{G}$ with respect to some I_{st} imbedding is denoted $\vec{G}^D_{I_{st}} = (V^D, E^{D+}, \overrightarrow{\rho}^+, (s',t'))$.

One algebraic method of representing a 2-cell imbedding is a *rotation system*. Given a 2-cell imbedding I, the rotation system representing I consists of a *rotational vector* r_i for each vertex i. Each r_i contains entries for edges incident to vertex i in the order they are encountered when making a circulation around i consistent with its local orientation (which in this paper is always counterclockwise). If an edge is of type-1 we add a superscript 1 to it when including it in the rotation system. There is a 1-1 mapping between rotation systems and locally oriented graph imbeddings (up to equivalence of imbeddings). It is a very simple matter to determine the faces of an imbedding and thus its associated dual from its rotation system representation. Finally a rotation system containing only type-0 edges is called *pure*. There is a 1-1 correspondence (up to equivalency of imbeddings) between orientable imbeddings and pure rotation systems. For more details see [10].

3 Bounds via Duality

This first proposition is due to Richter and Shank [19]:

Proposition 1 *Let $G = (V,E)$ be a simple undirected graph and let I denote any imbedding of G on some surface S. Let $G^D_I = (R,E)$ be the dual of G with respect to I. Let C be a minimal cut of G and C^D its corresponding subgraph in G^D_I.*

Then the degree of every vertex in C^D is even, i.e. C contains an even number of boundary edges from every region in G.

Define a *cycle* as a loop or closed path and a *circuit* to be a connected set of one or more edge disjoint cycles.

Corollary 3.1 *C^D can be expressed as a set of circuits.*

This follows from the fact that all vertices in C_D have even degree. Thus the number of cyclic subgraphs of size i in G^D_I is an upper bound for the number of cutsets of size i in G.

Let $Sp_i(G)$ indicating the number of spanning forests on i edges of G.

Corollary 3.2 *For an all-terminal undirected network G, $F_i \geq Sp_i(G^D)$.*

This follows from the fact that all network cutsets of size i map to cyclic subgraphs of size i in the dual.

The corollary indicates that for all-terminal networks with equal edge operational probabilities it may be possible to improve the lower bound on some of the F_i's. An algorithm of Liu and Chow [15] can be used to compute the number of k component spanning forests in time polynomial in m but exponential in k . Therefore if there are n_d vertices in the topological dual, we can efficiently compute lower bounds for $F_{n_d-1}, ..., F_{n_d-k}$ where k is some small integer constant. If any of these are greater than the lower bounds produced by conventional Kruskal-Katona methods, they can be used instead to produce a better lower bound. Similarly, if any of these F_i lower bounds are better than the lower bounds produced by the Ball-Provan bounds, we can determine a better Ball-Provan lower bound using the improved F_i lower bounds (See Section 4). For more information about the Kruskal-Katona and Ball-Provan techniques see [5].

We now turn our attention to two-terminal networks.

Proposition 2 *An $s-t$ cut C in an $s-t$ network $G = (V, E, \rho, \{s,t\}) \cup e_{st}$ is a cyclic subgraph C^D in any $s-t$ dual $G^D_{I_{st}} = (V^D, E^D, \overline{\rho}, \{s', t'\})$ of G. Furthermore s' and t' both lie on some common circuit of C^D.*

Corollary 3.3 *C^D is an $s'-t'$ pathset in $G^D_{I_{st}} - e_{st}$.*

By the above proposition any $s-t$ cut $\hat{C}$ in $G \cup e_{st}$ forms a cyclic subgraph $\hat{C}^D$ in $G^D_{I_{st}}$ where s' and t' are contained on some circuit. $C = \hat{C} - e_{st}$ is a cut in G and C^D is a subgraph of $G^D_{I_{st}} - e^D_{st}$ in which s' and t' are connected.
From this point on, $G^D_{I_{st}}$ refers to the topological $s-t$ dual with edge e^D_{st} removed.

Corollary 3.4 *Let $G = (V, E, \rho, \{s,t\})$ be a two-terminal network and let*

$$G^D_{I_{st}} = (V^D, E^D, \overline{\rho}, \{s', t'\})$$

be any $s-t$ dual of G. Then $Rel_2[G] \geq 1 - Rel_2[G^D_{I_{st}}]$.

Let $C = \{C_1, \ldots, C_k\}$ be the set of $s-t$ cutsets of G and $P = \{P_1, P_2, \ldots, P_r\}$ be the set of $s'-t'$ pathsets of $G^D_{I_{st}}$. By corollary 1, $C \subseteq P$. Thus $\sum_{i=1}^{k} \prod_{e \in C_i} q_e \leq \sum_{i=1}^{r} \prod_{e \in P_i} q_e$ so $1 - \sum_{i=1}^{k} \prod_{e \in C_i} q_e \geq 1 - \sum_{i=1}^{k} \prod_{e \in P_i} q_e$ since the sums are between 0 and 1.

Thus $Rel_2[G] \geq 1 - Rel_2[G^D_{I_{st}}]$. This implies that a lower bound for $Rel_2[G]$ can be obtained from an upper bound for $Rel_2[G^D_{I_{st}}]$.

Proposition 3 *Let $\vec{G} = (V, E, \rho, (s,t))$ be a directed 2-terminal network and*

$$\vec{G}^D_{I_{st}} = (V^D, E^{D+}, \overrightarrow{\rho}^+, (s', t'))$$

any directed $s-t$ topological dual of $\vec{G}$. Then the $Rel_2[\vec{G}] \geq 1 - Rel_2[\vec{G}^D]$.

4 All-terminal Implementation

A major stumbling block in implementing this bounding technique is to find an imbedding that produces a large number of spanning forests in the dual. Here we examine the suitability of this method using orientable imbeddings. We first attempt to sample a small percentage of pure rotation systems randomly and select the one producing the dual with the most spanning trees. The higher the genus of the imbedding, the higher the likelihood of a reduction in unique spanning forests due to loops and multiple edges. Unfortunately, the number of high genus imbeddings in our test cases far exceeds the number of low genus imbeddings. Thus, random sampling does not appear to be very productive. Consequently, we develop a heuristic which attempts to construct low genus imbeddings (imbeddings with lots of regions) by ensuring that a maximal set of cycles become faces in the imbedding. The heuristic is as follows:

Let $G = (V, E, \rho)$. Given a subgraph S, let $edges(S)$ denote the set of edges in S

1. Find a minimal cycle C_1 in G. Set $E_u = edges(C_1)$. Set $E_a = E - E_u$. Set $\Theta = \{C_1\}$.
2. For each $e = (v_1, v_2) \in E_u$ find if possible a shortest path P from v_1 to v_2 in E_a. If such a path is found, set $E_a = E_a - edges(P)$, $E_u = E_u + edges(P)$, and $\Theta = \Theta \cup \{P + e\}$. In any case, set $E_u = E_u - \{e\}$. Repeat this step until either $E_u = \emptyset$ or $E_a = \emptyset$.
3. If $E_u = \emptyset$ but $E_a \neq \emptyset$ then find a cycle C in E_a. If such a cycle is found then set $E_a = E_a - edges(C)$, $E_u = edges(C)$, $\Theta = \Theta \cup \{C\}$ and repeat step 2.
4. Find a rotation system for the subgraph $S = (V, E_u)$ that ensures the circuits in Θ form face boundaries. This is always possible because $\cup_{1 \leq C_i \leq |\Theta|} C_i$ defines a planar imbedding.
5. If $E_a \neq \emptyset$ then add the remaining edges in E_a at the end of their respective incident vertices' rotation vectors.

The cycles found in step 2 become faces in the imbedding defined by the rotation system constructed above. Thus, if k cycles are found by the above algorithm, we are assured of at least $k+1$ faces in the produced imbedding. In the final paper, we present computational results using this heuristic.

5 Two-terminal Implementation

Dualization results for two-terminal reliability lower bounds appear to be much better than those for all-terminal reliability. For two-terminal reliability, we have a direct correlation between the reliability of the dual and the reliability of the network. Here we examine a heuristic which generates an orientable imbedding.

Once again the major difficulty in implementing this bounding technique is to select a suitable $s-t$ imbedding. As in the all-terminal case, we wish to minimize the genus of our produced imbedding. However, minimizing the genus is not sufficient to ensure that a good imbedding is produced. The longer the minimum $s'-t'$ path in the dual, the better the produced bounds. We therefore employ the following modified version of the all-terminal imbedding heuristic.

Let $G=(V,E,\rho,\{s,t\})$. Given a subgraph S, let $edges(S)$ denote the set of edges in S.

1. Find the shortest $s-t$ circuit C in G (for example, via a mincost flow algorithm), and add edge $e_{st}=(s,t)$ to C to get the two circuits C_1 and C_2. C_1 and C_2 become s' and t' in $G^D_{I_{st}}$. Set $E_u = edges(C_1) \cup edges(C_2)$. Set $E_a = E - E_u - \{e_{st}\}$. Set $\Theta = \{C_1, C_2\}$.

2. For each $e=(v_1,v_2) \in E_u$ find if possible a shortest path P from v_1 to v_2 in E_a. If such a path is found, set $E_a = E_a - edges(P)$, $E_u = E_u + edges(P)$, and $\Theta = \Theta \cup \{P+e\}$. In any case, set $E_u = E_u - \{e\}$. Repeat this step until either $E_u = \emptyset$ or $E_a = \emptyset$.

3. If $E_u = \emptyset$ but $E_a \neq \emptyset$ then find a cycle C in E_a. If such a cycle is found then set $E_a = E_a - edges(C)$, $E_u = edges(C)$, $\Theta = \Theta \cup \{C\}$ and repeat step 2.

4. Find a rotation system for the subgraph $S=(V,E_u)$ that ensures the circuits in Θ form face boundaries. This is always possible because $\cup_{1 \leq C_i \leq |\Theta|} C_i$ defines a planar imbedding.

5. If $E_a \neq \emptyset$ then add the remaining edges in E_a at the end of their respective incident vertices' rotation vectors.

This heuristic for imbedding gives an efficient implementation of the bound in proposition 3. In the final paper, we explore the accuracy of the resulting bound compared to currently available methods.

6 Conclusions

Surface dualization techniques deliver bounds that are competitive with the current best methods. Furthermore, they can be utilized for state space reduction in Monte Carlo and

Most Probable State methods. In the final paper, these applications are discussed further.

References

[1] H.M. AboElFotoh, C.J. Colbourn, "Series-Parallel Bounds for the Two-Terminal Reliability Problem", *ORSA J. Comp 1* 1988, 209-222.

[2] M.O. Ball, J.S. Provan, "Calculating Bounds in Reliability and Connectedness in Stochastic Networks", *Networks 13* 1983, 253-278

[3] T.B. Brecht, C.J. Colbourn, "Lower Bounds on Two-Terminal Network Reliability", *Discrete Appl. Math 21* 1988, 185-198.

[4] M.K. Chari, *Combinatorial Aspects of Degeneracy in Linear Systems and Network Reliability*, Ph.D. Thesis 1990, Univ. of North Carolina at Chapel Hill.

[5] C.J. Colbourn, *The Combinatorics of Network Reliability*, Oxford University Press, Oxford, 1987.

[6] C.J. Colbourn, "Edge-Packings of Graphs and Network Reliability", *Discrete Math. 72* 1988, 49-61.

[7] C.J. Colbourn, D.D. Harms, "Evaluating Performability: Most Probable States and Bounds", *Tech. Report CMPT TR 92-06* Dept. Comp. Sci., Simon Fraser University, Burnaby, BC, Canada, 1992.

[8] C.J. Colbourn, L.D. Nel, T.B. Boffey, D.F. Yates, "Network Reliability and the Probabilistic Estimation of Damage from Fire Spread",*Annals of Operations Research*, to appear.

[9] G.S. Fishman, "A Monte Carlo Sampling Plan for Estimating Network Reliability",*Oper. Res. 34* 1986, 581-594.

[10] J.L. Gross, T.W. Tucker, *Topological Graph Theory*, John Wiley and Sons, New York, 1987.

[11] D.D. Harms, C.J. Colbourn, "Renormalization of Two-Terminal Reliability", *Networks*, to appear.

[12] D.D. Harms, *A Symbolic Algebra Environment for Research in Network Reliability* Ph.D. Thesis, Simon Fraser University 1992.

[13] V.A. Kaustov, Y.I. Litvak, I.A. Ushakov, "The Computational Effectiveness of Reliability Estimates by the Method of Nonedge-Intersecting Chains and Cuts", *Soviet J. Comput. Systems Sci. 24* 1986, 70-73.

[14] J.B. Kruskal, "The Number of Simplices in a Complex", *Mathematical Optimization Techniques*, Univ. of Calif. Press, edited by C.R. Bellman 1963, 251-278.

[15] C.I. Liu, Y. Chow, "Enumeration of Forests in a Graph", *Proc. of the Amer. Math. Soc. 83* 1981, 659-662.

[16] L.D. Nel, C.J. Colbourn, "Locating a Broadcast Facility in an Unreliable Network". *INFOR 28* (1990), 363-379.

[17] L.D. Nel, H.J. Strayer, "Two-Terminal Reliability Bounds based on Edge Packing by Cutsets", *J. Comb. Math. Comb. Comp.* (to appear)

[18] J.S. Provan, "The Complexity of Reliability Computations in Planar and Acyclic Graphs", *SIAM J. Computing 15* (1986), 694-702.

[19] B Richter, H. Shank, "The Cycle Space of an Embedded Graph", *J. Graph Theory 8* (1984), 365-369.

[20] J.G. Shanthikumar, "Reliability of Systems with Consecutive Minimal Cutsets", *IEEE Trans. on Reliab. 36* (1987), 546-550.

[21] J.G. Shanthikumar, "Bounding Network Reliability Using Consecutive Minimal Cutsets", *IEEE Trans. on Reliab. 37* (1988), 45-49.

[22] A.W. Shogan, "Sequential Bounding of the Reliability of a Stochastic Network", *Operations Research 34* (1976), 1027-1044.

[23] S. Stahl, "Generalized Embedding Schemes", *Journal of Graph Theory 2* (1978), 41-52.

[24] H. Strayer, C. Colbourn, "Bounding Two-terminal Network Reliability via Surface Duality", *CLAIO 6*, to appear.

A LOCATION-DISTRIBUTION MODEL FOR A BEER BREWER

Haldun Süral*, Murat Köksalan* and Ömer Kırca *,+

**Industrial Engineering Department, Middle East Technical University,*

06531 Ankara, TURKEY

+Department of Decision Sciences, National University of Singapore,

0511 SINGAPORE

In this paper we discuss a location-distribution problem and its implementation for a beer brewer company. The company has three beer breweries, two malt factories, three product types and about 270 customer zones. We consider 15 candidate locations for the new breweries to be established in the near future. We study the current distribution plan and evaluate the alternative locations for the new breweries.

We develop two models to solve the problem. In both models we aggregate different types of beer into a single type as the transportation cost differences between different types are small. This aggregation reduces the problem size and it makes it easier to manage on PC's. We also excluded the fixed cost of construction from the models as this cost did not vary much among alternative locations.

The first model is a mixed integer programming model that considers the transportation costs of malt to the breweries and beer to the demand points. The model solves for the distribution plan of malt and beer, and the locations of new breweries. In the second model, in addition to the above, we incorporated inventory carrying. The seasonality of demand is an important issue in beer consumption and this brings serious implications on the amount of inventory carried. The effect of high inflation rate in the economy also magnifies the importance of carrying inventories. In this case the model becomes a multi-period model where months represent periods.

The application of the models is done in four modules: the data manipulation module in Lotus 123, the model generation module in Fortran, the solution in LINDO, and the reporting module in Fortran. The program is designed so that different applications of models, such as clustering of the customer zones, varying the number of product types, and including the fixed costs of new establishments are possible.

A series of runs for both models is executed on an IBM 486 compatible computer. Each run takes several minutes. Since the estimation of the true inventory holding cost is not straightforward, we represent the trade-off between transportation and inventory expenses by restricting the amount of budget tied up in inventory. We solve the

problem a number of times by setting this budget to different levels and plot the transportation cost vs. inventory budget which is a compatible basis for comparing the two cost items.

We present the results of our study and discuss the implementation under several scenarios. The results obtained by the model have been found useful by the management and they decided to locate their plant at the location suggested by our study.

Keywords: Location, distribution

UNIVERSIDADE DO MINHO
ESCOLA DE ENGENHARIA
DEPARTAMENTO DE PRODUÇÃO E SISTEMAS

DISCRETE DYNAMIC PROGRAMMING IN ENVIRONMENTAL OPTIMISATION

by

Maria Madalena Teixeira de Araújo

Dept. of Engineering Production and Systems
University of Minho
Quinta da Veiga - Azurém - 4800 Guimarães
PORTUGAL

ABSTRACT

Forward discrete dynamic programming was used to optimise the pollution abatement effort along a river basin, through the adequate location and operation of treatment plants, at minimum cost. Upper and lower bounds were set in terms of efficiency of pollution removal, which is the 'semi-independent' variable, in order to bound the solution in a small neighbourhood. The efficiencies had to be adequately translated into the independent variable, the river water quality standard (measured in mg/l of BOD, Biochemical Oxygen Demand), since it determines whether the value obtained is a feasible solution or not.

Keywords: dynamic programming, global optimisation, interval approach.

1. INTRODUCTION

This paper presents part of a major research undertook in order to develop a computerized framework to assist in the management of a river basin industrial wastewater system. It is applicable to rivers following a Dobbins type model and the quality parameters under surveillance are the Biochemical Oxygen Demand (BOD) and the Dissolved Oxygen (DO). WODA commercial package was used and adapted for the attainment of 4 BOD and 1 DO standards, using dynamic programming, the subject of this paper. Geometric programming was used to select the minimum cost preliminary treatment plants' design. Other pollution abatement measures considered are flow augmentation and artificial aeration. A 'compromised' minimum cost solution between a completely centralised allocation of pollution abatement effort and a single company most economical solution is determined. The number of concentrated effluent discharge points (n) determine the number of stages or reaches being analysed (n-1), one at a time.

The objective function to be minimised is the cost of the pollution abatement measures, which is a sum of a maximum of three terms, depending on the number of different types of measures being considered (treatment plants, artificial aeration, and low flow augmentation). Four BOD standards, 8.0, 6.5, 5.0, 3.5 mg/l, and one DO standard 5.0 mg/l, were tested. These include, as middle and lower values, the EEC standards. The optimisation problem was solved using a forward dynamic programming procedure which

evaluated the minimum cost abatement efficiency.

2. OPTIMISATION PROCEDURE

2.1. Preliminary Approaches

The simulation routine would run for the initial conditions of the data to check if any violation of each standard occurs. When it happens, the concentration of BOD in the concentrated discharge immeadiately upstream is reduced. The first reduction is 35% for technological reasons, and after that the reduction is done in steps of 5%, until no violation occurs. If the violation is mainly of the DO standard, then artificial aeration can be tested. If the improvement is not enough, or if the violation is not only on the DO level, then flow augmentation can be studied to couple with treatment.

The cost of complying with each standard in each reach is calculated and a variable, ROOT, depending on the reach and on the standard is stored representing the pathway taken. Indeed such a variable characterises a node in the dynamic programming algorithm being implemented. It was called NODE. The position and the value of the node's digits give information about the reach number and the standard being attained, respectively. The node structure is:

Node digits position	=	1st	2nd	3rd	4th ...
Reach number:		1	2	3	4 ...
Node digits value	=	1	2	3	4
Standard codes:		8.0	6.5	5.0	3.5

Two procedures were developed for the analysis of the

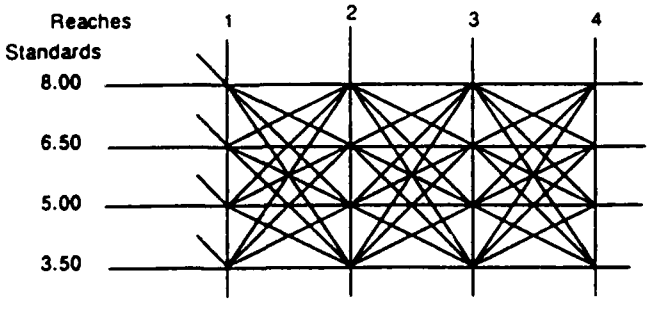
reaches. One of them is based on the assumption that any standard can be achieved in any reach, independently of the standard achieved in the previous reach or reaches (case A). The other procedure (case B) presupposes that it is reasonable to assume that any reach downstream will not be required to comply with a tighter standard than reaches upstream. In other words, this case evaluates the minimum cost of achieving at least a certain quality level in all reaches upstream. Results obtained in both cases with test data will follow.

Case A. The following final results (1) were obtained:

RESULTS AFTER REACH 5 (last reach)

BOD Stand. (mg/l)	Efficiency of BOD removal (reach i, i=1,reach) (%)	Node	Cost (US$ 1000)
8.00	0 ; 75 ; 0 ; 0 ; 0	23211	416.27
6.50	0 ; 75 ; 0 ; 0 ; 0	23212	416.27
"	0 ; 75 ; 0 ; 75 ; 0	23233	832.53
3.50	impossible	xzty 4 i	-

Analysing this final result we can say that only two standards, BOD$\leq$8.0 and 6.5 mg/l are being achieved everywhere, which is far from the objective of the simulation. This network corresponds to the most common dynamic programming structure



(1) All the costs are reported at 1969 values, and the total number of reaches tested was 5.

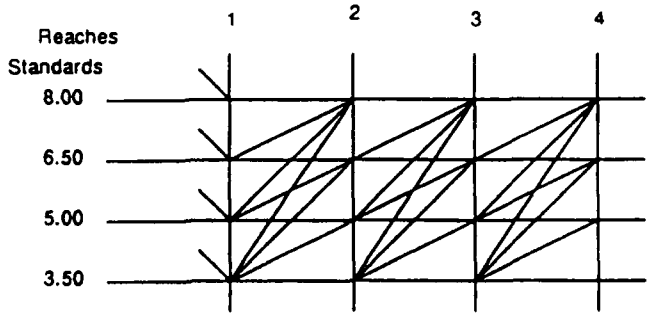
Another constraint should be tested and carried out at every stage of the dynamic programming procedure - the cost of attaining at least every standard in all reaches upstream, which was the single objective of case B.

Case B. The following final results were obtained:

RESULTS AFTER REACH 5 (last reach)

BOD Stand. (mg/l)	Efficiency of BOD removal (reach i, i=1,reach) (%)	Node	Cost (US$ 1000)
8.00	35 ; 0 ; 0 ; 35 ; 0	31111	536.43
6.50	0 ; 40 ; 40 ; 35 ; 0	22222	824.76
5.00	45 ; 70 ; 35 ; 45 ; 0	44333	1230.27
3.50	impossible	xzty 4 i	-

By considering, for instance the node 44333 we can see that the minimum cost of complying with, at least standard 5.00, (number 3, in the node) was obtained when in reaches 1 and 2 a tighter standard (3.5 - number 4, in the node) was obeyed. So, in spite of treating more than necessary, a smaller overall cost was obtained. Or, in other words, the more than necessary pollution reduction in a certain reach can result in an overall reduction in cost by avoiding the need of action(s) downstream. This network corresponds to the simplified dynamic programming structure



Conclusion: the constrained solution, case B, has the advantage of allowing us to specify, at any stage, the

minimum level of BOD attained in any point upstream. This is, in fact, the last standard to be checked, assuming that the optimisation procedure starts with the less tight standard. Case A, was discarded, provided adequate constraints were added to the formulation of Case B.

We notice that standard 3.5 has not been achieved in every reach, the solution obtained presenting it as impossible to attain. This fact would oblige us to study the use of flow-augmentation to couple with treatment, from the reach onwards where treatment alone became insufficient. The overall cost would certainly rise very substantially.

2.2. Charging Schemes

Several methods, without using any optimisation algorithm were tried, in order to compare different charging schemes for the usual four BOD standards being considered. These were a minimum treatment method, minimum treatment method but enforcing primary treatment in every reach, and a constant removal method.

The results obtained are sumarised below, in Table 1.. As we can see the closest method to the minimum cost is the Minimum Treatment Method.

2.3. Refined Approach

In fact, the Equal Treatment Method can be considered an upper limit for the minimum cost solution, as far as cost itself is concerned. The same does not apply to removal efficiencies, which have been the 'semi-independent' variables in the optimisation procedure above. The really

independent variables are the standards. When comparing the performance of the different charging schemes it was noticed that standard 3.5 can be achieved by treatment only, requiring an average removal rate of 75%. This fact proved that the above optimisation was wrong and needed to be altered. Consequently, the optimisation procedure was re-studied. The problems involved with discrete dynamic programming of this type, are very closely related to the combinatorial aspect of the resulting steps. In order to reduce them it is advisable to use upper and lower limits, if known, to bound the solution in a small neighbourhood.

Table 1. – Evaluation of Different Charging Methods

METHOD	Stand. Tested	Removal Effic.	Minimum Stand. attain. each reach 1				2				3				4				5				Cost Total 10**6	% MC sol
			1	2	3	4	1	2	3	4	1	2	3	4	1	2	3	4	1	2	3	4		
Minimum Treatm.	1(8.0)		X					X				X				X							536.42	100
	2(6.5)		X						X				X				X						836.13	100
	3(5.0)			X						X				X				X					1291.77	123
	4(3.5)				X						X				–				–				–	
Minimum Primary Treatm.	1		X					X				X				X				X			1341.07	250
	2		X						X				X				X				X		1341.07	163
	3			X						X				X				X				X	1559.98	148
	4				X						X				–				–				–	
Equal Treatm.	1	35	X					X				X				X				X			1341.07	250
	2	35	X						X				X				X				X		1341.07	163
	3	50		X						X				X				X				X	1478.69	141
	4	70			X						X				X				X				1799.33	117
Minimum Cost	1		X					X				X				X				X			536.42	
	2		X						X				X				X				X		836.13	
	3			X						X				X				X				X	1051.56	
	4				X						X				X				X				1532.96	

As we have seen, the Equal Treatment Method would be a good upper limit if the cost was the independent variable. Using the simulation routine we can change the efficiency of removal values, by a predefined 5% value, which, in turn,

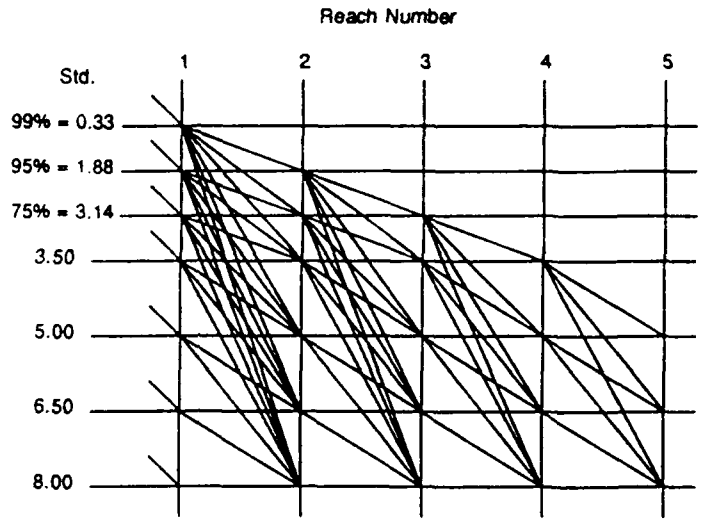
defines the discrete grade used.

We should be looking for an upper bound on the efficiency, since we know that treating more than necessary upstream may result in lower overall costs. If we knew that the optimum would be, say, limited by 85% efficiency of removal, then, for reach 1 we could start analysing the minimum efficiency needed to comply with the standard in that reach. All other upper efficiencies up to 85% would be possible, at a higher cost. When proceeding to reach 2 the same analysis would be done, for each of the possible outcomes from reach 1. The minimum cost solution would be selected, after ensuring that it could lead to a feasible solution, by which we mean a node which can lead to the achievment of the standard until the mouth of the river.

Lacking real data for the desired upper limit, we used the maximum possible efficiency of 99%, and two lower efficiencies, in order to split the possible space, and reduce iterations for some of the standards. One of the efficiencies was 95%, since it is the limit for secondary treatment, and the other was 75%, for no other reason than being the average value obtained by the Equal Treatment Method.

An important correspondance still to be established was the need to translate the efficiencies into standards, since these determine whether the value obtained is a solution or not. The main program was altered to allow for three introductory runs of the simulation routine for a constant removal rate 75%, 95% and 99%, for each of the concentrated

discharge BOD concentrations. The maximum BOD value obtained in each run is automatically selected and stored as a 'slack' or 'dummy' standard. Thus, the attainment of standard 8.0, for instance, will now have 6 'ceilings' to test in every reach. The complete diagram is shown below.



The optimisation proceeds by checking every solution for feasibility, by replacing downstream BOD concentrated effluent values for the minimum possible (99% reduction) and running the simulation routine. Only if the intermediate solution being determined passes this test, will it be stored. A new routine was written for this purpose. If no solution, for a certain standard, is achieved, then treatment is not enough and has to be coupled with flow augmentation. The final results obtained are shown below:

The cost and efficiency values are not completely equivalent with those found previously, also due to a slight difference in the value of the quality parameters.

Corrective action was taken, mainly by rewriting the proper software routines.

RESULTS AFTER REACH 5

BOD Stand. (mg/l)	Efficiency of BOD removal (reach i, i=I,reach) (%)	Node	Cost (US$ 1000)
8.00	35 ; 0 ; 0 ;35 ; 0	31111	536.42
6.50	45 ;40 ; 0 ;35 ; 0	43222	836.13
5.00	75 ;55 ; 0 ;55 ; 0	54333	1,051.56
3.50	75 ;75 ;55 ;70 ; 0	55444	1,532.96

3. CONCLUSIONS

Since the research was conducted on Portuguese data, the optimisation procedure based on several standards was devised in order to give decision-makers comparative costs. This would allow them to select the proper choice, for instance during the legal adaptation period to EEC legislation, of a slackened standard having less drastic effects in an already weakened economy.

It is known that the optimum can be improved by refining the grade. However, by using the simulation routine which increases the computing time considerably for any new standard tested, as well as the use of the available cost functions and preliminary treatment plant designs, it is thought that a sufficient degree of accuracy is achieved.

4. BIBLIOGRAPHY

[1] BECK, M.B., Water Quality Management: A Review of the Development and Application of Mathematical Models, (IIASA, Lecture Notes in Engineering, N.11), Springer-Verlag, (Berlin, 1985).

[2] BELLMAN, R., Dynamic Programming, Princeton University Press, (Princeton, New Jersey, 1957).

[3] BELLMAN, R. and DREYFUS, S., Applied Dynamic Programming, Princeton University Press, (Princeton, New Jersey, 1962).

[4] KRASZEWSKI, A. and SONCINI-SESSA, R., WODA - A Computer Package for the Identification and Simulation of a BOD-DO River Quality Model, Cooperativa Libraria Universitaria del Politecnico, (Milano, 1984).

[5] SHIH, C.S. and KRISHNAM, P., "Dynamic Optimization for Industrial Waste Treatment Design", Journal of the Water Pollution Control Federation, 41, 10, (1969), 1787-1802.

[6] TEIXEIRA DE ARAUJO, M.M., Economic Planning in Portuguese Industrial Wastewater, Ph.D. Thesis, Birmingham University (U.K.), 1986.

A Review of Goal Programming and its Applications

M.Tamiz[1], D.F.Jones
University of Portsmouth, U.K.
and

E.El-Darzi
University of Westminster, U.K.

Abstract

This paper presents a review of the current literature on the branch of multi-criteria decision modelling known as Goal Programming (GP). The result of our indepth investigations of the two main GP methods, lexicographic and weighted GP together with their distinct application areas is reported.

Some guidelines to the scope of GP as an application tool are given and methods of determining which problem areas are best suited to the different GP approaches are stated. The correlation between the method of assigning weights and priorities and the standard of the results is also ascertained.

Key Words: Goal Programming, Lexicographic, Weighted

1 Introduction

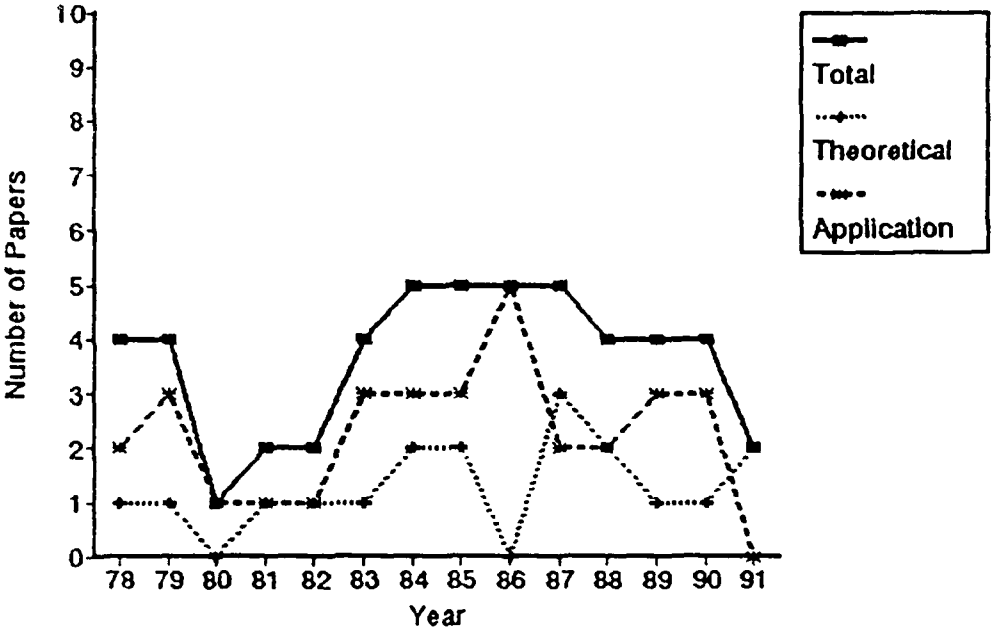
Goal Programming is a branch of multi-criteria decision analysis. It was first introduced by Charnes and Cooper in 1955 [1]; more explictly defined by the same authors in 1961 [2]; and further developed by Ijiri [3] during the 1960's. The first books dedicated to GP by Lee [4] and Ignizio [5] appeared during the early to mid 1970's.

In the 1970's GP and its variants were applied to many different subject areas. These include academic resource planning [6, 7], accounting [8], agricultural planning [9], energy forecasting [10], portfolio management [4, 11], water resource planning [12], library management [13], and media scheduling [14].

[1] *Address for Correspondence: Dr. M. Tamiz, School of Mathematical Studies, University of Portsmouth, Mercantile House, Hampshire Terrace, Portsmouth, PO1 2EG, England.*

Questions were raised as to the effectiveness of GP as an application tool by Zeleny [15] and Harrald [16] during the late 1970's and early 1980's, but GP still grew in popularity judging by the increase of papers applying GP during the 1980's. Table 1 shows the number of papers on the subject of GP (both theoretical and applicational) during the past decade, in the *Journal of the Operational Research Society*, which may well be considered to be a representative sample of GP publications.

Table 1 : Frequency of GP papers in the Journal of the OR Society



The results show a continuing healthy interest in GP. Among the application areas utilised or extended in the past ten years are farm growth planning [17], diet planning [18, 19], locational analysis [20, 21], academic resource planning [22, 23], manpower planning [24], police scheduling [25], portfolio analysis [26], interest rate models [27], engineering [28], and manufacturing [29].

With the onset of powerful computers, sophisicated algorithms have been developed by Ignizio [30], Schniederjans and Kwak [31], and others [32, 33, 34]. Olson [35] compares computational time for four GP algorithms and demonstrates the benefits of using Revised-Simplex and Primal-Dual algorithms to solve GP problems. These have made solutions to large scale GP problems possible and several papers have been published exploiting this [24, 25, 36].

Work has also continued into special case GP algorithms: Integer, Zero–One, Fuzzy, Interactive and Chance–Constrained. A breakdown of publications in these areas is given in Romero [37]. In total he lists 355 papers dealing with GP applications in 26 distinct areas.

Research has been done to apply other Multi-Criteria and Management Science techniques to Goal Programming. These include interactive multi-criteria methods [38], 'Delphi' techniques [39, 40], Saaty's [41] analytical hierarchy approach [36, 23, 39], and resource planning and management systems(RPMS) networks [42]. Recently papers have been published dealing with some of the perceived 'errors' in G.P [37, 40, 43]. and explaining how these can be avoided by the correct setting of weights, goals, priority levels etc.

The remainder of the paper will be divided into four sections. Section 2 will deal with lexicographic(pre-emptive) GP, section 3 with weighted GP(non pre-emptive), section 4 with the connection between utility functions and GP, finally section 5 will draw conclusions as to the current direction of GP and the direction of the authors' future research.

2 Lexicographic GP

Of the 355 papers mentioned by Romero [37], 226 use the concept of Lexicographic GP(LGP), which requires the pre–emptive ordering of priority levels. The standard LGP model can be algebraically represented as:

$$Lex\ min\ \mathbf{a} = (g_1(\mathbf{n},\mathbf{p}), g_2(\mathbf{n},\mathbf{p}), ..., g_K(\mathbf{n},\mathbf{p}))$$

subject to,

$$f_i(\mathbf{x}) + n_i - p_i = b_i \quad i = 1, ..., m$$

This model has K priority levels, and m objectives. a is an ordered vector of these K priority levels.

A standard 'g' (within priority level) function is given by:

$$g_k(\mathbf{n},\mathbf{p}) = \alpha_{k_1} n_1 + ... + \alpha_{k_m} n_m + \beta_{k_1} p_1 + ... + \beta_{k_m} p_m$$

This paper will summarize the development of algorithms to solve the LGP model, work on the multi–dimensional dual [30, 44], and current thinking on methods of priority ranking and weighting within the priority levels. Some applications of LGP will be commented on, in an effort to outline which types of problem are suitable for an LGP approach, and which are better solved using other techniques.

3 Weighted GP

Weighted (or non–pre-emptive) GP(WGP) requires no pre-emptive ordering of the objective functions. Instead all the different deviations are placed in a single priority level objective with different weights to represent their importance.

Work has also continued into special case GP algorithms: Integer, Zero–One, Fuzzy, Interactive and Chance–Constrained. A breakdown of publications in these areas is given in Romero [37]. In total he lists 355 papers dealing with GP applications in 26 distinct areas.

Research has been done to apply other Multi-Criteria and Management Science techniques to Goal Programming. These include interactive multi-criteria methods [38], 'Delphi' techniques [39, 40], Saaty's [41] analytical hierarchy approach [36, 23, 39], and resource planning and management systems(RPMS) networks [42]. Recently papers have been published dealing with some of the perceived 'errors' in G.P [37, 40, 43]. and explaining how these can be avoided by the correct setting of weights, goals, priority levels etc.

The remainder of the paper will be divided into four sections. Section 2 will deal with lexicographic(pre-emptive) GP, section 3 with weighted GP(non pre-emptive), section 4 with the connection between utility functions and GP, finally section 5 will draw conclusions as to the current direction of GP and the direction of the authors' future research.

2 Lexicographic GP

Of the 355 papers mentioned by Romero [37], 226 use the concept of Lexicographic GP(LGP), which requires the pre–emptive ordering of priority levels. The standard LGP model can be algebraically represented as:

$$Lex\ min\ \mathbf{a} = (g_1(\mathbf{n},\mathbf{p}), g_2(\mathbf{n},\mathbf{p}), ..., g_K(\mathbf{n},\mathbf{p}))$$

subject to,

$$f_i(\mathbf{x}) + n_i - p_i = b_i \quad i = 1, ..., m$$

This model has K priority levels, and m objectives. a is an ordered vector of these K priority levels.

A standard 'g' (within priority level) function is given by:

$$g_k(\mathbf{n},\mathbf{p}) = \alpha_{k_1} n_1 + ... + \alpha_{k_m} n_m + \beta_{k_1} p_1 + ... + \beta_{k_m} p_m$$

This paper will summarize the development of algorithms to solve the LGP model, work on the multi–dimensional dual [30, 44], and current thinking on methods of priority ranking and weighting within the priority levels. Some applications of LGP will be commented on, in an effort to outline which types of problem are suitable for an LGP approach, and which are better solved using other techniques.

3 Weighted GP

Weighted (or non–pre-emptive) GP(WGP) requires no pre-emptive ordering of the objective functions. Instead all the different deviations are placed in a single priority level objective with different weights to represent their importance.

Algebraically, a WGP has the following structure:

$$Min \ \ a = \sum_{i=1}^{k} (\alpha_i n_i + \beta_i p_i)$$

Subject to,

$$f_i(\mathbf{x}) + n_i - p_i = b_i \quad i = 1, ...m$$

$$\mathbf{x} \in C_s$$

Where C_s is an optional constraint set. Of interest here are the problems caused by incommensurability, i.e. objective functions being measured in different units, and techniques used to overcome this. As in the LGP case, application areas will be outlined.

4 Utility Functions

The third section will deal with the connections between utility functions and the different types of GP. It will explore the literature on the problems caused in reconciling LGP and utility function theory. It will also examine recently developed techniques to model GP's more closely around their underlying objective functions [45].

5 Summary and Conclusions

The final section will draw conculsions as to the scope and limitations of GP and highlight areas in which the authors intend to conduct further research.

References

[1] CHARNES, A., COOPER, W.W., and FERGUSON, R. Optimal Estimation of Executive Compensation by Linear Programming, *Management Science*, 1, 138-151. 1955.

[2] CHARNES, A. and COOPER, W.W.. *Management Models and Industrial Applications of Linear Programming.* John Wiley and Sons, New York. 1961.

[3] IJIRI, Y. *Management Goals and Accounting for Control* . North Holland, Amsterdam. 1965.

[4] LEE, S.M. *Goal Programming for decision analysis.* Auerback, Philadelphia, 1972.

[5] IGNIZIO, J.P. *Goal Programming and Extensions.* Lexington, Mass.: Heath (Lexington Books), 1976.

[6] ALBRIGHT, S.C. Allocation of Research Grants to University Research Proposals. *Socio-Economic Planning Sciences*, **9**, 189-195. 1975.

[7] JOINER, C. Academic Planning Through The Goal Programming Model *Interfaces*, **10**, 86-91. 1980.

[8] KILLOUGH, L.N. and SOUNDERS, T.L. A Goal Programming Model for Public Accounting Firms, *The Accounting Review*, **48**, 268-279. 1973.

[9] WHEELER, B.M. and RUSSELL, J.R.M. Goal Programming and Agricultural Planning, *Operational Research Quarterly*, **28**, 21-32. 1977.

[10] SAMOUILIDIS, J.E. and PAPPAS, I.A. A Goal Programming Approach to Energy Forecasting, *European Journal of Operational Research*, 5, 321-331. 1980.

[11] KUMAR, P.C., PHILIPPATOS, G.C., and EZZELL, J.R. Goal Programming and the Selection of Portfolios by Dual-purpose Funds. *Journal of Finance*, **33**, 303-310. 1979.

[12] CHISMAN J.A., and RIPPY, D. Optimal Operation of a Multipurpose Resorvior using Goal Programming, *The Clemson University Review of Industrial Management and Textile Science*, Fall, 69-82. 1977.

[13] HANNAN E.L. Allocation of Library Funds for Books and Standing Orders – a Multiple Objective Formulation, *Computers and Operational Research*, **5**, 109-114. 1978.

[14] DE KLUYVER C.A. An Exploration of Various Goal Programming Formulations–With Application to Advertising Media Scheduling, *Journal of the Operational Research Society*, **30**, 167-171. 1979.

[15] ZELENY, M. The Pros and cons of Goal Programmng. *Computers and Operations Research*, **8**, 357–359, 1982.

[16] HARRALD, J., LEOTTA, J., WALLACE, W.A. and WENDELL, R.E. A Note on the Limitations of Goal Programming as Observed in Resource Allocation for Marine Enviromental Protection. *Naval Research Logistics Quarterly*, **25**, 733-739. 1978.

[17] DOBBINS, C.L. and MAPP, H.P. A Comparison of Objective Function Structures used in a Recursive Goal Programming–simulation Model of Farm Growth. *Southern Journal of Agricultural Economics*, **14**, 9-16. 1982.

[18] LARA, P. and ROMERO, C. An Interactive Multigoal Progamming Model for Determining Livestock Rations: an Application to Dairy Cows in Andalusia, Spain. *Journal of the Operational Research Society*, **43**, 945-953. 1992.

[19] NEAL, H.L., FRANCE, J., and TREACHER, T.T. Using Goal Programming in Formulating Rations for Pregnant Ewes, *Animal Production* , **42**, 97-104. 1986.

[20] MIN, H. A Model–Based Decision Support System for Locating Banks. *Information and Management*, **17**, 207-215. 1989.

[21] KWAK, N.K. and SCHNIEDERJANS, M.J. A Goal Programming Model for Selecting a Facility Location Site. *RAIRO–Operations Research*, **9**, 1-14. 1985.

[22] GREENWOOD, A.G. and MOORE, L.J. An Inter–temporal Multi–goal Linear Programming Model for Optimizing University Tuitition and Fee Structures, *Journal of the Operational Research Society*, **38**, 599-613. 1987.

[23] DIMINNIE, C.B. and KWAK, N.K. A Hierarchical Goal–Programming Approach to Reverse Resource Allocation in Institutions of Higher Learning, *Journal of the Operational Research Society*, **30**, 59-66. 1986.

[24] FRANZ, L.S., BAKER, H.M., LEONG, G.K., and RAKES, T.R. A Mathematical Model for Scheduling and Staffing Multiclinic Health Regions, *European Journal of Operational Research*, **41**, 277-289. 1989.

[25] SALADIN, B.A. Goal Programming Applied to Police Patrol Allocation, *Journal of Operations Management*, **2**, 239-249. 1982.

[26] LEVARY, R.R. and AVERY, M.L. On the Practical Application of Weighting Equities in a Portfolio via Goal Programming, *Opsearch*, **21**, 246-261. 1984.

[27] BOOTH, G.G. and BESSLER, W. Goal Programming Models for Managing Interest Rate Risk, *Omega*, **17**, 81-89. 1989.

[28] IGNIZIO, J.P. Antenna Array Beam Pattern Synthesis via Goal Programming, *European Journal of Operational Research*, **6**, 286-290. 1981.

[29] O'GRADY, P.J., and MENON, U. A Multiple Criteria Approach for Production Planning of Automated Manufacturing, *Engineering Optimization*, **8**, 161-175. 1985.

[30] IGNIZIO, J.P. An Algorithm for Solving the Linear Goal Programming Problem by Solving its Dual, *Journal of the Operational Research Society*, **36**, 507-515. 1985.

[31] SCHNIEDERJANS, M.J. and KWAK, N.K. An Alternative Solution Method for Goal Programming Problems : A Tutorial, *Journal of the Operational Research Society*, **33**, 247-251. 1982.

[32] ARTHUR, J.A. and RAVINDRAN, A. An Efficent Goal Programming Algorithm Using Constraint Partitioning and Variable Elimination, *Management Science*, **24**, 1109-1119. 1978.

[33] LEE, S.M. and LUEBBE, R.L. A Zero–One Goal–Programming Algorithm Using Partitioning and Constraint Aggregation. *Journal of the Operational Research Society*, **38**, 633-640. 1987.

[34] MARKLAND, R.E. and VICKERY, S.K. The Efficent Computer Implementation of a Large–Scale Integer Goal Programming Model, *European Journal of Operational Research*, **26**, 341-354. 1986.

[35] OLSON, D. A Comparison of Four Goal Programming Algorithms, *Journal of the Operational Research Society*, **35**, 347-354. 1984.

[36] GASS, S.I., A Process for Determining Priorities and Weights for Large–Scale Linear Goal Programmes, *Journal of the Operational Research Society*, **37**, 779-785. 1986.

[37] ROMERO, C. *Handbook of Critical Issues in Goal Programming*. Pergamon Press, 1991.

[38] MASUD, A.S. and HWANG, C.L. Interactive Sequential Goal Programming, *Journal of the Operational Research Society*, **32**, 391-400. 1981.

[39] KHORRAMSHAGOL, R. and AZANI, H. A Decision Support System for Effective Systems Analysis and Planning, *Journal of Information and Optimization Sciences*, **9**, 41-52. 1988.

[40] KHORRAMSHAGOL, R. and HOOSHIARI, A. Three Shortcomings of Goal Programming and their Solutions. *Journal of Information and Optimization Sciences*, **12**, 459-466. 1991.

[41] SAATY, T.L. *The Analytical Hierarchy Process*, McGraw–Hill International, New York. 1981.

[42] SHIM, J.P. and CHIN, S.G. Goal Programming: The RPMS Network Approach, *Journal of the Operational Research Society*, **42**, 83-93. 1991.

[43] MIN, H. and STORBECK, J. On the Origin and Persistence of Misconceptions in Goal Programming, *Journal of the Operational Research Society*, **42** , 301-312, 1991.

[44] IGNIZIO, J.P. *Linear Programming in Single and Multiple Objective Systems.*, Prentice-Hall, Inc., Englewood Cliffs, New Jersey. 1982

[45] MARTEL J.M. and AOUNI B. Incorporating the Decision-maker's Preferences in the Goal–programming Model, *Journal of the Operational Research Society*, **41**, 1121-1132, 1990.

MULTI-STAGE ECONOMIC LOT SCHEDULING PROBLEM

Ayşegül Toker Terzi and Nesim Erkip

Department of Industrial Engineering
Middle East Technical University
06531, Ankara, Turkey

The Economic Lot Scheduling Problem (ELSP) is to economically schedule lots of one or more products on a single machine. Demand is constant, backlogging is not allowed and the planning horizon is infinite. The problem is to minimize total operating cost per unit time which is comprised of setup costs and inventory costs. Setup costs are incurred whenever a production for a lot is begun and inventory carrying costs can be defined as the time value of money tied up in inventory.

An extension to single machine/facility problem is the study of environments where products are manufactured through several operations. Such systems are, in general, called as multi-stage production systems. Multi-stage production systems received a lot of academic attention in recent years focussing on the control of work-in-process inventory and its functional relationship to the manufacturing cycle time. It is a very well known fact by now, the larger the production lot size, the longer the manufacturing cycle, which in turn, increases the work-in-process inventory. There exists a vast literature modelling this relationship to varying degrees in different models for different system configurations.

The Multi-stage Economic Lot Scheduling Problem (MS-ELSP) brings together two important problem characteristics inherent to multi-item and multi-stage problems. In a multi-item problem, the main issue is that of creating schedules which avoids the interference that is likely to occur when two or more products compete for the same facility. We will refer to this as the "feasibility issue". In a multi-stage environment, the production should be synchronized so that concurrent production of the same lot is not possible in the consecutive stages. This characteristic leads to the definition of work-in-process inventory which, in fact, is a tool for the synchronization of production among stages. Thus, in multi-stage problems, creating schedules owing this property will be referred to as "consistency issue". This study addresses the Multi-stage Economic Lot Scheduling Problem with the objective of determining feasible and consistent schedules which result from the conventional tradeoff between setup costs and inventory holding costs comprising the total cost of a schedule.

In this research, we restrict the study of MS-ELSP to serial systems where there are m products to be manufactured through n distinct stages. We first analyze the two product - two stage problem. In order to guarantee feasibility, common cycle solutions in which the possible values of cycle times for all items are constrained to a single cycle time value, T, are sought for. In a two-stage production system, production of a lot on the second stage cannot begin until its production on the first stage is completed. Therefore, production between stages should be synchronized so that we end up with consistent schedules. To ensure consistency, we define a constraint

for each product which also provide information about the work-in-process inventories.

Another important point in this study is the presence of nonnegative setup times. Setup times mean a loss in the productive capacity and their effect on lot sizes is the most significant when the capacity utilization is high. On the other hand, work-in-process inventories tend to increase with increasing capacity utilizations. Therefore, ignorance of setup times will result in overestimated lot sizes due to underestimation of work-in-process inventories.

The mathematical programming formulation of this problem is developed where the objective function is nonlinear with a linear set of constraints. Setting the cycle time to a fixed value, we first linearize the objective function. By using the dual problem and complementary slackness, the optimal solution of this problem and thus the optimal cycle time for the two product - two stage problem are obtained. Besides, we have the exact terms for the work-in-process inventories (queueing inventories: inventory that built up on the previous stage if the successor stage is busy with processing the other products) since they can be expressed explicitly as analytical functions of the cycle time. Then, we generalize our result to multi-product case in a two stage system which constitutes a basis for the analysis of the m-product, n-stage economic lot scheduling problem.

Key words: Economic Lot Scheduling Problem, multi-stage

COMPUTATIONAL EXPERIENCE WITH A PRIMAL-DUAL INTERIOR-POINT METHOD FOR SMOOTH CONVEX PROGRAMMING

J.-P. Vial*
Département d'Ecomomie Commerciale et Industrielle
Université de Genève
102 Bd Carl Vogt, CH-1211 Genève 4, Switzerland

August 1992

Key words: Interior-point method, Primal-dual algorithm, Convex programming, Computational experience.

*This research has been supported by the Fonds National de la Recherche Scientifique, grant #12 – 26434.89.

In this paper we present computational experience with a primal-dual interior point for smooth convex programming problems of the type

$$\begin{array}{ll} \min & c^T x \\ \text{s.t.} & \\ & g(x) \leq 0, \end{array} \tag{1}$$

where $c \in \mathbb{R}^n$ and $g : \mathbb{R}^n \to \mathbb{R}^m$ is a vector-valued function. We assume that each component g_i is convex. Let $s \in \mathbb{R}^m$, be the vector of slack variables. The inequality constraints in (1) are replaced by

$$g(x) + s = 0, \quad s \geq 0.$$

Given a parameter $\mu > 0$, we associate with (1) the barrier problem

$$\begin{array}{ll} \min & c^T x - \mu \sum_{i=1}^{m} \ln s_i \\ \text{s.t.} & \\ & g(x) + s = 0 \\ & s \geq 0. \end{array} \tag{2}$$

We assume that Slater's condition holds:

Assumption 0.1 *There is an $x \in \mathbb{R}^n$ such that $g(x) < 0$.*

We also assume

Assumption 0.2 *The set $\{x : g(x) \leq 0 \text{ and } c^T x \leq \theta\}$ is bounded for all θ.*

Under these assumptions Problem (2) has a solution. The necessary and sufficient conditions for optimality, namely the Karush-Kuhn-Tucker equations, or KKT equations, are

$$Ys - \mu e = 0 \tag{3}$$

$$g(x) + s = 0 \tag{4}$$

$$\left(\frac{\partial g}{\partial x}\right)^T y + c = 0, \tag{5}$$

with $s \geq 0$ and $y \geq 0$. Here

$$\frac{\partial g}{\partial x} = \left\{ \frac{\partial g_i(x)}{\partial x_j} \right\}$$

is the Jacobian matrix of g and $y \in \mathbb{R}^m$ is a vector of dual variables.

Let

$$F : \mathbb{R}^m \times \mathbb{R}^m \times \mathbb{R}^n \to \mathbb{R}^m \times \mathbb{R}^m \times \mathbb{R}^n$$

be a multi-valued function defined by

$$F(z) = \begin{pmatrix} F_c \\ F_p \\ F_d \end{pmatrix} = \begin{pmatrix} Ys - \mu e \\ \frac{\partial L}{\partial y} \\ \frac{\partial L}{\partial x} \end{pmatrix},$$

with $z = (y, s, x)$. F also depends on the parameter $\mu > 0$. With this notation, the KKT system is simply $F(z) = 0$.

We also introduce the Lagrangean

$$L(y; x, s) = c^T x + y^T (g(x) + s). \tag{6}$$

The KKT system (3) – (5) can be rewritten as

$$Ys - \mu e = 0, \quad \frac{\partial L}{\partial x} = 0, \quad \text{and} \quad \frac{\partial L}{\partial y} = 0.$$

Following usual terminology, a point $z = (y, s, x)$ is interior if $y > 0$ and $s > 0$. We do not require it to be primal or dual feasible. At such a point, we define the Newton direction $dz = (dy, ds, dx)$ by

$$\frac{\partial F}{\partial z} dz + F = 0. \tag{7}$$

Note that

$$\frac{\partial F}{\partial z} = \begin{pmatrix} S & Y & 0 \\ 0 & \frac{\partial^2 L}{\partial y \partial s} & \frac{\partial^2 L}{\partial y \partial x} \\ \frac{\partial^2 L}{\partial x \partial y} & 0 & \frac{\partial^2 L}{\partial x^2} \end{pmatrix},$$

with

$$\frac{\partial^2 L}{\partial y \partial s} = I, \quad \frac{\partial^2 L}{\partial y \partial x} = \frac{\partial g}{\partial x}, \quad \frac{\partial^2 L}{\partial x \partial y} = \left(\frac{\partial g}{\partial x}\right)^T, \quad \text{and} \quad \frac{\partial^2 L}{\partial x^2} = \sum_{i=1}^{m} y_i \frac{\partial^2 g_i}{\partial x^2}.$$

Since the g_i are convex, $\frac{\partial^2 L}{\partial x^2}$ is positive semi-definite. Let us make the further assumption

Assumption 0.3 *Let $y > 0$ and $s > 0$. The matrix*

$$H := \frac{\partial^2 L}{\partial x^2} + \frac{\partial^2 L}{\partial x \partial y} Y S^{-1} \frac{\partial^2 L}{\partial y \partial x}$$

is positive definite.

A sufficient condition for that is:

$$\frac{\partial^2 L}{\partial x^2} = \sum_{i=1}^{m} y_i \frac{\partial^2 g_i}{\partial x^2}$$

is positive definite, or $\frac{\partial g}{\partial x}$ has full row rank, or both.

Under Assumption 0.3, $\frac{\partial F}{\partial z}$ is regular at any interior point. Thus

$$dz = -\left(\frac{\partial F}{\partial z}\right)^{-1} F.$$

Let us explicitly write and solve the system (7) in dy, ds and dx:

$$\begin{aligned} S dy + Y ds + F_c &= 0 \\ ds + \frac{\partial^2 L}{\partial y \partial x} dx + F_p &= 0 \\ \frac{\partial^2 L}{\partial x \partial y} dy + \frac{\partial^2 L}{\partial x^2} dx + F_d &= 0. \end{aligned}$$

In these expressions we used the fact that $\frac{\partial^2 L}{\partial y \partial s} = I$.

The algorithm goes as follows: Given an interior, but not necessarily feasible, point, we compute the search direction dx associated with μ. Then a step is taken along that direction such that the interior property is maintained. Namely, let $\bar{\alpha} := \max\{\alpha : y + \alpha dy \geq 0, s + \alpha ds \geq 0\}$ and let $0 < \gamma < 1$. Then the next iterate is given by

$$\begin{aligned} x &:= x + \gamma\alpha dx \\ s &:= s + \gamma\alpha ds \\ y &:= y + \gamma\alpha dy. \end{aligned}$$

Te choice of μ is adaptive. For "normal" steps, we take $\mu = \frac{y^T s}{m^2}$. If $\min y_i s_i \leq \gamma \frac{y^T s}{m}$, the vector Ys is considered excessively unbalanced and we take $\mu = \frac{y^T s}{m}$. This step is named "centering".

We tested our algorithm on a sample of medium size random problems. We primarily studied the effect of varying the size of the problems. We observed that the number of iterations increases slowly with the number of constraints and, surprisingly enough, it decreases with the number of free variables in the case of quadratically constrained problems.

We analyzed the influence of centering and showed it to be positive. We also studied alternative strategies for the step size. It turns out that taking a fixed fraction of the maximal step size works well in practice. Moreover the fraction can be extremely close to 1 without any negative effect on the performance of the method. Finally, we looked at different choices for the starting point.

We applied this algorithm to linear programming problems. The algorithm behaves a bit differently than with quadratic constraints. The iteration count increases both with the number of constraints and the number of free variables. For the former the increase is slower. The figures are reasonable.

It Is Difficult to Find a Difficult Problem for Scheduling of Identical Parallel Machines

Béla Vizvári and Ramazan Demir
Bilkent University, 06533 Bilkent, Ankara, Turkey, VIZVARI@TRBILUN.BITNET

1 INTRODUCTION

Mathematical programming and theory of scheduling have a lot of optimization problems which are NP-hard in spite of their very simple structure. Thus these problems are considered to be difficult to solve. But some of them are easy in the sense that there are straightforward ways to generate feasible solutions of them, e.g. the knapsack problem, the TSP problem and many scheduling problems.

One of them is the scheduling of identical parallel machines where the maximal completion time has to be minimized. This problem is the topic of this experimental study. It has several heuristics. The two basic ones are Graham's list scheduling and the multi-fit algorithm. There are known upper bounds for the relative accuracy of the heuristic solutions provided by these methods. The two algorithms have quite different strategies. This is the reason that some problems worst from the point of view of list scheduling can be solved exactly by the multi-fit algorithm and vice versa. This gives the question that *how bad accuracy can have the better of the list scheduling and the multi-fit solutions.* This was the initial question of this research. Another algorithm called interchanging method has been also investigated. The research made necessary to sharpen the well-known lower bound of the optimal value of the objective function, too.

2 THE SCHEDULING PROBLEM

In the classical problem of the scheduling of parallel machines n jobs have to be distributed among m identical machines in such a way that the makespan is minimal.

The whole operation starts at time 0. The machine independent processing times are denoted by $p_j (j = 1, ..., n)$ which are positive integers. It is easy to see that there is at least one optimal solution such that the machines start to work at t=0 and are working without any idle time until all jobs assigned to them have been finished.

Let C_j be the completion time of job j. The maximal completion time, i.e. $\max\{C_j : j = 1, ..., n\}$, is denoted by C^*.

Theorem 1 *[Graham 69], [Coffman et al. 78] In any problem*

$$\begin{array}{c} \max\{\frac{1}{m}\sum_{j=1}^{n} p_j, \max\{p_j : j = 1, ..., n\}\} \\ \leq \ C^* \ \leq \\ \max\{\frac{2}{m}\sum_{j=1}^{n} p_j, \max\{p_j : j = 1, ..., n\}\}. \square \end{array} \quad (1)$$

The interval in which the optimal value must lie is denoted by $[L, U]$, i.e.

$$L = \lceil \max\{\frac{1}{m}\sum_{j=1}^{n} p_j, \max\{p_j : j = 1, ..., n\}\}\rceil \quad (2)$$

and

$$U = \lfloor \max\{\frac{2}{m}\sum_{j=1}^{n} p_j, \max\{p_j : j = 1, ..., n\}\}\rfloor. \quad (3)$$

Both the list scheduling and the multi-fit algorithm start with the determination of the nonincreasing order of the processing times. The two algorithms assign the jobs to machines in that order. Therefore without loss of generality it may be assumed that

$$p_1 \geq p_2 \geq \ldots \geq p_n \quad (4)$$

The rule of the list scheduling is that

every job is assigned to a machine having minimal current load. (LS)

Theorem 2 *[Graham 69] Let $C(LS)$ be the value of the solution provided by the list scheduling. Then*

$$\frac{C(LS)}{C^*} \leq \frac{4}{3} - \frac{1}{3m}.\square \quad (5)$$

Theorem 3 *[Graham 69] If there is an optimal solution which assigns to each machine at most 2 jobs, then the solution given by the list scheduling is optimal.* □

The multi-fit algorithm consists of two parts. A greedy method is the internal part and a logarithmic search is the external part which organizes the applications of the greedy method. For the internal part an upper bound K of the optimal value is assumed. The greedy method assigns each job to the first machine into it fits not exceeding the upper bound K. In the external part a current lower bound and a current upper bound are assumed and are denoted by lc and uc. For the internal part K is chosen as $\frac{lc+uc}{2}$. If the greedy method was able to find a solution not worst then K, then uc becomes $\lfloor K \rfloor$, otherwise $lc = \lceil K \rceil$. The process is repeated until the condition

$$uc = lc$$

is not satisfied. Notice that it follows from the assumption of the integrality of the processing times that the number of applications of the greedy method is $O(log(U - L))$. Thus the multi-fit algorithm is polynomial.

Theorem 4 *[Friesen 84] Let $C(MF)$ be the value of the solution provided by the multi-fit algorithm. Then*

$$\frac{C(MF)}{C^*} \leq 1.2\ \square \quad (6)$$

A third heuristic method called interchanging algorithm has been applied in this research. It makes the following steps starting from any solution. It interchanges one job of the most loaded machine with one job of another machine. The interchange is possible if and only if the maximal completion time is decreased in this way. Let s and t, resp., be indices of the most loaded and the other machines, resp. The current load of the machines are denoted by L_s and L_t. Suppose that the job i of machine s is interchanged with job j. Then the following two conditions must hold

$$p_i > p_j \tag{7}$$

and

$$L_t + p_i - p_j < L_s. \tag{8}$$

In the current version if a possible interchange is found then it has been executed. The order of checking Conditions (7) and (8) is as follows. The jobs of the most loaded machine are compared with the jobs of another machine taking the other machines in an increasing load order. The jobs of the two machines are taken in a decreasing processing time order. One job of the most loaded machine is compared with all of the jobs of the other machine. If no possible interchange is found then the next job of the most loaded machine is taken. The number of comparisons of one iteration are $O(n^2)$. To get a polynomial algorithm the number of interchanges has been limited by $m+2$. In the current version the solution provided by list scheduling is the starting point. This algorithm is one of simplest possible interchanging methods. In more general a subset of jobs can be interchanged for another subset of jobs. In that case the complexity of the selection of the two subsets is much higher.

3 IMPROVEMENTS OF THE LOWER BOUND

The randomly generated problems have not been solved with any kind of enumerative methods. One easy way to prove the optimality of a solution is that the value of it and the lower bound coincide. Therefore it was important to find some ways to improve the lower bound.

In (2) only two information are taken into consideration, the average load and the maximal processing time. The following two sharpening of the lower bound are based on the fact that what is the number of jobs which must be assigned to certain machines.

Theorem 5 *Assume that (4) holds. Then*

$$C^* \geq p_{n-\lceil \frac{n}{m} \rceil+1} + \ldots + p_n \ \square \tag{9}$$

Theorem 6 *Assume that (4) holds. Let*

$$r = n - \left\lfloor \frac{n}{m} \right\rfloor m$$

. *If* $r > 1$

$$C^* \geq \min\left\{\frac{\sum_{j=n-\lceil\frac{n}{m}\rceil r+1}^{n} p_j}{r}, \sum_{j=n-\lceil\frac{n}{m}\rceil}^{n} p_j\right\}. \quad \square \tag{10}$$

In some cases there are jobs which are not effecting C^*, because their processing times are relatively very small. In that cases the following observation is useful.

Theorem 7 *Let S be any subset of the jobs. Let L^S be any lower bound for the problem defined by the jobs those in S. Then L^S is a lower bound for the original problem.* $\square$

Theorem 8 *Let k be any index with $1 \leq k \leq n$. Assume that: (i) the list scheduling has assigned until that point exactly k jobs to machines, (ii) none of the machines has more than two jobs, (iii) $p_{k-2} + p_{k-1} + p_k$ is at least as great as the current load of any machine. Then the current maximal load is a lower bound for the optimal value of the problem.* $\square$

Theorem 9 *Let k be a fixed index and*

$$t_j = \left\lfloor \frac{p_j}{p_k} \right\rfloor \quad j = 1, \ldots, n.$$

Then

$$\left\lceil \frac{\sum_{j=1}^{n} t_j}{m} \right\rceil p_k \leq C^*.$$

4 COMPUTATIONAL EXPERIENCES

The computational experiences have been made in three phases. In the first phase about 500.000 problems belonging to different classes have been generated. In this phase some observations have been made which modified the objectives of the research. The second phase was the main one in which 1.200.000 problems have been generated in a wide range of problem classes to find difficult problems. Further attempts have been made to find more difficult problems in the most hopeful problem classes.

Definition 1 *Let $C(LS)$ and $C(MF)$ and $C(IC)$ and C^* be, resp., the value of the solution provided by the list scheduling and the multi-fit algorithm and the interchanging method and of the optimal solution, resp. A particular problem is called* first order difficult *if the value*

$$\frac{\min\{C(LS), C(MF)\}}{C^*} \tag{11}$$

is high. It is called second order difficult *if the value*

$$\frac{\min\{C(LS), C(IC), C(MF)\}}{C^*} = \frac{\min\{C(IC), C(MF)\}}{C^*} \tag{12}$$

is high.

This definition is not correct in a strict mathematical sense, because the meaning of the word "high" is undefined. This meaning has been determined during the experiences.

A problem class is determined by the following parameters: m - the number of machines, n - the number of jobs, p - the maximal possible processing time; the processing times are generated randomly by the $[1, p]$ integer uniform distribution.

In the experiences the following formulas have been used instead of (11) and (12)

$$\frac{\min\{C(LS), C(MF)\}}{\hat{L}} \tag{13}$$

and

$$\frac{\min\{C(IC), C(MF)\}}{\hat{L}} \tag{14}$$

where $\hat{L}$ is some lower bound of the optimal value of the objective function.

4.1 Observations of the First Phase

In the first phase only the list scheduling and the multi-fit algorithm have been used.

At the beginning of the experiences L has been chosen as $\hat{L}$. Some problems seemed to be difficult although an optimal solution has been obtained by one of the methods. In some cases this fact could be proven by one of the improvements of the lower bound discussed in Section 3.

Some problems had just the opposite behaviour. Here the lower bound coincided with the optimal value. In many cases this fact could be proven by the interchanging algorithm. This is the reason that this method had to be involved into the investigations.

Among the most difficult problems found in this phase there were many such that the smallest processing time was relatively great. Therefore in the second phase of the experiences the generation of the the problems has been modified as follows. The first thousand problems has been generated as earlier. In the case of the problems of the second thousand the processing times were increased by 1, in the case of the third thousand by 2, e.t.c. This cannot be applied for all of the classes, because in some cases if the increase is not less than a certain value, the problem regardless the generated random numbers becomes trivial.

The problems which seemed to be difficult were belonging to two different categories. The first one is the set of first order difficult problems. The most difficult problem in this sense was the following. $n = 10$, $m = 3$ and the processing times are 30, 29, 24, 18, 17, 17, 17, 14, 13, 13. The solution provided by the list scheduling is as follows: M1: 30, 17, 13; M2: 29, 17, 14; M3: 24, 18, 17, 13. The multi-fit solution is: M1: 30, 29, 13; M2: 24, 18, 17, 13; M3: 17, 17, 14. Both of them have the value 72. But the optimal solution is the following: M1: 30, 17, 17; M2: 29, 18,

17; M3: 24, 14, 13, 13. The value of it is 64. Since that time Definition 1 has had the meaning that a problem is first order difficult if

$$\frac{\min\{C(LS), C(MF)\}}{C^*} > \frac{9}{8}.$$

The computationally difficult problems belong to the second category. In the case of such a problem it is difficult either to find the optimal solution or to prove the optimality of a solution generated by one of the heuristics.

4.2 Experiences of the Main Phase

In the second phase an intensive search has been carried out for difficult problems. 100.000 problems have been generated in each of the problem classes. The generated solutions are within 105% and even 101% of the improved lower bound in the case of a very great part of the problems in each class. These results are summarized in Table 1.

It turned out that none of the list scheduling and the multi-fit algorithm is superior to the other one. This is indicated by the numbers of problems such that the appropriate heuristic solution is within 101%. The number of problem classes for which a method is superior to the other one is approximately is the same for both algorithms. The behaviour of both methods are very different in the different classes. But the "the better of list scheduling and multi-fit" seems to be much stable.

n/m/p	101% LS-MF	101% IC-MF	105% LS-MF	105% IC-MF
10/3/15	88067	94179	98817	99897
15/3/15	95177	99441	99997	99999
10/3/30	73970	85831	97418	99815
15/3/30	89662	99002	99999	100000
10/3/60	45787	69949	94304	99582
15/3/60	80167	98599	99996	100000
30/3/15	99910	100000	100000	100000
30/3/30	99917	100000	100000	100000
30/3/60	99132	100000	100000	100000
10/5/15	98244	98245	99043	99044
20/5/15	89275	97461	99995	100000
60/5/60	100000	100000	100000	100000
$\sum$	960051	1043707	1089569	1098337
percentage	87.27	94.88	99.05	99.85

Table 1: The numbers of problems having good heuristic solution

parameters	MF	LS
10/3/15	87795	1318
15/3/15	7324	93662
10/3/30	73529	950
15/3/30	12688	86078
10/3/60	44898	1568
15/3/60	23804	72245
30/3/15	26577	99725
30/3/30	50626	99591
30/3/60	42090	99132
10/5/15	98236	97997
20/5/15	3649	87973
60/5/60	100000	99111

Table 2: Comparison of the list scheduling and multi-fit heuristics

The most first order difficult problem which has been found in this phase is the following. $n = 10$, $m = 3$ and the processing times are 15, 14, 12, 9, 8, 8, 8, 7, 6, 6. The solution provided by the list scheduling is as follows: M1: 15, 8, 6, 6; M2: 14, 8, 7; M3: 12, 9, 8. The multi-fit solution is: M1: 15, 14, 6; M2: 12, 9, 8, 6; M3: 8, 8, 7. Both of them have the value 35. But the optimal solution is the following: M1: 15, 8, 8; M2: 14, 9, 8; M3: 12, 7, 6, 6. The value of it is 31.

4.3 Further difficult Problems

The aim of the third phase has been to find further difficult problems. Some new problem classes are introduced, because it is likely on the basis of the previous experiences that these classes contain the desired items. At the end of this phase the number of the generated problems have exceeded 2.000.000.

The class 19/8/15 contained the known most difficult problem. The processing times of it are: 21, 21, 20, 20, 19, 18, 17, 17, 16, 16, 16, 16, 12, 12, 12, 11, 11, 10, 10. The multi-fit solution is: M1: 21, 21; M2: 20, 20; M3: 19, 18; M4: 17, 17; M5: 16, 16, 10; M6: 16, 16, 10; M7: 12, 12, 12; M8: 11, 11. The value of it is 42, which is achieved at M1 and M5 and M6. The solution provided by the list scheduling with value 43 is this: M1: 21, 11, 11; M2: 21, 12; M3: 20, 12, 10; M4: 20, 12, 10; M5: 19, 16; M6: 18, 16; M7: 17, 16; M8: 17, 16; In the optima; solution the completion time is 37 on all of the machines except the last one where it is 36: M1: 21, 16; M2: 21, 16; M3: 20, 17; M4: 20, 17; M5: 19, 18; M6: 16, 11, 10; M7: 16, 11, 10; M8: 12, 12, 12.

The development of the accuracy of the most known first order difficult problems has been: $\frac{9}{8} < \frac{35}{31} < \frac{42}{37}$. The value 42/37, which in not proven to be an upper bound, is less than the value 72/61 guaranteed by the algorithm of [Friesen-Langston 86], which uses many operations from a practical point of view.

There was no improvement in the position of most second order difficult problem in this phase.

4.4 Some Other Observations

Some other observations are obtained from the experiences. An important one is the following. If $L \neq U$ then U is far from the optimal value. The 10/5/15 class is the only one where the ratio (13) had a value greater then 1.22. The observed greatest value is 1.42.

The aim of the improvements of the lower bound was to decrease the number of cases to check. In Table 5 the the number of problems which have been proved to be solved within 101%, and the observes worst (14) ratio observed before any improving and after improving (without the application of Theorem 9) are provided for the better of multi-fit and interchanging procedure.

parameters	101%		(14)	
	before	after	before	after
10/3/15	88078	89871	1.217	1.120
15/3/15	99344	99344	1.030	1.030
10/3/30	65089	68313	1.262	1.102
15/3/30	99098	99098	1.032	1,032
10/3/60	44046	71572	1.211	1.100
15/3/60	99045	99045	1.032	1.032
30/3/15	100000	100000	1.005	1.005
30/3/30	100000	100000	1.007	1.007
30/3/60	100000	100000	1.005	1.005
10/5/15	37056	88469	1.412	1.200
20/5/15	97240	97240	1.040	1.040
60/5/60	100000	100000	100000	100000

Table 3: The effect of the improvements of the lower bound

References

[Coffman et al. 78] Coffman, E.G.Jr., Garey, M.R., Johnson, D.S., An application of bin-packing to multiprocessor scheduling, SIAM J. Comput., 7(1978) 1-17.

[Friesen 84] Friesen, D.K., Tighter bounds for the multifit processor scheduling algorithm, SIAM J. Comput., 13(1984), 170-181.

[Friesen-Langston 86] Friesen, D.K., Langston, M.A., Evaluation of a MULTIFIT-based scheduling algorithm, J. Algorithms, 7(1986), 35-59.

[Graham 69] Graham, R.L., Bounds on multiprocessing timing anomalies, SIAM J. Appl. Math. 17(1969) 416-429.

Concepts for Parallel Tabu Search

Stefan Voß
Technische Hochschule Darmstadt, FB 1 / FG Operations Research,
Hochschulstraße 1, D - 6100 Darmstadt, Germany

Abstract

Tabu Search is a metastrategy for guiding known heuristics to overcome local optimality. Successful applications of this kind of metaheuristic to a great variety of problems have been reported in the literature. Recently some implementations of tabu search on parallel computers have come up. Whereas these implementations are tailored to specific problems we attempt to provide ideas for a more general concept for developing parallel tabu search algorithms.

1 Introduction

Due to the complexity of a great variety of combinatorial optimization problems, heuristic algorithms are especially relevant for dealing with large scale problems. The main drawback of algorithms such as deterministic exchange procedures is their inability to continue the search upon becoming trapped in a local optimum. This suggests consideration of recent techniques for guiding known heuristics to overcome local optimality. Following this theme, the application of the *tabu search* metastrategy for solving combinatorial optimization problems is investigated.

The key issue in designing parallel algorithms is to decompose the execution of the various ingredients of a procedure into processes executable by parallel processors. Improvement procedures like tabu search or simulated annealing at first glance, however, have an intrinsic sequential nature due to the idea of performing the neighbourhood search from one solution to the next. Therefore, there is not yet a common or generally applicable parallelization of tabu search in the literature. In the sequel we attempt to describe some general ideas and a classification scheme for parallel tabu search algorithms.

In Section 2, we present an outline of tabu search. Before describing some concepts for parallel tabu search algorithms in more detail (see Section 4), we briefly discuss some of the common parallel machine models and algorithms in Section 3. Some examples are given in Section 5 and finally some conclusions are drawn (Section 6). The attempt, of course, is not to give a complete treatment of parallel tabu search but to sketch the potential this area of research carries. For a more detailed treatment of the ideas of this paper see Voß (1992).

2 Tabu Search

Many solution approaches are characterized by identifying a neighbourhood of a given solution which contains other (*transformed*) solutions that can be reached in a single iteration. A transition from a feasible solution to a transformed feasible solution is referred to as a *move* and may be described by a set of one or more *attributes*. In a zero-one integer programming context, e.g., these attributes may be the set of all possible value assignments or changes in such assignments for the binary variables. (Then two attributes denoting that a certain binary variable is set to 1 or 0, may be called *complementary* to each other.) Following a steepest descent/mildest ascent approach, a move may either result in a best possible improvement or a least deterioration of the objective function value. Without additional control, however, such a process can cause a locally optimal solution to be re-visited immediately after moving to a neighbour.

To prevent the search from endlessly cycling between the same solutions, tabu search may be visualized as follows. Imagine that the attributes of all moves are stored in a *running list*, representing the trajectory of solutions encountered. Then, related to a sublist of the running list a so-called *tabu list* may be defined. Based on certain restrictions, it keeps some moves, consisting of attributes complementary to those of the running list, which will be forbidden in at least one subsequent iteration because they might lead back to a previously visited solution. Thus, the tabu list restricts the search to a subset of admissible moves (consisting of admissible attributes or combinations of attributes). This hopefully leads to 'good' moves in each iteration without re-visiting solutions already encountered. A general outline of a tabu search procedure (for solving a minimization problem) may be described as follows:

Tabu Search

Given: A feasible solution x^* with objective function value z^*.
Start: Let $x := x^*$ with $z(x) = z^*$.
Iteration:
while stopping criterion is not fulfilled[1] **do begin**

(1) select best admissible move that transforms x into x' with objective function value $z(x')$ and add its attributes to the running list
(2) perform tabu list management: compute moves to be set tabu, i.e., update the tabu list
(3) perform exchanges: $x := x', z(x) = z(x')$
if $z(x) < z^*$ **then** $z^* := z(x), x* := x$ **endif**

endwhile
Result: x^* is the best of all determined solutions, with objective function value z^*.

* * *

For a background on tabu search and a number of references on successful applications of this metaheuristic see, e.g., Glover (1989, 1990), Glover and Laguna (1992), and Voß (1992).

[1] A possible stopping criterion can be, e.g., a prespecified time limit.

Tabu List Management

Tabu list management concerns updating the tabu list, i.e., deciding on how many and which moves have to be set tabu within any iteration of the search. Up to now, the most popular approach in literature is to apply static methods like the *tabu navigation method* (TNM).

In TNM, single attributes are set tabu as soon as their complements have been part of a selected move. The attributes stay tabu for a distinct time, i.e. number of iterations, until the probability of causing a solution's re-visit is small. The efficiency of the algorithm depends on the choice of the tabu status duration, i.e. the length tl_size of the tabu list. (In the literature often a 'magic' tl_size=7 is proposed.) For the sake of an improved effectivity, a so-called *aspiration level criterion* is considered, which permits the choice of an attribute even when it is tabu. This can be advantageous when a new best solution may be calculated, or when the tabu status of the attributes prevent any move from feasibility.

The static approach, though successful in a great number of applications, seems to be a rather limited one. Another probably more fruitful idea is to define an attribute as being *potentially tabu* if it belongs to a chosen move and to handle it in a *candidate list* first. Via additional criteria these attributes can be definitely included in the tabu list if necessary, or excluded from the candidate list if possible. Therefore, the candidate list is an intermediate list between a running list and a tabu list. Glover (1990) suggests the use of different candidate list strategies in order to avoid extensive computational effort without sacrificing solution quality. In the sequel, we sketch the following dynamic strategies for managing tabu lists: the *cancellation sequence method* (CSM, in a revised version, cf. Dammeyer et al. (1991)), and the *reverse elimination method* (REM).

CSM as well as REM both use additional criteria for setting attributes tabu. The primary goal is to permit the reversion of any attribute but one between two solutions to prevent from re-visiting the older one. To find those *critical* moves, CSM needs a candidate list that contains the complements of attributes being potentially tabu. This *active tabu list* (ATL) is built like the running list where elimination of certain attributes is furthermore permitted. Whenever an attribute of the last performed move finds its complement on ATL this complement will be eliminated from ATL. All attributes between the cancelled one and its recently added complement build a *cancellation sequence* separating the actual solution from the solution that has been left by the move that contains the cancelled attribute. Any attribute but one of a cancellation sequence is allowed to be cancelled by future moves. This condition is sufficient but not necessary, as some additional aspects have to be taken into account so that CSM works well.

The method works well for the case that a move consists of exactly one attribute, i.e., when so-called *single-attribute moves* are considered instead of *multi-attribute moves*. In addition, the corresponding parameters have to be chosen appropriately (e.g. the tabu list duration of a tabu attribute, and how to apply the aspiration level criterion). Applying CSM to multi-attribute moves needs additional criteria to prevent errors caused by uncovered special cases. E.g. for *paired-attribute moves* (moves consisting of exactly two attributes) those moves must be prohibited that may cancel a cancellation sequence consisting of exactly two attributes (because none of them is tabu when choosing a move). In addition, for building a cancellation sequence, the remaining attributes of the older and the current move are not necessarily taken into consideration. This depends on the

order in which the move's attributes are added to ATL.

The conditions of TNM and CSM need not be necessary to prevent from re-visiting previously encountered solutions. Necessity, however, can be achieved by REM. The idea of REM is that any solution can only be re-visited in the next iteration if it is a neighbour of the current solution. Therefore, in each iteration the running list will be traced back to determine all moves which have to be set tabu (since they would lead to an already explored solution). For this purpose, a *residual cancellation sequence* (RCS) is built up stepwise by tracing back the running list. In each step exactly one attribute is processed, from last to first. After initializing an empty RCS, only those attributes are added whose complements are not in the sequence. Otherwise their complements in the RCS are eliminated (i.e. cancelled). Then at each tracing step it is known which attributes have to be reversed in order to turn the current solution back into one examined at an earlier iteration of the search. If the remaining attributes in the RCS can be reversed by exactly one move then this move is tabu in the next iteration. For single-attribute moves, for instance, the length of an RCS must be one to enforce a tabu move. Correspondingly, in a slightly modified method REM2 all common neighbours of the current solution and of an already explored one will be forbidden. These neighbours were implicitly investigated during a former step of the procedure (due to the choice of a best non-tabu neighbour) and need not be looked at again (cf. Voß (1992)).

Obviously, the execution of REM and of REM2 represents a necessary and sufficient criterion to prevent from re-visiting known solutions. Since the computational effort of REM increases if the number of iterations increases, ideas for reducing the number of computations have been developed (cf. Glover (1990) and Dammeyer and Voß (1991a)).

For applications and (sequential) comparisons of TNM, CSM, and REM see Dammeyer and Voß (1991b) and Domschke et al. (1992).

Search Intensification and Search Diversification

A general idea for reducing the computational effort in a tabu search algorithm is that of search intensification using a so-called *short term memory*. Its basic idea is to observe the attributes of all performed moves and to eliminate those from further consideration that have not been part of any solution generated during a given number of iterations. This results in a concentration of the search where the number of neighbourhood solutions in each iteration, and consequently the computational effort, decreases. Obviously the cost of this reduction can be a loss of accuracy.

Correspondingly, a search diversification may be defined as a *long term memory* to penalize often selected assignments. Then the neighbourhood search can be led into not yet explored regions where the tabu list operation is restarted (resulting in an increased computation time). An appealing opportunity for search diversification is created by the idea of REM and REM2 resulting in REMt for $t > 2$ and integer. If at any tracing step the attributes that have to be reversed to turn the current solution back into an already explored one equal exactly t moves then it is possible to set these moves tabu for the next iteration. Note that for the case of multi-attribute moves, due to various combinations of attributes to moves, even more than t moves may be set tabu in order to avoid different paths through the search space leading to the same solution. Accordingly, search diversification is obvious.

3 Parallel Machine Models

Over the years a great variety of architectures have been proposed for parallel computing. The most widely known classification of parallel machine models (although somehow limited) is given by Flynn (1966). He distinguishes four general classes based on the idea of whether single or multiple instruction streams are executed on either one or multiple data set streams:

- **SISD** *(Single Instruction, Single Data)* including the classical sequential computers
- **SIMD** *(Single Instruction, Multiple Data)* including vector computers and array processors
- **MISD** *(Multiple Instructions, Single Data)*
- **MIMD** *(Multiple Instructions, Multiple Data)* with the processors performing each successive set of instructions either simultaneously (synchronous) or independently (asynchronous)

The above classification of parallel machine models may lead to different classes of parallel algorithms. *Vectorized algorithms* operate uniformly on vectors of data sets (SIMD). *Systolic* ones operate rhythmically on streams of data sets (SIMD and synchronous MIMD). *Parallel processing algorithms* operate on a set of synchronously communicating parallel processors (synchronous MIMD). Correspondingly, asynchronous communication leads to *distributed processing algorithms* (asynchronous MIMD and neural networks).

In addition to architectural aspects communication networks are used to classify parallel machine models. For instance, it makes a difference whether processors have simultaneous access to a shared memory, allowing communication between two arbitrary processors in constant time, or whether they communicate through a fixed interconnection network. Less formally, in certain models it is assumed that there is a *master* processor controlling the communication of the network, with the remaining processors of the network called *slaves*. For a comprehensive survey on parallel machines and algorithms see e.g. Akl (1989) and Van Leeuwen (1990).

The quality of parallel algorithms may be judged by a number of quantities, the most important one being the *speedup*, which is the running time of the best sequential implementation of the algorithm divided by the running time of the parallel implementation executed on a number of p processors. Similarly, given a prespecified time limit (cf. footnote 1) a *scaleup* may be defined as the ratio of the average problem sizes solvable with a parallel implementation to a sequential implementation of the algorithm. With heuristics, the solution quality attainable may also be measured. The *processor utilization* or *efficiency* is the speedup divided by p. The best one can achieve is a speedup of p and an efficiency equal to one.

4 Parallel Tabu Search Algorithms

Due to the success and the underlying simplicity of the main idea of tabu search, recently some implementations on parallel computers have come up tailored to specific problems.

Surprisingly, to the best of our knowledge, they are solely devoted to problems using the notion of paired-attribute moves: the travelling salesman problem, the job shop problem, and the quadratic assignment problem (compare Section 5).

In a first step we shall describe a *classification of different types of parallelism* that is applicable to most iterative search techniques (cf. Voß (1992)). Its basis is the idea of having different starting solutions or candidate solutions (so-called balls, motivated by the idea of mountains' like solution space where a ball is rolling to find a stable low altitude state) as well as a number of different strategies, e.g. based on various possibilities of the parameter setting or on the tabu list management.

- **SBSS** *(Single Ball, Single Strategy)*

  The algorithm starts from exactly one given feasible solution and performs its moves following exactly one strategy.

- **SBMS** *(Single Ball, Multiple Strategies)*

  The algorithm starts from exactly one given feasible solution by the use of different strategies where each strategy is performed on a different processor.

- **MBSS** *(Multiple Balls, Single Strategy)*

  The algorithm starts from different initial feasible solutions, each on a different processor. The same type of instruction, i.e. strategy, is performed on each processor.

- **MBMS** *(Multiple Balls, Multiple Strategies)*

  The algorithm starts from different initial feasible solutions performing different strategies.

In what follows we discuss the above ideas in more detail with special emphasis on further principles of parallelism within specific strategies. For ease of description we assume the notion of parallel or distributed processing algorithms.

SBSS

The single ball, single strategy idea is the simplest version, and obviously corresponds to the idea of classical sequential computations (cf. the SISD-model). This, however, does not restrict the possibility of parallelization.

Starting from an initial feasible solution, the best move which is not tabu must be performed. The search for this move may be done in parallel by decomposing the set of admissible moves into a number of subsets. E.g. in a master-slave architecture each (slave) processor may evaluate the best move in a specific subset. The best move of each subset is communicated to the master who picks the overall best as the transformed solution and also performs the tabu list management.

To restrict the amount of communication necessary for synchronizing the data each slave could determine the best possible move in its subset without observing any tabu list, while the tabu list in the same time is updated by the master. Then the master picks among all answers the best which is not tabu. If no such move exists, a second trial must be made while each processor has to receive and to observe the tabu list. Otherwise the next iteration is to be performed.

Additional ideas may be developed with respect to the specific strategies. In TNM, the tabu list management may be done by each processor itself by simply providing the most recent move (whose complement will be in the list). In CSM, the master builds the cancellation sequences and partitions them to the slaves, i.e., every slave has to evaluate a certain number of sequences. In subsequent iterations, the attributes of the current moves are communicated. Whenever a cancellation sequence is reduced to 1 it will be re-communicated to the master.

SBMS

In SBMS each processor executes a process which is one of the above tabu search strategies with different tabu conditions and parameters, like e.g. REMt for various t. For TNM this can be different (eventually randomly modified) tabu list lengths; for CSM, different tabu durations may be considered. The (slave) processors are halted after a prespecified time and the results are compared and the best one is calculated. A restart is possible with the best or a good seed solution. Each strategy may take a different path through the search space because of different tabu list management or parameter setting. A restart may be performed either with empty running and tabu lists or with a previously encountered list.

MBSS

The multiple balls approaches start from at most p (the number of processors available) *different initial feasible solutions*, whose calculation can vary. They may be determined either randomly or by applying different heuristics to the same problem. This may also incorporate ideas involving different diversification and intensification strategies as described above. A third possibility assumes one given feasible solution and starts with a suitable subset of its transformed (neighbourhood) solutions. (Especially with REM2 it may be assured that even in future iterations there is no overlap with the initial feasible solutions of the other processors.) The single strategy approach assumes the application of exactly one tabu search algorithm with the same parameter setting for all processors.

As with SBMS, the processes may be halted after a specific time period to coordinate their results and possibly to initiate a restart with new (hopefully) improved solutions. If the processes are performed synchronously, then the stopping may be initiated after having generated, say, m successive moves. On synchronous MIMD machines the latter approach may be especially relevant. Note that the above-mentioned possibility of parallelization within SBSS is related to a method with $m = 1$ where the best transition is evaluated.[2] With respect to MBSS, this modifies to the evaluation of the p best moves usable for a restart. For $m \geq 2$ this approach may be used as a *look ahead method*.

MBMS

The multiple balls, multiple strategies approach subsumes all previous classes, allowing search within the solution space from different starting points with different methods or parameter settings.

[2]This gives reference to incorporate different candidate list strategies. (Note the correspondance to ideas of beam search, cf. Glover (1990).)

5 Examples

In the sequel we sketch some of the ideas given in the previous sections with respect to well known combinatorial optimization problems. As mentioned above, we only found some work on problems with the idea of paired-attribute moves to perform the neighbourhood search. We start with respect to binary integer programming, exploiting single-attribute moves.

Consider the SBSS concept. Also consider n decision variables in a binary problem with no (implicit or explicit) restriction on the number of variables set to either 1 or 0. We may define simple ADD- or DROP-moves by complementing the corresponding entries of the binary variables x_i. Assume the existence of $n+2$ processors with $n+2$ being the master processor. The tabu list management is performed by processor $n+1$. In any iteration of the search, each of the synchronously controlled processors $i \in \{1,\ldots,n\}$ receives the information whose variables' entry has been chosen to be exchanged as the most recent move. This move is performed together with the reversion of x_i. This usually can be done quite efficiently by reconstructing the previous solution stored at i with at most one assignment complemented. Then i offers its objective function value to the master who re-calls all results of processors referring to non-tabu moves (evaluated by processor $n+1$). Obviously this approach may be generalized in various ways to the more general classes described above.

This concept may be applied, e.g., to the warehouse location problem (WLP), to Steiner's problem in graphs (SP), and to the multiconstraint zero-one knapsack problem (MCKP). E.g., for WLP this neighbourhood search means a reallocation of costumers, i.e., opening a new location i results in re-allocating all costumers for which i is closer than the depot currently used. Correspondingly, closing a location i forces all costumers receiving service from i to its second nearest location.

An even more challenging reoptimization problem arises within SP. There, an iteration of the neighbourhood search may consist of changing a node-oriented binary variable and calculating a minimum spanning tree (MST) on the set of all nodes with entry 1 of the corresponding variables. The question is, whether reoptimization may be carried out either by solving the modified problem anew or by starting from a previous optimal solution found by the same processor (see Glover et al. (1992) for a corresponding sequential approach with respect to MST).

If the number or weighted number of variables with value 1 is limited (as for MCKP) or fixed (as e.g. in the p-median problem) then the same approach may be applied with combined ADD/DROP- or SWAP-moves leading to paired-attribute moves.

Malek et al. (1989) follow the SBMS approach to solve travelling salesman problems (TSP) by TNM with the 2-opt exchange as moves. The tabu attributes follow different strategies in that they are restricted either to one or to the two cities that have been swapped or to the cities and their respective positions in tour. In addition different tabu parameters were used on different processors. For another parallel tabu search algorithm for the TSP see Fiechter (1990).

The quadratic assignment problem (QAP) is treated by Chakrapani and Skorin-Kapov (1991, 1992) by the use of SBSS and TNM with search intensification and search diversification performed sequentially while evaluating the moves in parallel. The set of moves is partitioned into disjoint subsets, each one on a different processor as described above.

The neighbourhood search is performed by pairwise interchanges such that for $O(n^2)$ processors available all moves can be evaluated in constant time, achieving a speedup of $O(n^2/\log n)$. Battiti and Tecchiolli (1992) use TNM together with a hashing function and compare their algorithm also with a parallel genetic algorithm. Another parallel algorithm for QAP based on TNM (with randomly varying tl_size) has been presented by Taillard (1991). It is an SBSS approach, too. The same idea has also been applied to the job shop as well as to the flow shop problem (see Taillard (1989, 1990)). The latter, in fact, also describes a single-attribute based implementation with attributes corresponding to objective function values. Chakrapani and Skorin-Kapov (1992) is especially relevant since its implementation is based on a connectionist approach related to a Boltzmann machine (cf. Aarts and Korst (1989)).

6 Conclusions

We have summarized some ideas for developing parallel tabu search algorithms. Motivated by a famous classification scheme for parallel machine models we proposed a classification scheme for parallel tabu search algorithms. While research in this field is still in its infancy we believe that reasonable achievements in the following two aspects will be provided.

- Development of a framework for a general parallel tabu search algorithm that can be applied to a wide range of combinatorial optimization problems.
- Empirical results for parallel tabu search algorithms tailored to specific problems.

Some results known from the literature (cf. Section 5) support this feeling. Despite the emphasis on parallel tabu search, sequential testing is still far from complete. In addition, the tabu search metastrategy should be tested on different classes of parallel algorithms and machine models. Especially relevant seems to be a comparison of algorithms tailored to different hardware specifications like vector computers versus synchronous and asynchronous MIMD machines. However, one should take into account identical user specifications with respect to tabu search (e.g. parameter setting, definition of the neighbourhood). Note that our classification scheme is not restricted to parallel tabu search, but may be applied for nearly any iterative search procedure, such as simulated annealing or genetic algorithms.

References

[1] E. Aarts and J. Korst (1989), *Simulated Annealing and Boltzmann Machines* (Wiley Chichester).

[2] S.G. Akl (1989), *The Design and Analysis of Parallel Algorithms* (Prentice-Hall, Englewood Cliffs).

[3] R. Battiti and G. Tecchiolli (1992), Parallel biased search for combinatorial optimization: genetic algorithms and tabu. *Microprocessors and Microsystems*, to appear.

[4] J. Chakrapani and J. Skorin-Kapov (1991), Massively parallel tabu search for the quadratic assignment problem. *Annals of Operations Research*, to appear.

[5] J. Chakrapani and J. Skorin-Kapov (1992), A connectionist approach to the quadratic assignment problem. *Computers & Operations Research* **19**, 287-295.

[6] F. Dammeyer, P. Forst and S. Voß (1991), On the cancellation sequence method of tabu search. *ORSA Journal on Computing* **3**, 262-265.

[7] F. Dammeyer and S. Voß (1991a), Dynamic tabu list management using the reverse elimination method. *Annals of Operations Research*, to appear.

[8] F. Dammeyer and S. Voß (1991b), Application of tabu search strategies for solving multiconstraint zero-one knapsack problems. Working paper TH Darmstadt.

[9] W. Domschke, P. Forst and S. Voß (1992), Tabu search techniques for the quadratic semi-assignment problem. In: G. Fandel, T. Gulledge and A. Jones (eds.), *New Directions for Operations Research in Manufacturing* (Springer, Berlin), 389-405.

[10] C.-N. Fiechter (1990), A parallel tabu search algorithm for large traveling salesman problems. Paper presented at 1st Int. Workshop on Project Management and Scheduling, Compiegne.

[11] M.J. Flynn (1966), Very high-speed computing systems. *Proc. IEEE* **54**, 1901-1909.

[12] F. Glover (1989), Tabu search - part I. *ORSA Journal on Computing* **1**, 190-206.

[13] F. Glover (1990), Tabu search - part II. *ORSA Journal on Computing* **2**, 4-32.

[14] F. Glover, D. Klingman, R. Krishnan and R. Padman (1992), An in-depth empirical investigation of non-greedy approaches for the minimum spanning tree problem. *European Journal of Operational Research* **56**, 343-356.

[15] F. Glover and M. Laguna (1992), Tabu search. Working Paper Univ. of Colorado at Boulder, to appear.

[16] M. Malek, M. Guruswamy, M. Pandya and H. Owens (1989), Serial and parallel simulated annealing and tabu search algorithms for the traveling salesman problem. *Annals of Operations Research* **21**, 59-84.

[17] E. Taillard (1989), Parallel taboo search technique for the jobshop scheduling problem. Working Paper Ecole Polytechnique Federale de Lausanne.

[18] E. Taillard (1990), Some efficient heuristic methods for the flow shop sequencing problem. *European Journal of Operational Research* **47**, 65-74.

[19] E. Taillard (1991), Robust taboo search for the quadratic assignment problem. *Parallel Computing* **17**, 443-455.

[20] J. Van Leeuwen (1990), *Algorithms and Complexity* (Elsevier, Amsterdam).

[21] S. Voß (1992), Tabu search: Applications and prospects. Working Paper TH Darmstadt, to appear.

A Dual Strategy for Solving the Linear Programming Relaxation of a Driver Scheduling System

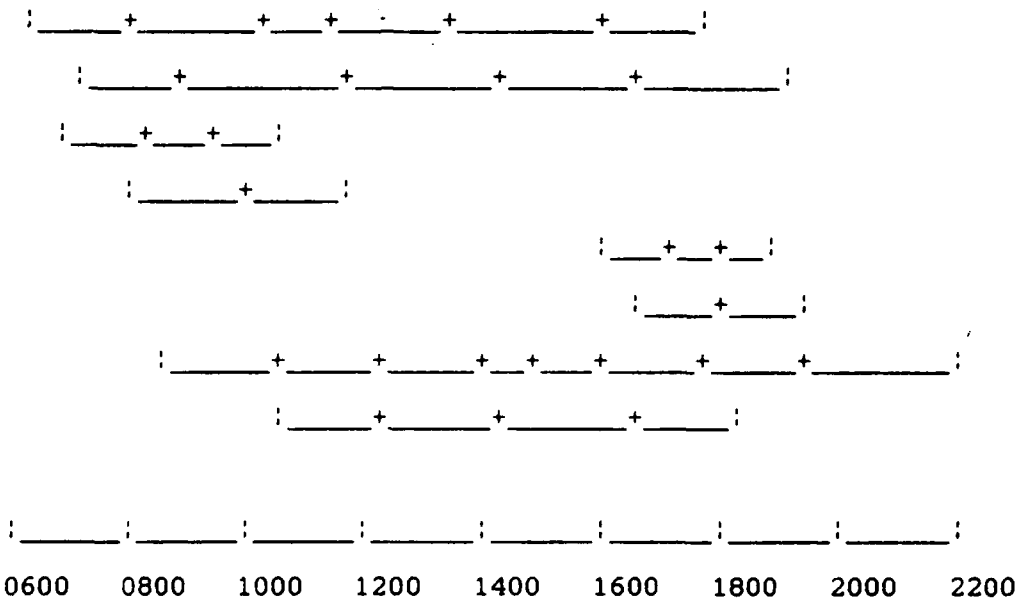
Phil Willers*, Les Proll** and Anthony Wren**

* School of Computing and Mathematics
University of Teesside
Middlesbrough
Cleveland TS1 3BA
United Kingdom

** School of Computer Studies
University of Leeds
Leeds LS2 9JT
United Kingdom

The work of a transport company (bus, train, etc.) may be represented by a *schedule* which specifies the journeys to be undertaken. Figure 1 is a graphical representation of part of such a schedule, with each line showing the times that a service begins and ends, and each '+' showing the time of a *relief opportunity* at which the driver of that service may be replaced by another driver. An indivisible period which must be worked by the same driver (e.g. between two consecutive relief opportunities) is called a *workpiece*.

Figure 1 - Graphical Representation of a Schedule



Each driver's working day consists of a number of workpieces. A complete specification of a driver's working day, including sign-on, sign-off and mealbreak times, is called a *duty*. Every transport company has many conditions that its duties must satisfy, usually called the "union agreement". This agreement may specify, for example, the maximum length of a working day and durations of mealbreaks. There is usually a very large number of different duties that could be used to cover a schedule.

There are several computer systems which can be used to determine a set of valid drivers' duties to cover a schedule provided by a transport company. This paper will consider enhancements that have recently been devised for one such system called IMPACS (Integer Mathematical Programming for Automatic Crew Scheduling). This system was developed at the University of Leeds by Wren & Smith[1] and is now marketed by the Hoskyns Group. IMPACS has mainly been used by bus companies (throughout the world) but it has also been used by train and tram companies.

At the heart of the IMPACS system is an Integer Programming model which has two pre-emptively ordered objectives: to minimise the total number of duties used to cover a given schedule and to minimise a cost function which reflects both the wage cost and undesirable features of duties. The model's constraints ensure that all workpieces are covered at least once, with some specially selected workpieces being covered exactly once. Also, each duty is classified according to its type (e.g. early, late, overtime) and side constraints can be added which limit the number of duties of any type that are to be used.

Thus, the model is of the mixed set covering/partitioning type, possibly with the addition of side constraints. Ongoing research attempts to exploit further the special structure of the IMPACS model and to take advantage of recent developments in mathematical programming algorithms.

The IMPACS model has previously been solved using the following four-stage process. For the first three stages, the Linear Programming relaxation of the model is used.

Stage 1 Minimise the total number of duties using a Primal Simplex algorithm.

Stage 2 Add a constraint which ensures that the integral number of duties does not increase and minimise the cost function using a Primal Simplex algorithm.

Stage 3 If the total number of duties is not integral, add a suitable constraint, and reoptimise using a Dual Simplex algorithm.

Stage 4 Determine an integer solution using Branch and Bound techniques with constraint branching.

Optimisation within the IMPACS system is based on Ryan's ZIP package[2]. The performance of this package has been improved by incorporating Goldfarb & Reid's Primal Steepest Edge algorithm[3] and a Dual Steepest Edge algorithm due to Forrest & Goldfarb[4].

This paper will consider a new strategy for solving the Linear Programming relaxation. Enhancements to stage 4 are the subject of separate work.

Each of stages 1 and 2 of the previous strategy typically involve a large number of iterations, resulting in the time to solve the Linear Programming relaxation being a significant proportion of the total solution time. This is due to the objectives for stages 1 and 2 being different and the high degree of degeneracy inherent in the model. Also, the constraint that is added at stage 2 is fully dense, and this substantially increases iteration timings.

These difficulties have been addressed by:

1. Using a single weighted objective function.

and 2. Solving the resulting model using a Dual Steepest Edge algorithm.

The weight that is used to combine the two objectives is relatively small, and is determined by applying an algorithm due to Sherali[5] to the IMPACS model. To initiate the Dual Simplex algorithm, an heuristic has been developed to produce initial basic dual feasible solutions.

The paper will conclude with the presentation of computational results for real world problems with numbers of constraints in the range 125 to 450 and numbers of variables in the range 4000 to 11000. The results suggest that the new strategy significantly reduces solution timings.

References

1. Wren A & Smith B M (1988) Experiences with a Crew Scheduling System based on Set Covering. In Computer-Aided Transit Scheduling (Eds. Daduna J R & Wren A), pp. 160-174, Springer-Verlag, Berlin.

2. Ryan D M (1980) ZIP - A Zero-One Integer Programming Package for Scheduling. Report CSS 85, AERE, Harwell, Oxfordshire, United Kingdom.

3. Goldfarb D & Reid J K (1977) A Practicable Steepest Edge Simplex Algorithm. Mathematical Programming 4, pp. 26-29.

4. Forrest J J & Goldfarb D (1991) Steepest Edge Simplex Algorithms for Linear Programming. IBM Research Report, T J Watson Research Center, Yorktown Heights, New York 10598, USA.

5. Sherali H D (1982) Equivalent Weights for Lexicographic Multi-Objective Programs: Characterizations and Computations. EJOR 18, pp. 57-61.

Extended Abstract

Applied Mathematical Programming and Modelling Symposium
Budapest, Hungary, January 1993

DERIVING THE DUAL OF AN INTEGER PROGRAMME: ITS INTERPRETATIONS AND USES

H.P. Williams

Faculty of Mathematical Studies, University of Southampton, U.K.

This talk will begin by discussing duality in Mathematics in a wider context e.g. in the areas of Set Theory and Logic, Projective Geometry and Convex Polytopes. Some of the **mathematical** properties which are normally expected of a dual will be listed e.g. **Reflexivity** and **Symmetry**. Linear Programming (LP) and Congruence duality will then be examined for both its mathematical properties and **computational** and **economic** uses e.g. **Proving Optimality, Sensitivity Analysis** and **Pricing Imputation**.

A number of possible Integer Programming (IP) duals will be mentioned e.g. the Gomory-Baumol dual, Lagrangean dual and Surrogate dual. They all lack some of the above properties and in particular do not provide a guaranteed proof of optimality.

It will be suggested that the most satisfactory dual arises from examining the **Value Functions** and **Consistency Testers** of IPs. For Pure IPs (PIPs) these take the form of **Gomory Functions**. Gomory functions are built up by the repeated applications of the operations of

(i) Non-negative linear contributions.
(ii) Integer round-up.
(iii) Taking Maxima.

These can be expressed in the form

$$\text{Max}(C_1, C_2, \ldots, C_n) \tag{1}$$

where the C_i are **Chvátal-Functions** which are built up from operations (i) and (ii).

By comparison the Value function of the Consistency Tester of the corresponding LP relaxation will involve functions of the form

$$\text{Max}(\bar{C}_1, \bar{C}_2, \ldots, \bar{C}_r) \tag{2}$$

where $r \leq n$ and $\bar{C}_i$ is obtained from C_i by dropping the operation (ii).

The $\bar{C}_i$ will therefore be non-negative linear conbinations of the right-hand-side coefficients, arising from the dual vertices of the LP.

It will be shown that those C_i which correspond to a $\bar{C}_i$ in (2) can be obtained by finding the Value function of PIPs over cones. This may be done by obtaining the Hermite Normal Form of the corresponding basis matrix for the LP relaxation. The resulting doubly recursive function of the right-hand-side coefficients gives the Value function (and Consistency tester). It is suggested that the depth of this recursion is a measure of complexity. The problem of extending this method to give the Value function and Consistency tester for a general PIP will be considered.

It will be shown that the Value function for a Mixed IP (MIP) is not generally a Gomory function although the Consistency tester is. By incorporating this objective as a constraint and finding the consistency tester of this system it is then possible to characterise the Value function of the MIP.

The Value function for certain MIP applications has considerable economic importance since it shows how indivisible resources should be "priced". This aspect will be considered in relation to the Fixed Charge Problem and the Power Systems Loading Problem.

1 General Problem Description

Analysts frequently face the following problem: given a multivariate (possibly correlated) population, how does one determine a good estimate of the probability function (or some number of its moments) for a complicated function of the population's variables? The primary problem to consider then is what is the most *efficient* way to sample from the input population, especially when sampling is extremely expensive and must therefore be limited to a predetermined (small) sample size. The desire is to generate a sampling plan which will be representative of the population, and produce estimates of moments which have desirable statistical properties. However, since the larger the sample, the larger the cost, there is a trade-off between generating the best estimates and reducing the amount of sampling. In order to obtain better estimates from sampling, analysts may determine them by using data collected from a stratified sampling of the population.

A special form of stratified sampling is latin hypercube sampling. In this stratification, the cumulative distribution function for each of the n population variables is divided into m blocks. The intersection of these blocks makes up a hypercube having m^n cells. If all m^n cells were sampled, the sampling approach would be a "full factorial design". Since sampling is assumed to be expensive, LHS limits the sampling to only m of the m^n possible cells. Thus, a LHS plan is not a hypercube, but is equivalent to a $m \times n$ matrix such that each of the m rows defines one sampling cell of a m^n hypercube.

The i^{th} row of a LHS sampling plan makes up what will be referred to as "run i". Defining this grouping as a *run* is motivated by the fact that typical applications of LHS involve computer-based models where the number of runs, m, is predetermined. To ensure that a plan offers a cross section of the sampling space, an additional feature of LHS is that each block of each variable must be picked once. Thus, each column of a LHS plan is a permutation of the numbers 1 to m.

Obtaining Minimum-Correlation Latin Hypercube Sampling Plans Using Discrete Optimization Techniques

Dr. Leslie-Ann Yarrow
Department of Mathematics and Statistics
Brunel University
Uxbridge, Middlesex UB10 3PH England

Combinatorial optimization, by its broad nature, has been used to model and solve a variety of problems including those arising in decision, engineering, and physical sciences. The focus of this work is to consider the solution of a sampling design problem using combinatorial optimization. The particular design problem of interest here is minimum-correlation latin hypercube sampling (hereafter referred to as MCLHS). The central point of this research is the development of combinatorial optimization procedures which provide MCLHS plans. This is an entirely new approach for finding MCLHS plans.

We introduce integer programming (IP) formulations of this problem and develop a procedure for determining minimum-correlation sampling designs. We provide the obvious IP formulation of the MCLHS problem which results in a problem having an exponential number of variables and a large (polynomial) number of constraints. We then transform the problem into a sequence of assignment problems with side knapsack equations, having a polynomial number of variables. This decomposition was found by exploiting the special structure of the problem and finding tight objective function lower bounds. We note that even after the decomposition, the problem still belongs to the NP-hard class. Although the decomposition and subsequent development of solution procedures for the smaller problems are discussed within the context of the sampling design problem, the approach may be applicable to various permutation-related IP problems such as the general quadratic assignment problem, assignment problems with side constraints, and the asymmetric travelling salesman problem variation where the objective is to find a tour which meets a specific cost value. Thus, while the research presented here focuses on solution approaches for the MCLHS problem, the general theory and findings might well prove useful for the solution of other problems known to be NP-complete.

We begin with a description of the general LHS and MCLHS problems, followed by integer programming formulations and a discussion of optimization procedures developed.

To describe the standard approach to LHS, we begin by writing the vector of variables as $(X_1, X_2, ..., X_n)$ and assume for the time being that the variables are mutually independent. The range of each X_i is divided into m (= number of runs) ascending intervals of equal probability and a random value is drawn on each interval for each variable. Next, we generate the order in which the m values of each variable are to be used in each run by creating a sequence of n random permutations of the integers 1 to m. Finally, we form the required vector for the i^{th} run by taking the i^{th} number from each of the n random permutations.

Latin hypercube sampling plans generated by the standard approach are restricted only in the sense that for each variable, a value must be picked once and only once from each of its m intervals. A point we have not yet considered is the impact that correlations between the columns of a LHS sampling plan may have on the generated estimates. For ease of explanation, we will continue the assumption that the population variables are mutually independent, although similar results are obtained for any given population covariance matrix. For the n variables, although their sampling plan permutations are determined independently, a standard LHS plan will, in general, have some level of correlation between the pairs of permutations. Thus, the sampling plans will not, in general, parallel the correlations of the true joint distributions. If LHS sampling is done without concern for the correlation pattern (or lack thereof), the estimators cannot be guaranteed to be unbiased or even consistent.

The desire then is to design LHS plans which incorporate the variables' true pairwise correlations. For two variables, X_i and X_j, with distribution functions having strictly positive standard deviations, σ_i and σ_j, the correlation coefficient between the variables is defined as

$$\rho_{ij} = \frac{\text{cov}(X_i, X_j)}{\sigma_i \sigma_j}$$

where $\text{cov}(X_i, X_j)$ denotes the covariance between variables X_i and X_j.

To *approximate* the pairwise correlation coefficients ρ_{ij}, we will consider the correlation coefficients between the pairs of LHS plan permutations associated with variables X_i and X_j. (The two forms of correlations are equal when X_i and X_j are both uniformly distributed.) For permutations of the integers from 1 through m, it can be shown that the correlation coefficient of the indices of any pair of permutations is

$$\hat{r} = 1 - \frac{6 \sum_{v=1}^{m} D_v^2}{m(m^2 - 1)},$$

where D_v is the difference between the v^{th} integer elements in the vectors. This is known as the Spearman rank correlation coefficient and can take on values in the interval $[-1, 1]$. The expected value of the rank correlation coefficient is 0, and its variance is $1/(m-1)$. Throughout the remainder of this paper, we denote the rank correlation estimate between the column permutations of variables X_i and X_j by $\hat{r}_{ij}$.

For illustration, suppose we want to run a model with three mutually independent uniformly distributed variables (for simplicity, x, y, and z), each to be represented by values chosen from their respective sample spaces. Assuming further that we are allowed only six runs, consider the LHS plan given below:

Table 1: Latin Hypercube Sampling Example

Model Run	Variable Values		
1	x_1	y_1	z_5
2	x_2	y_6	z_3
3	x_3	y_5	z_4
4	x_4	y_3	z_1
5	x_5	y_2	z_2
6	x_6	y_4	z_6

The rank correlation coefficients for this example are

$$\hat{r}_{12} = 0.00, \quad \hat{r}_{23} = 0.00, \quad \hat{r}_{13} = 0.00,$$

and hence, it appropriately models the mutual independence of the three variables. If, for example, the variables were dependent with true joint distributions having pairwise rank correlations of say, $r_{12} = -.6$, $r_{23} = -.42$, and $r_{13} = .14$, then this particular sampling plan would not suitably parallel these true rank correlations.

The objective of the restricted LHS problem we consider is the selection of column permutations which attempt to meet exactly the true rank correlations associated with the variables. In this way, sampling is intended to match more closely the true marginal distributions of the input variables. Specifically, the minimization problem, called minimum-correlation LHS (MCLHS), provides a sampling specification minimizing the sum of the absolute values of the pairwise differences $(\hat{r}_{ij} - r_{ij})$. In much of the discussion however we will *minimize* the sum of the absolute values of the pairwise rank correlations $\hat{r}_{ij}$. This models the situation when independence of the variables is likely (i.e., $r_{ij} = 0$).

2 Integer Programming Models for MCLHS

The minimum-correlation latin hypercube sampling problem described can be formulated as a n-index assignment problem with side knapsack equation constraints (APSEC). To begin, define:

$$x_{v_1 \ldots v_n} = \begin{cases} 1 & \text{if } v_1 v_2 \ldots v_n \text{ is a sampled cell} \\ & \text{where the } n\text{-indices on the } x\text{-variable,} \\ & v_1, v_2, \ldots, v_n, \text{ can each take a value from 1 to } m \\ 0 & \text{otherwise} \end{cases}$$

and also $d_{ij}^+,\ d_{ij}^- \in \mathcal{R}_1^+$ such that:

$$d_{ij}^+ - d_{ij}^- = (\hat{r}_{ij} - r_{ij})\, m(m^2 - 1)/6 \qquad i = 1 \ldots n, \quad j > i.$$

Thus, d_{ij}^+ and d_{ij}^- are the positive and negative magnitudes of the deviations from the true rank correlation of the rank correlation between column permuations i and j.

Equivalent to minimizing the sum $\sum_{j=2}^{n} \sum_{i=1}^{j-1} \frac{m(m^2-1)}{6} \mid \hat{r}_{ij} - r_{ij} \mid$, is minimizing the objective function

$$\min \left\{ \sum_{i=1}^{n-1} \sum_{j>i} (d_{ij}^+ + d_{ij}^-) \right\}.$$

Although the formulations described below are applicable to cases with nonzero r_{ij}, for ease of presentation, we will assume $r_{ij} = 0$. It can be shown that

$$m(m^2-1)\hat{r}/6 \;=\; m(m^2-1)/6 - \sum D_v^2$$

will be integer-valued for all pairs of permutations. Thus, we can now define $d_{ij}^+,\ d_{ij}^- \in \mathcal{Z}_1^+$ such that:

$$d_{ij}^+ - d_{ij}^- = m(m^2-1)\hat{r}_{ij}/6 \qquad i = 1 \ldots n,\ \ j > i.$$

In order that the IP formulation fully encompasses the MCLHS, it must include assignment constraints that draw a one-to-one correspondence between the positive-valued $x_{v_1 \ldots v_n}$ of a feasible solution and n-tuples of column permutations. Thus, the j^{th} column permutation requires the m assignment constraints

$$\underbrace{\sum_{v_1=1}^{m}\sum_{v_2=1}^{m}\cdots\sum_{v_n=1}^{m}}_{\text{excluding } \sum_{v_j=1}^{m}} x_{v_1 \ldots v_n} \;=\; 1 \qquad v_j \;=\; 1 \ldots m$$

Additional constraints are needed to enforce that
$m(m^2-1)\hat{r}_{ij}/6 = d_{ij}^+ - d_{ij}^-$ holds for all i and j, $i < j \leq n$. These constraints are

$$\sum_{v_1=1}^{m}\sum_{v_2=1}^{m}\cdots\sum_{v_n=1}^{m}(v_i - v_j)^2\, x_{v_1 \ldots v_n} \;=\; m(m^2-1)/6 \quad \forall\, i < j \leq n$$

In addition to belonging to the class of NP-complete problems, we see that this formulation requires m^n x-variables as well as a total of $n(n-1)$ deviational (d_{ij}^+, d_{ij}^-) variables. There are nm assignment constraints and $\binom{n}{2}$ constraints to ensure that $m(m^2-1)\hat{r}_{ij}/6 = d_{ij}^+ - d_{ij}^-$. Hence, although this formulation is the most straightforward, we will present other APSEC formulations which have more reasonable problem size growth.

To develop alternative formulations, we use the objective function lower bound of $\binom{n}{2}6/(m(m^2-1))$ when $m = 2+4l$ for some nonnegative integer l, and zero otherwise, and make the assumption that given $k-1$ column permutations with minimum $\sum_{j=2}^{k-1}\sum_{i=1}^{j-1} \mid \hat{r}_{ij} \mid$, it is possible to fix these columns and find an optimal k^{th} column permutation. Our research and empirical results have shown that these are valid assumptions.

Suppose we have a solution to the $(k-1)$-dimensional problem, and wish to use this solution to obtain a solution to the k-dimensional problem. Let $(p^1, p^2, \ldots, p^k)$ denote the corresponding column permutation vectors, and define

$$x_{ij} = \begin{cases} 1 & \text{if the } i^{th} \text{ element of column } k \\ & \text{is assigned value } j \\ 0 & \text{otherwise} \end{cases}$$

To ensure that column k is a permutation of numbers $1 \ldots m$, we add the assignment constraints :

$$\sum_i x_{ij} = 1 \qquad j = 1, \ldots, m$$

$$\sum_j x_{ij} = 1 \qquad i = 1, \ldots, m.$$

We see that the positive elements of an x-solution define a k^{th} column. We will henceforth interchangably use the terms "an x-solution" and "the k^{th} column defined by the positive elements of x".

There are $(k-1)$ additional constraints of the following form:

$$(1) \quad d_{tk}^{+} - d_{tk}^{-} = m(m^2-1)/6 - \sum_{i=1}^{m}\sum_{j=1}^{m}(p_i^t - j)^2 x_{ij} \qquad t = 1, \ldots, k-1$$

where p_i^t is the i^{th} entry of the column permutation vector p^t. With these constraints, we implicitly fix the $(k-1)$ previously found column permutations.

The formulation defined thus far with objective function

$$\min \sum_{i=1}^{k-1} (d_{ik}^{+} + d_{ik}^{-}),$$

is a general formulation for finding a k^{th} column permutation, having fixed the $(k-1)$ column permutations that minimize $\sum_{i=1}^{k-2} \sum_{j>i} \mid \hat{r}_{ij} \mid$. Empirical evidence strongly supports that there exists a k^{th} column that meets the lower bound for $\mid r_{ik} \mid$, $i = 1, \ldots, k-1$. Hence, there will exist a solution to the assignment constraints that generates a k^{th} column satisfying

$\mid m(m^2-1)\hat{r}_{ik}/6 \mid =$

$$\mid m(m^2-1)/6 - \sum D_v^2 \mid \; = \begin{cases} 1 & \text{if} \quad m = 2+4l, \;\; l \in Z_1^+ \\ 0 & \text{otherwise} \end{cases}$$

for all $i = 1, \ldots k-1$. To incorporate this into the formulation, we require that d_{ik}^{+} and d_{ik}^{-}, $i = 1, \ldots k-1$ be binary variables. For $m \neq 6+4l$, $l \in Z_1^+$, any solution that obtains the lower bound must have $d_{ik}^{+} + d_{ik}^{-} = 0$. If however, $m = 2+4l$, $l \in Z_1^+$, we can conclude that $d_{ik}^{+} + d_{ik}^{-} = 1, \quad t = 1, \ldots k-1$. In either case, the problem can be restated as a *feasibility* problem with no objective function. We shall refer to this feasibility assignment problem with side equations as **FASE**.

The **FASE** formulation follows the conjecture that one can iteratively solve k-dimensional problems using the previously determined $(k-1)$-dimensional solutions. Thus, rather than solving one large APSEC program with $m^n + n(n-1)$ variables and $nm + \binom{n}{2}$ constraints, one could solve a sequence of smaller two-dimensional **FASE** problems with at most $m^2 + 2k$ variables and $2m + k$ constraints ($2 \leq k < n$).

In the presentation, we shall discuss heuristic and Lagrangean-based solution procedures developed to solve the MCLHS problem and its equivalent formulation FASE.

Late Papers

Introduction to Object-Oriented Modeling with GAMS 2.25

Robert Entriken

December 7, 1992

Abstract

This paper describes a standard for the use of GAMS 2.25 as an object-oriented modeling language. The over-riding benefit of using this technique is the ease with which many individuals can simultaneously develop extraordinarily complex modeling systems. Lesser, but still important benefits include: structured user-interface design, plug-in/plug-out models, isolating portions of the problem, easy maintenance and updates, and model re-use. Simultaneous model development stems from the latter benefits, while all of these advantages derive from the clear, rigorous organization of your model as specified in the following standard.

We present the concepts of encapsulation (forming objects) and hierarchical modeling in the context of mathematical modeling. Encapsulation is a well-known programming technique that is newly applied to modeling, and our version of hierarchical modeling differs slightly from past notions. Traditionally, a hierarchical model embodies the concept of forming larger models from a collection of sub-models. The following method is based on a partition of the relations (equations) of the model, where the elements of the partition are partially ordered.

1 Overview

Object-oriented modeling (OOM) is a method of modeling that closely imitates object-oriented programming (OOP) [?,?,?]. We have developed a

standard for using GAMS 2.25 [?] as an OOM language. The difference being that the OO models are much more structured and abstract. This makes them more user friendly because their use is well defined by the structure and their details are hidden within. OO Models thus appear simpler and more uniform to the user.

Four essential properties set OOM apart from standard GAMS 2.25:

Routines: Structuring the assignment statements into procedures as in Pascal.

Encapsulation: Combining data and variables with the equations and assignment statements that manipulate them to form a new data type—a model.

Information Inheritance: Defining a model that uses other models in its formulation, with each sub-model inheriting the information from its ancestors. The use of models within models defines the *use hierarchy* which forms a partial ordering of all used models.

Polymorphism: Giving a model's routine one name that is shared by all descendants in the use hierarchy, with each descendant implementing the routine in a way appropriate to itself.

Routines are implemented using the `$INCLUDE` statment. Encapsulation, inheritance, and polymorphism are implemented in GAMS 2.25 through self-discipline. The following is a detailed discussion of the principles and implementation of OOM in GAMS 2.25 through self-discipline. We hope that the future will bring the language extensions need for a proper implementation. In which case, the standard described below would be enforced by the compiler.

There are now a variety of experimental modeling languages offering object-oriented features, notably ASCEND [?] and MODEL.LA [?]. We offer a form of inheritance that differs from the class inheritance of standard OOP and OOM languages. This is an extra restriction placed on our models based on deferred requirements, and the use of models within other models. Data and variables are legated (passed down) to the descendants, while methods are used by ancestors to ensure that deferred information[1] is properly defined.

[1] Data and variables that have been declared but are yet undefined.

There is a restricted form of communication control between the models of the use hierarchy. Essentially, descendants can inspect ancestor information, but ancestors can only ask that certain information be provided. In this way, siblings communicate through the parent, and its deferred information.

We further expound on these concepts and offer a full accounting of the presentation. First we introduce a model and how it is encapsulated. This leads to an overview of traditional hierarchical modeling. Then we explain how OOM fits into this background. The final section gives the standard itself—how to implement OOM in GAMS 2.25.

References

[1] Bjarne Stroustrup (1986). *The C++ Programming Language*, Addison-Wesley, Reading, Massechusetts, USA.

[2] Borland International (1990). Turbo Pascal 6.0 - User's Guide, Scotts Green, California, USA.

[3] Bertrand Meyer (1988). *Object-Oriented Software Construction*, Prentice Hall International, Hertfordshire, UK.

[4] Anthony Brooke, David Kendrick and Alexander Meeraus (1992). *GAMS - A user's Guide, Release 2.25*, The Scientific Press, South San Francisco, California.

[5] P. Piela, T.G. Epperly, K.M. Westerberg (1991). ASCEND: An object-oriented computer envieronment for modeling and analysis: The modeling language. *Computers and Chemical Engineering* **15**, 53–72.

[6] G. Stephanopoulos, G. Henning, and H. Leone (1990). MODEL.LA: A modeling language for process engineering. Part I. The formal framework. Part II. Multi-facetted modeling of process systems. *Computers and Chemical Engineering* **14**, 813–869.

TOWARD A FORMAL THEORY TO A MODELS' INTEGRATION

M. GAGLIARDI, C. SPERA
Department of Quantitative Methods
University of Siena - Italy

e-mail: Spera@Sivax.cineca.it

EXTENDED ABSTRACT

1. Motivations for a formal theory.

The definition of a specific model is often conceived as a work which has to be done from scratch. In fact, the variety of the variables describing the modelled reality seems to exclude the possibility that a model can be defined assembling pieces of correlated sub-models. To define models from scratch greatly decreases the productivity of the work.

It seems that the keyword in increasing modelling productivity is "reusability". Models can be reused and integrated so to produce new models. Naturally models to be integrated have to be expressed using a common base and the result has to lie on the same framework. In this paper the chosen framework is the Structured Modeling, as formally defined by Geoffrion, [3].

Here we define three integration levels, according to the degree of influence of the operator in the procedures used to merge the models:

- **Level 1** - All the procedures are automated. This means that the user selects the input models and the genera to be integrated, and the output integrated model is automatically produced.
- **Level 2** - The user selects the input models and the order of integration among the genera, and the output integrated model is automatically produced.
- **Level 3** - The user select the input models, the genera to be integrated and formulate the steps necessary to integrate. The output integrated model is not automatically produced, since the steps can vary according to the situation.

2. Preliminary results.

In the rest of this paper we assume that the reader is familiar with the formal theory of the Structured Modeling.

Given a Structured Model M_i, let $G_i = \{g_j, j = 1, \dots, k\}$ be the set of all the genera; this can be partitioned into three disjoined sets: PC, A and FT such that:

PC $= \{g_j \in G_i$: g_j is a primitive or a compound entity genus$\}$
A $= \{g_j \in G_i$: g_j is an attribute genus$\}$
FT $= \{g_j \in G_i$: g_j is a function or a test genus$\}$.

Lemma 1: *Any genus $g_j \in PC_i$ does not have references to any other genera $g_k \in (A \vee FT)$*

Proof: Primitive entity elements, by definition, have no calling sequence, therefore they do not have references to any other element; compound entity elements, by definition, are construct only on primitive entity elements. ■

Lemma 2: *Any genus $g_j \in A_i$ has only references to another genera $g_k \in PC_i$.*

Proof: Attribute elements, by definition, characterize only primitive and compound elements. ■

Lemma 3: *Any genus $g_i \in FT_i$ does not have references to any other genera $g_k \in PC_i$.*

Proof: Function and test elements call, by definition, attribute, function and test elements; therefore, they cannot call primitive and compound entity elements.■

Definition 1: Connected Module, Sub-Model.
*A module is a **Connected Module** if its genera and their calling sequences define a connected graph. A **Sub-model** is a connected module with at least one primitive entity genus.*

Definition 2: Behaviour Equivalence on $\overline{FT}_i \subseteq FT_i$.
*Two structured models M_1 and M_2 are **Behaviour Equivalent** on $\overline{FT}_1 \subseteq FT_1$ and $\overline{FT}_2 \subseteq FT_2$ if the following two conditions hold:*
a) The set $\overline{A}_1$ of the attribute genera directly or indirectly called by the $g_j \in \overline{FT}_1$, and the set $\overline{A}_2$ of the attribute genera directly or indirectly called by the $g_j \in \overline{FT}_2$ have the same structure;
b) $\overline{FT}_1$ and $\overline{FT}_2$ give as output the same values.

We shortly write "behaviour equivalent" when the sub-set $\overline{FT}_i$ coincides with FT_i.

Definition 3: Normal Model.
*A model is called **normal** if an isomorphic relation exists between attribute and compound genera, and their elements.*

The graph of the elements of a normal model is shown in figure 1; dotted rectangles identify genera.

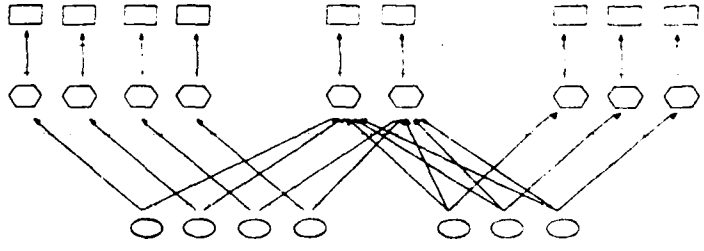


figure 1

Proposition 1. *Given a Structured Model M_i, it is always possible to construct a normal model $N(M_i)$ which is behaviour equivalent to M_i.*

Proof: Let us consider a generic attribute genus $g_j \in A_i \subset M_i$. It is always possible to define a new compound entity genus, $c_k \in PC_i$, with the same calling sequence of g_j. Lemmas 1 and 2 ensure that genera which are called by an attribute genus can be called by a compound entity genus too. An

isomorphic relation can be set among the elements of g_j and c_k: the first element of g_j calls the first element of c_k, etc. This process is repeated for every attribute genus of M_i.
If we indicate with $N(M_i)$ the modified model, the set $B = \{c_k, g_j\} \subset N(M_i) \Leftrightarrow g_j \in M_i$ for every genus $g_k \in FT_i \subset N(M_i)$. ■

Definition 4: Index Basis, Index Basis Set.
*An **Index Basis** of a normal model $N(M_i)$ is a couple of genera $B_j = \{a_j, c_j\}$, where $a_j \in A_i \subset M_i$ is an attribute genus, and c_j is the compound entity genus called by a_j. The genus a_j is called **value component** of B_j, while the genus c_j is called **index component**. The set $BS_i = \{B_j, j:1, \ldots, k\}$ containing all the index basis of $N(M_i)$ is called **Index Basis Set**.*

Definition 5: Index Function.
*An **Index Function** $i(g_j)$ is a rule which associate to every genus $g_j \in N(M_i)$ the cardinality of its index set.*

As example, given a genus g_i indexed by $j \times k \times l$, its index function $i(g_i)$ returns as value 3.

3. Main results.

In this paragraph we try to give an example for each level of integration previously defined.

3.1. Level I integration example.

To show the first level of integration we need to introduce the definition of a function sub-model.

Definition 6: Function Sub-Model.
*SubM(f) is called **Function Sub-model** if the following properties hold:*
a) SubM(f) is a normal model.
b) SubM(f) has at least a function genus $f \in FT_i \subset M_i$ indexed as singleton.

In the following we give a procedure, which transforms a Structured Model M_i with at least one function genus indexed as singleton into a function sub-model.

The following procedure, CREATE_FUNCTION_SUBMODEL, needs as input a model M_i and a singleton genus $f \in FT_i \subset M_i$, and produce as output a function sub-model. The proof of this is given in Proposition 2.

```
procedure CREATE_FUNCTION_SUBMODEL (input: Mi, f; output: SubM(f));
/* Modify Mi into a function sub-model SubM(f) */
begin
     /* step I. "Normalize the model" */
     NORMAL (Mi);
     /* step II. "Merge functions" */
     Create a LIST of calling sequence segments si of f;
     repeat
          Examine the segment si ∈ LIST;
          if the referred genus gk ∈ FTi
          then
               /* a */    Substitute si with the calling sequence of gk ;
               /* b */    Substitute the value field of gk with its rule;
               /* c */    Delete gk;
               Delete the segment si;
     until (end of LIST);
```

```
     /* step III *Delete genera having no intluence on f* */
     Create a LIST of g_j ∈ M_i;
     repeat
          Examine g_j ∈ LIST;
          if (g_j ∈ FT_i and g_j ≠ f)
          then delete g_j;
          if (g_j ∈ A_i ∪ PC_i and g_j is not called directly or indirectly by f)
          then delete g_j;
     until (there are no more g_j ∈ FT_i ⊂ M_i, g_j ≠ f) and (there are no more g_j ∈
     A_i ∪ PC_i ⊂ M_i not called directly or indirectly by f)
end
```

Proposition 2: *Given a Structured Model M_i and an arbitrary singleton function genus $f \in FT_i \subset M_i$, it exists a transformation T such that:*

$T(M) = SubM(f)$

and SubM(f) and M_i are behaviour equivalent on f.

Proof: By applying procedure CREATE_FUNCTION_SUBMODEL which defines the procedure T.■

Let us show how a function genus f can be reused as an input parameter for other models. This action is totally automated, here is an example.

Suppose we have two models M_1 and M_2, we want to substitute the genus $g_j \in A_1 \subset M_1$ with the computed value given by the genus $f \in FT_2 \subset M_2$. This goal is achieved applying the following procedure (the symbol $[M_1, SubM_2]$ means the integrated output model):

```
procedure REUSE (input: M_1, M_2, g_j, f; output: [M_1, SubM_2]);
/* Integrate M_1 and M_2. f is substituted to g_j */
begin
     /* Step I  *Changes in M_2* */
     CREATE_FUNCTION_SUBMODEL (M_2, f; SubM_2(f));
     Create a LIST of genera g_i ∈ A_2 ⊂ SubM_2(f);
     repeat
          Add the calling sequence's segments of g_j ∈ A_1 to the calling sequence of
          g_i ∈ A_2 ⊂ SubM_2(f);
     until end of LIST;
     /* Step II *Changes in M_1* */
     Create a LIST of genera g_i ∈ FT ⊂ N(M);
     repeat
          Select g_i from LIST;
          if g_i calls g_j ∈ A_1
          then
               Substitute g_j with f in the calling sequence of g_i;
          LIST := LIST - g_i;
     until end of LIST
     /* Step III  *Delete attribute genus* */
     Delete g_j ∈ A_1;
end.
```

Proposition 3: *Given two Structured Models M_1 and M_2, it is always possible to substitute an attribute genus $g_j \in A_1 \subset M_1$ with a singleton function genus $f \in FT_2 \subset M_2$. The result is a Structured Model.*

Proof: By applying the procedure REUSE we obtain as result the model $[M_1, SubM_2]$. Its graph of genera must be finite, closed and acyclic.

a) **Finiteness.** Step III guaranties that the number of genera of $[M_1, SubM_2]$ is equal to the number of genera of $(M_1 \cup SubM_2(f)) - g_j$.

b) **Closure.** By steps I and II, there is at least one genus of M_1 calling a genus of $SubM_2(f)$ and at least one genus of $SubM_2(f)$ calling a genus of M_1. From closure of M_1 e $SubM_2(f)$ it follows the closure of $[M_1, SubM_2]$.

c) **Acyclicity.** Let us consider an arbitrary sub-set of genera $G_i \subseteq [M_1, SubM_2]$, and let us assume that it is cyclic. Therefore, G_i contains genera belonging to both models, because no new references are set by the procedure among genera belonging to only one model. By construction, the sequence must be of the type:

$\{ \ldots, a_i \in A_2 \subset SubM_2(f), f, \ldots \}$.

The genus following f in the sequence has to be a function genus, while the genus preceding a_i has to be a compound entity genus. By Lemma 3 there are no references among function genera, and compound entity genera. Therefore, G_i cannot by cyclic. ■

Figure 2 shows how two models are integrated.

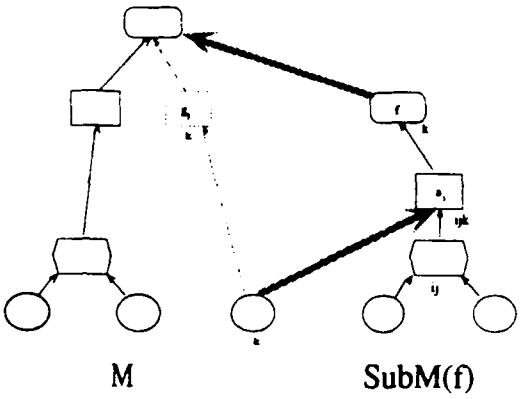


Figure 2

Proposition 4. *Given two normal models M_1 and M_2, the integrated model obtained substituting an input parameter $g_1 \in A_1 \subset M_1$, with an output parameter $g_2 \in FT_2 \subset M_2$ is a Structured Model if $i(g_1) = i(g_2)$.*

Proof: It follows the same line of proposition 3. The necessary condition given by the equality of the index functions ensures the closure and acyclicity of the graph of the elements.■

Given the result of proposition 4 the following procedure can be constructed. The input parameters are the two normal models, an index basis of the model M_1 and a function genus of the model M_2.

```
procedure USE (Input: N(M1), N(M2), B1, f; Output: [N(M1), N(M2)]);
begin
     Select a1 ∈ B1;
     Compute i1(a1);
     Compute i2(f);
     if i1(a1) ≠ i2(f) then exit;
     Create a LIST of genera gi ∈ FT1 ⊂ N(M1);
     repeat
          Select gi from LIST;
          if gi has a reference to a1 then
               Substitute the reference to ai with a reference to f;
          LIST := LIST - gi;
     until end of LIST;
     Delete B1;
end.
```

The following steps create an integrated model, which is the same result as in Geoffrion. The final graph of genera obtained applying sequentially step 0 - step IV is shown in figure 4.

```
Step 0.
NORMAL (fin);
NORMAL (mkt);
NORMAL (mar);
NORMAL (mfg);

Step I.
SUBSTITUTE (mkt, mar, [P,D1], [P,D2]);
USE (mar, mkt, [V,D3], V);

Step II.
USE (mfg, mar, [V,D5], V);
USE (mar, mfg, [E,D4], E);

Step III.
SUBSTITUTE (fin, mar, [P,D6], [P,D2]);
USE (fin, mar, [E,D8], E);
USE (fin, mar, [V,D9], V);

Step IV.
MERGE (mfg, mar, P, U);
```

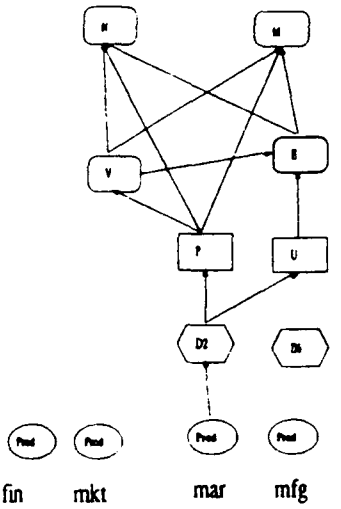


Figure 4

3.3 Level 3 integration example.

At this level of integration the user needs to define the steps to integrate the models, and there are no automated procedure. Let us present another example extracted from Geoffrion [4]. The steps are informally defined, since the user will formalize them.

```
Step 1
Delete DEM and T:DEM genera from TRANS1
Delete SUP and T:SUP genera from TRANS2

Step II
Merge genus CUST from TRANS1 with genus
PLANT from TRANS2;

Step III
Create new genera T:DC and define its
reference;

Step IV (Optional)
Create a new genus TOT$ being the sum of
the TOT$ function genera of the two
models;

Step V (Optional)
Rename genera;
```

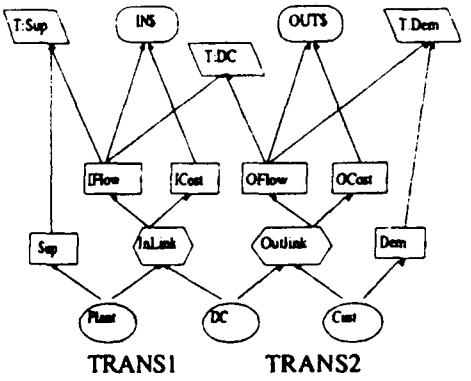


Figure 5

4. Conclusions.

The first remark about the definition of a formal theory to models' integration is *modularity*. This can be easily achieved projecting the theory of the Structured Modeling into the same space of the Object Orientation Principles.

The second remark regards the construction of three sub-sets which contain the procedures characterizing the formal rules of the three integration levels.

Both aspects will be deeper developed in the future.

3.2. Level 2 integration example.
In this case the role of the user is relevant, since the input parameters to be merged are only identified by him.

Proposition 5. *Given two normal models $N(M_1)$ and $N(M_2)$ and the corresponding index basis set BS_1 and BS_2, the integrated model obtained substituting in $N(M_1)$, $B_j \in BS_1$ with $B_k \in BS_2$ is a Structured Model.*

Proof: To substitute B_j with B_k implies that every genus $g_j \in FT_1$ has to replace the reference in its calling sequence to $a_j \in B_j$ whit $a_k \in B_k$. The graph of genera of the integrated model has to be: (a) finite; (b) closed and (c) acyclic. (a), (b) hold by construction; (c) hold by lemma 2.■

Proposition 6. *Given two normal models $N(M_1)$ and $N(M_2)$, and the corresponding index basis set BS_1 and BS_2, the integrated model obtained substituting, in $N(M_1)$, an index component $c_j \in B_j \in BS_1$ with an index component $c_k \in B_k \in BS_2$ is a Structured Model*

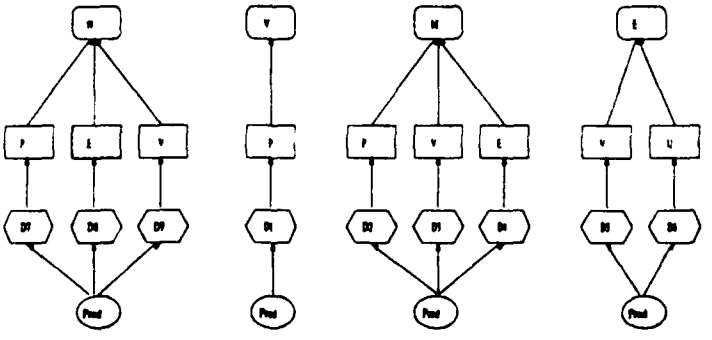
Proof: It follows the lines of proposition 5.■

Given the results of the proposition 5 and 6 the following procedures can be constructed.

```
procedure SUBSTITUTE (Input: N(M1), N(M2), B1, B2;
                            Output: [N(M1), N(M2)]);
begin
     Create a LIST of genera gi ∈ FT1 ⊂ M1;
     repeat
          Select gi from LIST
          Substitute A1 ∈ B1 with A2 ∈ B2 in the calling sequence of gi;
          LIST := LIST - gi;
     until end of LIST;
     Delete B1;
end.

procedure MERGE   (Input: N(M1), N(M2), B1, B2;
                         Output: [N(M1), N(M2)]);
begin
     Select c1, a1 ∈ B1, c2 ∈ B2;
     Substitute c1 with c2 in the calling sequence of a1;
end.
```

In the next we treat the core example extracted from Geoffrion [4]. The sub-models to be integrated are shown in figure 3 (the details are omitted):



fin mkt mar mfg

Figure 3

References

[1] ANDRONICO, A., COSSA, L. , GAGLIARDI, M. e SPERA, C. (1992b)."An Object Oriented Approach to a Model Management System: Characteristics and Examples", in *Atti 36° Convegno Nazionale ANIPLA*, Bottaro, Zoppoli Editors, Pirella Genoa.

[2] GEOFFRION, A.M. (1987). "An introduction to Structured Modeling", *Management Science*, vol. 33, pp. 547-589.

[3] GEOFFRION, A.M. (1989). "The formal aspect of Structured Modeling", *Operations Research*, vol. 37, pp. 30-51.

[4] GEOFFRION, A.M. (1990c). "Reusing Structured Models via Model Integration", Working paper #362, Western Management Science Institute, UCLA.

[5] KOTTEMAN, J.E. e DOLK, D.R. (1990). "Model Integration and Modeling Languages", Working paper, Naval Postgraduate school.

[6] TSAI, Y. (1987). "An Operational Approach to Model Integration Using a Structured Modeling Framework", Research Paper, Anderson Graduate School of Management, UCLA.

The Model Recognition Phase in a Model Management System

P. LOMAGISTRO, C. SPERA
Department of Quantitative Methods
University of Siena, Italy

EXTENDED ABSTRACT

1. Introduction

As pointed out by many authors, a Model Management System (MMS) provides for creation, storage, manipulation, and access to models. MMS functions can be divided in two main groups: Model storage functions and Model manipulation functions. The former includes Model Building, Model Representation, physical and logical Model Storage and Model Retrieval; the latter includes Model Instantiation, Interface with Databases, Model Maintenance, Links between model and Algorithms, and Model Solving.

Model representation schemes plays a key role in the implementation of effective MMSs. To fully implement the functions of MMSs, we need to state a rigorous conceptual framework with a single model representation leading to:

1) independence of model representation and model solution,
2) representational independence of general model structure and detailed data needed to describe specific model instances.

A system based on these ideas would show its usefulness for most phases of the life-cycle associated with model-based work (Geoffrion 1987). For example, consider a mathematical programming problem. Once a model of this problem has been constructed, a MMS should allow the user to perform the following steps:

1) select the solution technique (if any),
2) solve the model,
3) conduce sensitivity analysis.

To automate steps 1 and 2, the system has to be able:

a) to *recognize* what kind of model arises (so that it could automatically select the appropriate solver);
b) to *translate* data instantiating the model (querying the Database where they are stored) into the format required by the selected solver.

This paper will focus on the model recognition phase. We will try to give its theoretical foundations and to define which conditions a model definition language has to satisfy so that the resulting representation is "recognizable".

Our formalization of the recognition process is based on the concept of "minimal representation". A representation of a model is minimal if any other equivalent representation of the same model can be "reduced" to it.

2. Model Recognition Problem: Preliminary Results.

The aim of this section is to provide for some formal definitions. In the next, we will use them to illustrate how recognition process can be carried out.

The recognition process we are trying to formalize is based on the concept of minimal representation. A representation of a model is minimal if any other equivalent representation of the same model can be reduced to it.

In the rest of this paper we will define and explain minimality, equivalence and reduction of model representations; first we need to define what we intend for "model" and "model representation".

Definition 1

We define the system M to be a model of the system P if:
— M does not interact neither directly nor indirectly with P
— M is used to obtain information about P
— M comprises all the elements of P relevant for the intended purpose of the model.

Definition 2

Given a formal language L and a model M_i we define $L(M_i)$ to be its formal representation under L, if it comprises the expression in language L of all the elements of M_i and of the interactions existing among them.

In the following we will use the terms "*model representation*" or simply "representation" to indicate the "formal" representation of a given model under some formal language.

Let us consider, as an example, model M_i as the model of the system P that computes the mean of a given series of values belonging to P; if L is the standard algebraic notation, then $L(M_i)$ will be:

$$\text{mean} = \frac{\sum_{i=1}^{n} x_i}{n} \qquad [1]$$

If $L(M_i)$ exists and is unique, then the recognition problem has a trivial solution, because there is a 1:1 correspondence between model and its representation. Unfortunately, except for very few cases, the model M_i has many representations $L_j(M_i)$, j=1, ..., n, n>1. Referring to the previous example, two other ways to represent the same model are the following ones:

$$\begin{cases} sum = \sum_{i=1}^{n} x_i \\ mean = \dfrac{sum}{n} \end{cases} \quad [2] \qquad result = \frac{\sum_{k=1}^{z} y_k}{z} \quad [3]$$

It is intuitive that all previous representations are equivalent, in so far as they "do the same thing". Nevertheless, for our purposes we need a more rigorous definition of equivalence based on the concept of "transformation rule".

We can think to a transformation rule as to a function or procedure whose input is the whole model representation or a part of it, and whose output is a new model representation or a part of it. Obviously, the output of a transformation rule must be semantically consistent with its input. Let us give its formal definition:

Definition 3

Consider a formal language L and two distinct sets E_1 and E_2 of expressions of L semantically identical. Let R be the set of all transformation rules defined on L; $r \in R$ is defined to be a **transformation rule** on L if applied to E_1 transforms it into E_2.

The existence of transformation rules is very important to state formally the equivalence of model representations. Two equivalent representations must be semantically identical; in other words, there exist two (sets of) transformation rules that transform one into the other, and viceversa. We can formalize the equivalence between model representations as follows:

Definition 4

Let $S_L = \{ L_j(M_i): j=1, ..., n; n>1 \}$ be the set of all possible representation of M_i in the language L. Two representations $L_j(M_i)$, $L_k(M_i) \in S_L$, $j \neq k$, are defined to be **equivalent** if there are two sets of transformation rules, R_1 and R_2, defined on L such that R_1 applied to $L_j(M_i)$ transform it into $L_k(M_i)$, and R_2 applied to $L_k(M_i)$ transform it into $L_j(M_i)$. If $R_1 = R_2$ then the two representations are defined **identical**. Obviously, identical representations are also equivalent.

As an example, let us consider two transformation rules, called ***split*** and ***join*** suitable to be applied to representations [1] and [2]. The terms LHS and RHS stand for respectively "left hand side" and "right hand side".

```
transformation split
input
    in_fraction type fraction
output
    out_assignment type assignment statement
    out_fraction type fraction
begin
set RHS of in_assignment to numerator of in_fraction
set numerator of out_fraction to LHS of out_assignment
set denominator of out_fraction to denominator of in_fraction
end
```

```
transformation join
input
    in_assignment type assignment statement
    in_fraction type fraction
output
    out_fraction type fraction
begin
if LHS of in_assignment = numerator of in_fraction then
    exit join
set numerator of out_fraction to RHS of in_assignment
set denominator of out_fraction to denominator of in_fraction
end
```

The rule ***split*** performs the following operations: given a fraction, it reads its numerator and assigns it to an intermediate variable, and then it builds another fraction whose numerator and denominator are respectively the intermediate variable and the denominator of the given fraction.

The rule ***join*** acts as follows: given an assignment statement and a fraction whose numerator is the variable on the left hand side of the assignment statement, it builds a new fraction whose numerator and denominator are respectively the right hand side of the assignment statement, and the denominator of the of the given fraction.

Since we can transform representation [1] into representation [2] and vice versa by applying respectively transformation rules *split* and *join*, they are equivalent in the sense expressed in Definition 3. They are not identical, since transformation rules we need to apply are different.

Let us now consider a third rule, called ***rename***, which renames all the elements of a model definition, or a part of them, subject to the simple constraint that all elements with identical name in the input model representation must have identical name in the output one. Model representation [3] is one of the possible results of applying rule *rename* to [1]. Since transformation rule we need to apply to transform representation [1] into representation [3] and vice versa is the same, they are identical.

3. Model Recognition Problem: Basic Ideas.

As asserted in first section of this paper, our main task is to determine which conditions have to be satisfied so that the recognition of a model can be performed. For this purpose, we state that the language L must allow that the set of model representations it produces can be ordered by rank. The rank is a measure, defined on some measurable aspect of L, which allows to class and order model representations. We formalize that as follows:

Definition 5

A formal language L satisfies the property of **rankability** if:

- all model representations $L_j(M_i) \in S_L$ are equivalent;
- all model representations $L_j(M_i) \in S_L$ can be ranked
- S_L can be partitioned by rank and all the elements in the same cell of the partition are identical.

In previous examples we might consider the number of equations as rank. If so, then representation [1] and [3] are of rank 1 while representation [2] is of rank 2. Since all representations are equivalent, and representations [1] and [3] are identical, then the property of rankability holds.

Now, let us explain how recognition process can be carried on. To recognize a model

representation means that we have to determine the model it represents. The basic idea of this process is to transform the representation to recognize into another one that we know the kind of model it represents. So doing, we have "recognized" the model.

If the representation we deal with are expressed in a language L satisfying rankability, then all representations of the same model are equivalent. So, if we know all transformation rules that language L allows, then we can recognize any representation simply transforming it into the known one by applying to it the appropriate transformation rules.

The set of all transformation rules may be incredibly large or even not finite. This fact can influence the efficiency of the recognition process. The recognition process can be carried out more efficiently if it is based on the ideas of "minimal representation" and of "reduction rule" defined as follow:

Definition 6

A model representation $L_i(M_j)$ is defined to be **minimal** if:

— L satisfies the property of rankability;

— it has the lowest possible rank.

Definition 7

Given a language L satisfying rankability, **reduction rules** are defined to be transformation rules which when applied to a model representation $L_k(M_j) \in S_L$ of rank k produce a model representation $L_j(M_j) \in S_L$ of rank $j < k$.

Referring to previous example, we can consider representations [1] and [2] as minimal ones.

Property of rankability plays a crucial role for our purposes; in fact, if L satisfies rankability, all reduction rules are known, and they form a finite set then:

— it always admit a minimal representation (i.e. a representation which has the lowest possible rank, and to which any other representation of the same model can be reduced);

— any model representation in language L can be reduced in its minimal form (by applying to it the appropriate reduction rule until no more rule can be applied);

— all minimal representations of the same model are identical.

Under the above mentioned condition, the recognition process of a given model representation can be based on the minimal representation by performing the following basic steps:

1) reduce the model representation to recognize to its minimal form;
2) search among the "known" minimal model representation for a template matching the minimal representation obtained by step 1.

Since for any given language L, the set of the reduction rules must necessarily be a subset of the set of the transformation rules, the recognition process of a given model based on the minimal representation is more efficient than the previous one.

Now, we can define formally the condition under which a given model definition language generates "recognizable" model representations:

Definition 8

A model representation is defined to be **recognizable** if the recognition process:
— can be based on its minimal representation
— can be performed in a finite number of steps.

Claim 1

A formal model definition language L generates «recognizable» model representations if:
— it satisfies the property of rankability,
— the set of all reduction rules it admits is finite.

Proof:

If L satisfies property of rankability then it always admit a minimal representation. If the set of the reduction rules is finite any model representation can be reduced to its minimal form in a finite number of steps. In this way both the conditions which state the recognizability of a model representation are satisfied. ■

3. Conclusions

Here we have sketched the fundamental lines to "recognize" models representation. It seems to us that the idea of minimality looks very promising to be further investigated.

Main References

A. ANDRONICO, C. SPERA, P. LOMAGISTRO, *Toward a Structured Modeling System»*, Quaderno del Dipartimento di Matematica, n. 200, August 1991. Submitted to *Decision Support System*.

A.M. GEOFFRION, "An introduction to Structured Modeling", *Management Science*, vol. 33, pp. 547-589.

A CONSTRAINT SATISFACTION APPROACH TO JOB SCHEDULING WITH TIME INTERVAL AND PRECEDENCE RELATIONSHIPS

Anna Sciomachen

Dipartimento Science dell'Informazione
Universita' degli Studi di Milano
Via Comelico 39/41 - 20135 Milano - ITALY

Extended Abstract

This paper describes a simple methodology for reasoning about temporal and precedence constraint satisfiability problems arising in job scheduling. In particular, a Constraint Satisfaction Problem (CSP) approach is presented.

Several researchers, coming both from Artificial Intelligence (AI) and Operations Research (OR) have investigated methods for dealing efficiently with time (see, e.g., [2, 3, 7, 12]); however, at least to the author's knowledge, only very few real and large scale scheduling applications have been approached using this relatively new technique [4].

In this paper, among all the job scheduling problems, an application in which a set V of n jobs has to be processed on a single machine is considered, such that a release date r_i, a deadline d_i and a process time p_i are associated with each job $i \in V$. The problem is formulated on a constraint network, i.e., a digraph $G = (V,A)$ of n nodes (jobs). An arc $(i,j) \in A$ means that job j can be processed immediately after job i. A weight p_j and the attributes r_j and d_j for each node $j \in V$ are given. Moreover, a digraph $P = (V,E)$, with $E \subseteq A$, is given such that an arc $(i,j) \in E$ represents a precedence constraint between

jobs i and j. The problem consists of determining the starting time for processing the jobs in V such that the time windows (defined by r_j and d_j) for scheduling the execution of each node (job) is satisfied and the precedence constraints between nodes given by the relationships defined in arc set E are satisfied within a time horizon (production plan).

Based on the Allen's model for temporal logic [1], a CSP formulation is first presented. A CSP consists of a set of variables $X = \{x_1, x_2, ..., x_n\}$, their associated domains $D_1, D_2, ..., D_n$ and a set C of constraints on these variables. A solution to a CSP consists of an instantiation of all the variables which does not violate any of the constraints. In the case of the application considered in this paper, let X be the set of variables such that x_i represents the starting time for processing job i, $\forall\, i \in V$. A domain D_i is associated with each variable x_i such that D_i = { set of available Time Machine Units (TMUs) for processing job i (production plan) }. The set C of constraints is defined by two classes of constraints, namely C_1 and C_2, such that $C = C_1 \cup C_2$, C_1 = { unary constraints (time interval) } = $\{ r_i \,\forall\, i \in V \} \cup \{ d_i \,\forall\, i \in V \}$ and C_2 = { binary constraints (precedences) } = $\{ (i,j) \in E \}$. The problem is to verify whether an instantiation of all the variables is possible such that all the jobs are completed within their time interval and no precedence relationship is violated.

Starting from the Allen's interval algebra, the temporal relations are specified by atomic relations. In particular, for each pair i,j of jobs the following atomic relations are defined:

- *After(j,i)*: this specifies the precedence relationship between i and j, i.e., $(i,j) \in E$;
- *Available(i,r_i,D_i):* this specifies the release date of job i within the production plan;

- *Due(i,d_i,D_i):* this specifies the deadline of job i within the production plan.

The constraint network G of this problem is then "preprocessed" such that to compute the tightest possible bound for both unary and binary constraints on the jobs. In particular, given the explicit precedence relationships between jobs the possibility of inferring additional implicit precedence relationships are explored; for instance, the transitivity of the predicate After(j,i) may allow to infer information such that

- $After(j,k) \cap After(k,i) \rightarrow After(j,i)$.

Moreover, the availability interval of each job within the production plan is computed by considering its release date, deadline and precedence relationships. The new domain D_i' for each job i in V is hence computed such that the predicate

- $D_i' = Interval(i,r_i,d_i) = D_i \cap Available(i,r_i,D_i) \cap Due(i,d_i,D_i)$

returns the restricted time interval in which each job has to be processed in order to obtain a feasible scheduling of the jobs. Note that all the possible instantiations of the corresponding variables are thus noticeably reduced after the computation of D_i', $\forall i \in V$.

It is worth mentioning that such a preprocessing approach allows for further generalization of the proposed scheduling problem; for instance, it could be necessary to take into account a possible decomposition of the jobs into different subtasks [13], to analyze periodic scheduling problems [9] or to consider setup times between jobs [5].

A Prolog-like algorithm is then presented for finding a consistent assignment for the variables, i.e., an instantiation of all the variables which does not violate any of the constraints given by both C_1 and C_2. In particular, the procedure

- $x_i = Assign(i,Di)$

associates a value in the new domain D_i' with the corresponding variable x_i, such that a feasible starting time for processing job i is given.

In this phase, following the most-constrained approach suggested in [12], the job having the tightest constraints is selected first. In particular, the procedure

- *Preorder(X)*

performs a sort of the set of variables in such a way that the most critical job, i.e., the most constrained job, is chosen first for its instantiation.

In this particular application the most constrained path is proven to be the most efficient implementative approach, in the sense that the number of backtrackings is minimized (see, e.g., [6, 7] for an overview of the complexity of this kind of temporal CSP problem).

Note that a different way for finding a feasible instantiation of all the variables is to look for an initial solution, possibly inconsistent, and then incrementally repair constraint violations until a consistent assignment is achieved. Such an approach is proposed in [10] in the case of scheduling problems without precedence and time window constraints.

The application field and computational experiences related to real-life cases are also given in the full paper. Some conclusions along with a comparison with a more traditional mathematical programming approach (see, e.g. [5, 8]) for solving the scheduling problem under consideration are finally derived.

References

[1] J.F. Allen, "Maintaining knowledge about temporal intervals", *Communications of ACM* 26 (1983), pp.832-843.

[2] J. Cohen: Constraint Logic Programming Languages - "*Communications of the ACM* 33-7 (1990), pp.52-67.

[3] M. Dincbas, P. Van Hentenryck, H. Simonis, A. Aggoun, T. Graf, F. Berthier: The Constraint Logic Programming Language CHIP - Proc. Int. Conf. on "Fifth Generation Computer Systems", ICOT, pp.693-702, 1988

[4] M. Dincbas, P. Van Hentenryck, H. Simonis - "Solving the car-sequencing problem in constraint logic programming" - European Conference on Artificial Intelligence, Munich, Germany, August 1988

[5] L. F. Escudero, A. Sciomachen: Local search procdures for improving feasible solutions to the sequential ordering problem - Accepted for publication in "Annals of OR" - G. Mitra, I. Maros Eds, 1992.

[6] M.S. Fox, N. Sadeh, "Why is scheduling difficult? A CSP perspective", Proc. of the 9th European Conf. on Artificial Intelligence, August 1990

[7] M.C. Golumbic, R. Shamir: Complexity and Algorithms for reasoning about time: a Graph-theoretic Approach - Rutcor Research Report RRR 22-91, 1991