A TECHNIQUE FOR SPEEDING UP THE SOLUTION OF THE LAGRANGEAN DUAL

Dimitris Bertsimas and James B. Orlin

OR 248-91 April 1991

Report Documentation Page

Form Approved
OMB No. 0704-0188

1. REPORT DATE	2. REPORT TYPE	3. DATES COVERED
APR 1991		**00-04-1991 to 00-04-1991**

4. TITLE AND SUBTITLE	5a. CONTRACT NUMBER
A Technique for Speeding Up the Solution of the Lagrangean Dual	5b. GRANT NUMBER
	5c. PROGRAM ELEMENT NUMBER
6. AUTHOR(S)	5d. PROJECT NUMBER
	5e. TASK NUMBER
	5f. WORK UNIT NUMBER
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) **Massachusetts Institute of Technology,Laboratory for Information and Decision Systems,77 Massachusetts Avenue,Cambridge,MA,02139-4307**	8. PERFORMING ORGANIZATION REPORT NUMBER
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)	10. SPONSOR/MONITOR'S ACRONYM(S)
	11. SPONSOR/MONITOR'S REPORT NUMBER(S)

12. DISTRIBUTION/AVAILABILITY STATEMENT
Approved for public release; distribution unlimited

13. SUPPLEMENTARY NOTES

14. ABSTRACT

15. SUBJECT TERMS

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON
a. REPORT	b. ABSTRACT	c. THIS PAGE			
unclassified	**unclassified**	**unclassified**		**31**	

A technique for speeding up the solution of the Lagrangean dual

Dimitris Bertsimas * James B. Orlin †

April 10, 1991

Abstract

We propose new techniques for the solution of the LP relaxation and the Lagrangean dual in combinatorial optimization problems. Our techniques find the optimal solution value and the optimal dual multipliers of the LP relaxation and the Lagrangean dual in polynomial time using as a subroutine either the Ellipsoid algorithm or the recent algorithm of Vaidya. Moreover, in problems of a certain structure our techniques find not only the optimal solution value, but the solution as well. Our techniques lead to significant improvements in running time compared with previously known methods (interior point methods, Ellipsoid algorithm, Vaidya's algorithm). We apply our method to the solution of the LP relaxation and the Lagrangean dual of several classical combinatorial problems, like the traveling salesman problem, the vehicle routing problem, the Steiner tree problem, the k-connected problem, multicommodity flows, network design problems, network flow problems with side constraints, facility location problems, K-polymatroid intersection, multiple item capacitated lot sizing problem, etc. In all these applications our techniques significantly improve the running time and yield the fastest way to solve them.

Key words. Lagrangean relaxation, linear programming relaxation, polynomial algorithm.

*Dimitris Bertsimas, Sloan School of Management, MIT.

†James B. Orlin, Sloan School of Management, MIT.

1 Introduction

During the last two decades combinatorial optimization has been one of the fastest growing areas in mathematical programming. One major success has been computational: researchers have been able to solve large-scale instances of some NP-Hard combinatorial optimization problems. The successful solution approaches typically rely on solving, either approximately or exactly, the linear programming (LP) relaxation or the Lagrangean dual of an integer programming formulation of the problem.

The successes in using linear programming methodology (LP relaxations, polyhedral combinatorics) have been very impressive. Starting with the seminal computational research conducted by Crowder and Padberg [8] and Padberg and Hong [26], researchers have now been able to solve to optimality a number of applications including traveling salesman problems with up to 2000 nodes (Padberg and Rinaldi [27]), a variety of large scale (up to about 2000 integer variables) real world zero-one business planning problems (Crowder, Johnson and Padberg [7]), input-output matrices that arise in economic planning (Grötschel, Jünger and Reinelt [14]), multiple item capacitated lot size problems that arise in production planning (Barany, Van Roy and Wolsey [5] , Eppen and Martin [10], Leung, Magnanti and Vachani [21]), strategic planning problems (Johnson, Kostreva and Suhl [19]), production planning problems with changeover costs (Magnanti and Vachani [23]), and certain spin glass problems that arise in physics (Barahona et. al. [4]).

These successes show that linear programming and Lagrangean relaxations can play an important role in solving many problems met in practice. The landmark in the development of Lagrangian relaxation (see for example, Geoffrion [12] or Fisher [11]) for combinatorial optimization problems were the two papers for the traveling salesman problem (TSP) by Held and Karp [16], [17]. In the first paper, Held and Karp [16] proposed a Lagrangian relaxation based on the notion of 1-tree for the TSP. Using a complete characterization of the 1-tree polytope, they showed that this Lagrangian relaxation gives the same bound as the LP relaxation of a classical formulation of the TSP. In the second paper, Held and Karp [17] introduced a method, which is now known under the name of subgradient optimization

(see also Held, Wolfe and Crowder [18]), to solve the Lagrangian dual. Despite its success in computational experiments subgradient optimization is not known to be a polynomial method.

In this paper we propose new techniques for the solution of the LP relaxation and the Lagrangean dual in combinatorial optimization problems. Our techniques find the optimal solution value and the optimal dual multipliers of the LP relaxation and the Lagrangean dual in polynomial time using as a subroutine either the Ellipsoid algorithm [20] or the recent algorithm of Vaidya [29]. Moreover, in problems of a certain structure our techniques find not only the optimal solution value, but the solution as well. Suprisingly, our method is significantly faster than interior point methods and Ellipsoid like methods directly applied to the problem. We apply our techniques to the solution of the LP relaxation and the Lagrangean dual of several classical combinatorial problems, like the traveling salesman problem, the vehicle routing problem, the Steiner tree problem, the k-connected problem, network design problems, network flow problems with side constraints, facility location problems, K-polymatroid intersection, multiple item capacitated lot sizing problem, etc. In all these applications our techniques significantly improve the running time and yield the fastest way to solve them. Our technique can also be used to speed up the solution of the dual of multicommodity flow problems in certain cases.

The paper is structured as follows: In section 2 we introduce our method and its variations. In section 3 we apply it to a wide variety of classical combinatorial optimization problems. The final section contains some concluding remarks.

2 The theoretical development of new techniques

In this section we develop the algorithm to solve linear programming problems. The method is particularly useful for problems of special structure as those appearing in the solution of LP relaxations of combinatorial optimization problems.

2.1 An algorithm for linear programs of special structure

Consider the following LP:

$$(P_1)\ z_1 = Min\ cx \tag{1}$$

subject to

$$Ax = b$$

$$x \in S,$$

where S is a polyhedron. Let x^k, $k \in J$ be the extreme points of S, and similarly let w^k, $k \in R$ be the extreme rays of S. Then applying the resolution theorem for the polyhedron S we obtain that any point $x \in S$ can be written as:

$$x = \sum_{k \in J} \lambda_k x^k + \sum_{k \in R} \theta_k w^k$$

$$\lambda_k, \theta_k \geq 0,\ \sum_{k \in J} \lambda_k = 1.$$

Substituting in (1) we obtain that problem (P_1) is equivalent with

$$(P_1')\ Min\ z_1 = \sum_{k \in J} \lambda_k (cx^k) + \sum_{k \in R} \theta_k (cw^k)$$

subject to

$$\sum_{k \in J} \lambda_k (Ax^k) + \sum_{k \in R} \theta_k (Aw^k) = b$$

$$\sum_{k \in J} \lambda_k = 1$$

$$\lambda_k,\ \theta_k \geq 0.$$

The dual of the above problem is

$$(D_1)\ z_1 = Max\ yb + \sigma$$

subject to

$$y(Ax^k) + \sigma \leq cx^k,\quad k \in J,$$

$$y(Aw^k) \leq cw^k,\quad k \in R.$$

Because of strong duality, problem (D_1) has exactly the same value as the original problem (P_1). Problem (D_1), however, has a potentially exponential number of constraints. In order to find z we will solve problem (D_1) using either the ellipsoid method or the recent algorithm of Vaidya [29].

Given a solution y_0, σ_0 for the dual problem, we need to solve the separation problem:

$$(SEP)\ u = Min\ (c - y_0 A)x$$

subject to

$$x \in S.$$

If the solution of this problem is bounded and $u \geq \sigma_0$ then the solution (y_0, σ_0) is feasible in (D_1) and thus we can start Vaidya's sliding objective algorithm (see [29]) or the Sliding objective Ellipsoid algorithm (see for example Nemhauser and Wolsey [25], p.155) with the same separation problem. If $u < \sigma_0$, a new extreme point is generated, which is added to (D_1). If, on the other hand, the subproblem is unbounded, then a new extreme ray of S is generated, which is also added to (D_1). In the last two cases the ellipsoid method or Vaidya's algorithm will continue with one extra constraint. The algorithm will compute the optimal value z_1^* of (D_1) (which is also equal to the optimal value of (P_1)), as well as the optimal dual solution y^* and σ^*.

In general, Vaidya's [29] algorithm for an LP with k variables takes $O(kL)$ iterations with a running time

$$O(TkL + M(k)kL), \tag{2}$$

where T is the number of arithmetic operations to solve the separation problem (SEP), L is the input size of problem (P_1) and $M(k)$ is the number of arithmetic operations for multiplying two $k \times k$ matrices. (It is known that $M(k) = O(k^{2.38})$ [6]). For comparison, the number of iterations of the ellipsoid algorithm is $O(k^2 L)$ and its running time is $O(Tk^2 L + k^4 L)$. Note that the Ellipsoid algorithm does not benefit from fast matrix multiplication. So, overall Vaidya's algorithm has a better worst-case time complexity than the ellipsoid method. For this reason we will use Vaidya's algorithm in the rest of the paper.

For problem (P_1) let n be the number of variables, let m be the number of constraints $Ax = b$ and let $T(n)$ be the number of arithmetic operations to solve the separation problem (SEP). Then, the number of variables in (D_1) is $k = m+1$ and thus from (2) the overall time complexity to find z_1^* and the optimal dual solution y^*, σ^* of (D_1) is $O(T(n)mL+M(m)mL)$. We summarize the previous developments in the following theorem.

Theorem 1 *The optimal solution value z_1^* of problem (P_1) and the optimal dual variables y^*, σ^* can be found in $O(T(n)mL + M(m)mL)$ arithmetic operations.*

A natural question is whether we can also find the optimal primal solution x^* of (P_1). For example, since our algorithm takes $O(mL)$ calls to the separation problem (SEP), there will be $O(mL)$ constraints in (D_1) and correspondingly $O(mL)$ variables λ_k, θ_k in (P_1'). Applying an interior point algorithm in (P_1') we can find the optimal λ_k^*, θ_k^* in $O((mL)^3 L)$, from which we can compute the optimal solution x^*. Unfortunately, this running time is not attractive except for problems where $L = O(1)$. Moreover, if the optimal solutions of problems (D_1), (P_1') are unique we can apply the complementary slackness property and thus we can find the optimal x^* in $O(mL)$, which does not change the overall complexity of the method. In applications, however, we want to find the optimal solution value rather than the solution of the LP or the Lagrangean relaxation, since the solution value can be later used in a branch and bound algorithm.

2.2 LPs with more than one subproblems

We now generalize the technique of the previous subsection to handle problems of the form:

$$(P_2)\ z_2 = Min \sum_{r=1}^{N} c_r x_r \tag{3}$$

subject to

$$\sum_{r=1}^{N} A_r x_r = b$$

$$x_r \in S_r,\ r = 1, \ldots, N$$

where each S_r is a polyhedron with the property that optimizing over S_r is easy.

Let x_r^k, $k \in J_r$ be the extreme points of S_r, and similarly let w_r^k, $k \in R_r$ be the extreme rays of S_r. Using the same technique as before we find that problem (P_2) is equivalent to problem (D_2)

$$(D_2)\ z_2 = Max\ yb + \sum_{r=1}^{N} \sigma_r$$

subject to

$$y(A_1 x_1^k) + \sigma_1 \leq c_1 x_1^k, \quad k \in J_1,$$

$$\ldots$$

$$y(A_N x_N^k) + \sigma_N \leq c_N x_N^k, \quad k \in J_N,$$

$$y(A_1 w_1^k) \leq c_1 w_1^k, \quad k \in R_1,$$

$$\ldots$$

$$y(A_N w_N^k) \leq c_N w_N^k, \quad k \in R_N.$$

In order to apply Vaidya's algorithm to (D_2) we should be able to solve efficiently the separation problem, which in this case decomposes to the N subproblems:

$$(SEP_r)\ Min\ (c_r - yA_r)x_r$$

subject to

$$x_r \in S_r.$$

As a result, Vaidya's algorithm applied to (D_2) with separation problems (SEP_r), $r = 1, \ldots, N$ computes the optimal value z_2^* of (D_2) (which is also equal to the optimal solution value of (P_2)) and the optimal dual solution y^*, σ_r^*, $r = 1, \ldots, N$.

Let n_r be the number of variables in S_r, let m be the number of constraints $\sum_{r=1}^{N} A_r x_r = b$ and let $T_r(n_r)$ be the number of arithmetic operations to solve the separation problem (SEP_r). Therefore, the total number of arithmetic operations to solve the entire separation problem is $\sum_{r=1}^{N} T_r(n_r)$. Since the number of variables in (D_2) is $k = m + N$, we obtain from (2) that the overall time complexity to solve (D_2) is

$$O([\sum_{r=1}^{N} T_r(n_r)](m+N)L + M(m+N)(m+N)L).$$

Therefore,

Theorem 2 *The optimal solution value z_2^* of problem (P_2) and the optimal dual variables y^*, σ_r^*, $r = 1, \ldots, N$ can be found in $O([\sum_{r=1}^{N} T_r(n_r)](m+N)L + M(m+N)(m+N)L)$ arithmetic operations.*

2.3 Cost Splitting

We now consider LPs, in which the feasible region is the intersection of K polyhedra. Examples in this category are the K-matroid and K-polymatroid intersection problems, LP relaxation problems of combinatorial problems with this property, etc. In order to speed up algorithms for the solution of such problems we combine our technique with cost splitting or Lagrangean decomposition (see for example Nemhauser and Wolsey [25], p.333), a method which has been applied to strengthen the bounds given by Lagrangean relaxation. Consider the LP

$$(P_3)\ z_3 = Min\ cx \tag{4}$$

subject to

$$x \in S_1 \cap S_2 \cap \ldots S_K,$$

where $S_1 \ldots S_K$ are polyhedra, over which we can optimize easily. We rewrite the problem as follows:

$$(P_3')\ z_3 = \ Min\ cx_1 \tag{5}$$

subject to

$$x_1 - x_2 = 0$$

$$x_2 - x_3 = 0$$

$$\ldots$$

$$x_{k-1} - x_k = 0$$

$$x_1 \in S_1,\ x_2 \in S_2,\ \ldots\ , x_K \in S_K.$$

Let x_r^k, $k \in J_r$ and w_r^k, $k \in R_r$ be the extreme points and extreme rays of S_r, $(r = 1, 2, \ldots, K)$. Applying the resolution theorem to the polyhedra S_r, $(r = 1, 2, \ldots, K)$ we rewrite (P_3') interms of x_r^k and w_r^k. Taking the dual of (P_3') we obtain

$$(D_3)\ z_3 = Max\ \sigma_1 + \sigma_2 + \ldots + \sigma_K$$

subject to

$$yx_1^k + \sigma_1 \leq cx_1^k, \quad k \in J_1,$$

$$-yx_2^k + \sigma_2 \leq 0, \quad k \in J_2,$$

$$\ldots$$

$$-yx_K^k + \sigma_K \leq 0, \quad k \in J_K,$$

$$yw_1^k \leq cw_1^k, \quad k \in R_1,$$

$$yw_2^k \leq 0, \quad k \in R_2.$$

$$\ldots$$

$$yw_K^k \leq 0, \quad k \in R_K.$$

In order to apply Vaidya's algorithm to (D_3) we should be able to solve efficiently the separation problem, which in this case decomposes to the K subproblems:

$$(SEP_1)\ Min\ (c - y)x_1$$

subject to

$$x_1 \in S_1$$

and for $r = 2, \ldots, K$

$$(SEP_r)\ Min\ yx_r$$

subject to

$$x_r \in S_r.$$

As a result, Vaidya's algorithm applied to (D_3) with separation problems (SEP_r), $r = 1, \ldots, K$ computes the optimal value z_3^* of (D_3) (which is also equal to the optimal solution

value of (P_3)) and the optimal dual solution y^*, σ_r^*, $r = 1, \ldots, K$. In order to analyze the running time of the algorithm let n be the number of variables, and let $T_r(n)$ be the number of arithmetic operations to solve the separation problem (SEP_r) respectively. From (2) with $k = Kn$ we obtain:

Theorem 3 *The optimal solution value z_3^* of problem (P_3) and the optimal dual variables y^*, σ_1^*, σ_2^*,... σ_K^*, can be found in $O([\sum_{r=1}^{K} T_r(n)]\ KnL + M(Kn)KnL)$ arithmetic operations.*

The cost splitting approach will be superior than applying Vaidya's method directly to (P_3) whenever the separation problem over S_r is more difficult than the optimization problem over S_r. Examples in this category include polymatroid polytopes. As a result, we will see that the cost splitting approach leads to significant improvements in the K-polymatroid intersection problem.

2.4 Applications to Lagrangean relaxation

Lagrangean relaxation is a primary method used in practice to find good bounds for combinatorial optimization problems. Consider the integer programming problem:

$$z_{IP} = \ Min\ cx \tag{6}$$

subject to

$$Ax \leq b$$

$$x \in S = \{x \in Z_+^n \ : A_1 x \leq b_1\}.$$

Suppose we want to solve the Lagrangean dual

$$(P_4)\ z_{LD} = Max_{\lambda \geq 0}\ Min_{x \in S}\ [cx + \lambda(b - Ax)].$$

It is well known that $z_{LD} = Min\{cx :\ Ax \leq b,\ x \in conv(S)\}$ and also $z_{LP} \leq z_{LD} \leq z_{IP}$, i.e., the Lagrangean dual gives better bounds than the LP relaxation (see for example Nemhauser and Wolsey [25], p. 327) .

In order to find z_{LD} we rewrite (P_4) as follows:

$$(P_4')\ z_{LD} = \ Max\ w$$

$$w \leq cx + \lambda(b - Ax) \text{ for all } x \in S$$

$$\lambda \geq 0.$$

In order to apply Vaidya's algorithm to (P_4') we need to solve the following separation problem: Given (w, λ), with $\lambda \geq 0$

$$(SEP)\ Max\ (c - \lambda A)x \tag{7}$$

subject to

$$x \in S.$$

If $T(n)$ is the number of arithmetic operations to solve the separation problem (7) and there m constraints $Ax \leq b$, then the application of Vaidya's algorithm to the reformulation (P_4') leads to:

Theorem 4 *The optimal solution value of the Lagrangean dual z_{LD} of problem (P_4) and the optimal Lagrange multipliers λ^*, can be found in $O(T(n)mL + M(m)mL)$ arithmetic operations.*

2.5 Variable relaxation

We will now consider LPs where instead of complicating constraints we have complicating variables. Our goal is again to speed up the computation. Consider the LP

$$(P_5)\ z_5 = \ Min\ cx + dy \tag{8}$$

subject to

$$Ax + By \geq b$$

$$x \geq 0,\ y \geq 0.$$

Suppose that the problem is easy to solve whenever y is fixed. Examples in this category are LP relaxations of fixed charge network design problems. We write problem (P_5) as follows:

$$z_5 = Min_{y \geq 0} \, [dy + Min_{x: Ax \geq b - By, \; x \geq 0} \, cx]$$

Taking the dual in the inner minimization we obtain:

$$z_5 = Min_{y \geq 0} \, [dy + Max_{\pi A \leq c, \pi \geq 0} \, \pi(b - By)]$$

which can be written as follows:

$$z_5 = Min \; dy + \sigma \tag{9}$$

subject to

$$\pi(b - By) \leq \sigma \quad \text{for all } \pi \in \{\pi : \; \pi A \leq c, \pi \geq 0\}.$$

We will solve problem (9) using Vaidya's algorithm. Given a (y, σ), the separation problem is

$$(SEP) \; Max \; \pi(b - By)$$

subject to

$$\pi A \leq c$$

$$\pi \geq 0$$

which by taking the dual is equivalent to

$$(SEP') \; Min \; cx$$

subject to

$$Ax \geq b - By$$

$$x \geq 0.$$

If in the original problem (P_5) $x \in R^n_+$ and $y \in R^m_+$ and $T(n)$ is the number of arithmetic operations to solve the separation problem (SEP') we obtain from (2) that Vaidya's algorithm takes $O(T(n)mL + M(m)mL)$. Note that in this case the algorithm not only produces

the optimal solution value, but in addition it finds the optimal y^*. Given the optimal y^*, we can solve problem (SEP') in $T(n)$ arithmetic operations to find the optimal solution x^* as well. Therefore, in this case we can derive the the optimal solution value as well as the optimal solution. Therefore,

Theorem 5 *The optimal solution value z_5^* of problem (P_5) and the optimal solution x^*, y^* can be found in $O(T(n)mL + M(m)mL)$ arithmetic operations.*

2.6 Summary of algorithms

In this section we summarize our findings in order to facilitate the reading and for future reference. In table I we summarize the problem type we considered and the separation algorithm we need to solve. Table II includes the running times. $T(n)$ always refers to the time to solve the separation problem in table I and $M(n)$ is the number of arithmetic operations to multiply two $n \times n$ matrices.

Table I: Problem type and its separation problem

Problem	Separation Problem
(P_1) $Min\ cx,\ s.t.\ Ax = b,\ x \in S$	$Min\ c'x,\ s.t.\ x \in S$
(P_2) $Min\ \sum_{r=1}^{N} c_r x_r,\ s.t.\ \sum_{r=1}^{N} A_r x_r = b,\ x_r \in S_r$	$Min\ c_r' x_r,\ s.t.\ x_r \in S_r$
(P_3) $Min\ cx,\ s.t.\ x \in S_1 \cap S_2 \ldots \cap S_K$	$Min\ c'x,\ s.t.\ x \in S_r$
(P_4) $Max_{\lambda \geq 0}\ Min_{x \in S}[cx + \lambda(b - Ax)]$	$Min\ c'x,\ s.t.\ x \in S$
(P_5) $Min\ cx + dy,\ s.t.\ Ax + By \geq b,\ x, y \geq 0$	$Min\ c'x,\ s.t.\ Ax \geq b - By,\ x \geq 0$

Table II: Running times

Problem	Running time
(P_1) $Min\ cx,\ s.t.\ Ax = b,\ x \in S$	$O(T(n)mL + M(m)mL)$
(P_2) $Min\ \sum_{r=1}^{N} c_r x_r,\ s.t.\ \sum_{r=1}^{N} A_r x_r = b,\ x_r \in S_r$	$O([(\sum_{r=1}^{N} T_r(n_r)) + M(m+N)](m+N)L)$
(P_3) $Min\ cx,\ s.t.\ x \in S_1 \cap S_2 \ldots \cap S_K$	$O((\sum_{r=1}^{K} T_r(n))KnL + M(Kn)KnL)$
(P_4) $Max_{\lambda \geq 0}\ Min_{x \in S}[cx + \lambda(b - Ax)]$	$O(T(n)mL + M(m)mL)$
(P_5) $Min\ cx + dy,\ s.t.\ Ax + By \geq b,\ x, y \geq 0$	$O(T(n)mL + M(m)mL)$

3 Applications

In this section we apply the theorems of the previous sections to solve LP and Lagrangean relaxations of several classical combinatorial optimization problems. Our intention is not to exhaust all the possible applications of our methods. It is rather to illustrate the significant computational savings that can result from our approach to some of the classical problems in combinatorial optimization. We start our investigation with the traveling salesman problem (TSP):

3.1 The Held and Karp lower bound for the TSP

Held and Karp [16] proposed a Lagrangian relaxation based on the notion of 1-tree for the TSP. Using a complete characterization of the 1-tree polytope, they showed that this Lagrangian relaxation gives the same bound as the LP relaxation of the following classical formulation of the TSP:

$$Min \sum_{i \in V} \sum_{j \in V, j>i} c_{ij} x_{ij} \tag{10}$$

subject to

$$\sum_{j \in V, j>i} x_{ij} + \sum_{j \in V, j<i} x_{ji} = 2 \qquad \forall i \in V \tag{11}$$

$$\sum_{i \in S} \sum_{j \in S, j>i} x_{ij} \leq |S| - 1 \qquad \forall \emptyset \neq S \subset V \tag{12}$$

$$0 \leq x_{ij} \leq 1 \qquad \forall i, j \in V, j > i \tag{13}$$

$$x_{ij} \in \{0, 1\} \qquad \forall i, j \in V, j > i \tag{14}$$

In the above formulation x_{ij} indicates whether cities i and j are adjacent in the optimal tour; c_{ij} represents the cost of traveling from city i to city j or, by symmetry, from city j to city i. One can compute the Held-Karp lower bound z_{HK} in polynomial time using the Ellipsoid or Vaidya's algorithm directly, since the separation problem corresponding to the polytope (11)-(13) reduces to a max-flow problem. Using this approach Vaidya's algorithm,

which is the fastest of the two, will take $O(n^3n^2L+M(n^2)n^2L)$ arithmetic operations, since from (2) there are $k = n^2$ variables and $T(n) = O(n^3)$ is the time to solve a max-flow problem on a possibly complete graph. Thus this approach takes $O(n^{6.76}L)$ where we used the bound that $M(k) = O(k^{2.38})$. Note that the ellipsoid algorithm will be even worse taking $O(n^8L)$.

The HK polytope (11)-(13) is exactly in the form of problem (P_1), where S is the spanning tree polytope described by the constraints (12), (13). Applying theorem 1 we obtain that using our approach we can compute z_{HK} in $O(n^2nL + M(n)nL) = O(n^{3.38}L)$ arithmetic operations, since the separation problem is a minimum spanning tree computation, i.e., $T(n) = O(n^2)$ and the number of extra constraints is $m = n$. As a result, our approach leads to the fastest known algorithm for HK. Moreover, if one does not use fast matrix multiplication (i.e., $M(k) = k^3$), then our approach leads to $O(n^4L)$ time complexity while Vaidya's or the Ellipsoid method take $O(n^8L)$, a savings of $O(n^4)$.

3.2 The Steiner tree and the k-connected problem

Goemans and Bertsimas [13] prove that under the triangle inequality, the cost of the LP relaxation of the Steiner tree problem $z_{steiner}$ and the k-connected problem z_{k-conn} are related in the following way

$$z_{Steiner} = \frac{1}{2}z_{HK}, \quad z_{k-conn} = \frac{k}{2}z_{HK},$$

where z_{HK} is the cost of the Held-Karp lower bound for the TSP. Therefore, if the cost satisfy the triangle inequality, we can apply the algorithm of the previous subsection to compute the value of the LP relaxation of the Steiner tree problem and the k-connected problem.

3.3 The vehicle routing problem

Consider the following classical vehicle routing problem: There is a set A of K vehicles, located at a single depot 0, such that the kth vehicle has capacity Q_k and is allowed to

travel a distance of at most d_k. These vehicles serve a set V of n customers. Customer i has demand p_i, while c_{ij}^k is the cost of vehicle k traveling from i to j and d_{ij} is the distance from i to j. The goal is to route the vehicles at minimum cost such that all constraints are satisfied. We formulate the problem as follows:

Let x_{ij}^k be 1 if vehicle k travels from i to j and 0 otherwise. Let S_k be the following polytope

$$S_k = \{x_{ij}^k \mid \sum_{i \in S} \sum_{j \in S, j>i} x_{ij}^k \leq |S| - 1 \; \forall \, \emptyset \neq S \subset V, \; \sum_{i \in V} x_{0i}^k + \sum_{j \in V} x_{j0}^k = 2\}.$$

The polytope S_k is the intersection of the spanning tree polytope on V (note that V does not include the depot 0) and and an additional constraint that 2 additional arcs are incident to the depot. For fixed k we denote all the x_{ij}^k's as the vector x^k. We are interested to compute the LP relaxation of the following formulation of VRP:

$$(VRP) \; Min \sum_{i,j,k} c_{ij}^k x_{ij}^k \tag{15}$$

subject to

$$\sum_{i \in V} \sum_{k \in A} x_{ij}^k = 1 \quad \forall j \neq 0 \tag{16}$$

$$\sum_{j \in V} \sum_{k \in A} x_{ij}^k = 1 \quad \forall i \neq 0 \tag{17}$$

$$\sum_{i,j \in V} x_{ij}^k d_{ij} \leq d_k \quad \forall k \in A \tag{18}$$

$$\sum_{i,j \in V} x_{ij}^k p_i \leq Q_k \quad \forall k \in A \tag{19}$$

$$x^k \in S_k \tag{20}$$

$$0 \leq x_{ij}^k \leq 1 \tag{21}$$

$$x_{ij}^k \in \{0, 1\} \tag{22}$$

In order to solve the LP relaxation of the above problem we will apply our approach of section 2.2. The above formulation is of the type of problems (P_2), where there are $N = K$ subproblems S_k, and the number of additional constraints (16)-(19) is $m = 2(n + K)$. Since we can optimize over S_k using the greedy algorithm, the time to solve the separation problem is $O(n^2)$. Thus, applying theorem 2, we can solve the LP relaxation of (VRP) in $O(Kn^2(n+K)L + M(n+K)(n+K)L) = O((n^3K + n^{3.38})L)$, since we can assume $K \leq n$.

For comparison if we applied Vaidya's algorithm directly to (15) it would lead to an $O(Kn^3(Kn^2)L + M(Kn^2)(Kn^2)L)$=$O(K^{3.38}n^{6.76}L)$ algorithm, since there are Kn^2 variables and the separation problem reduces to K max-flow problems.

3.4 Multicommodity flows

Consider the classical multicommodity flow problem: Given a network $G = (V, E)$ ($|V| = n$, $|E| = m$), a set C of K commodities and a supply (or demand) b_i^k of commodity k at node $i \in V$. Arc (i, j) has capacity u_{ij} and the cost of sending one unit of flow of commodity k across arc (i, j) is c_{ij}^k. The goal is to decide the amount of flow x_{ij}^k from commodity k to send across arc (i, j), so as to satisfy supply-demand and capacity constraints at minimum cost. The classical formulation of the problem is:

$$Min \sum_{i,j \in V,\ k \in C} c_{ij}^k x_{ij}^k \tag{23}$$

subject to

$$\sum_{k \in C} x_{ij}^k \leq u_{ij} \quad (i,j) \in E \tag{24}$$

$$\sum_{j \in V} x_{ij}^k - \sum_{j \in V} x_{ji}^k = b_i^k \quad i \in V,\ k \in C \tag{25}$$

$$x_{ij}^k \geq 0. \tag{26}$$

The above formulation is of the type (P_2) in section 2.2, with

$$S_k = \{x_{ij}^k |\ x_{ij}^k \geq 0,\ \sum_{j \in V} x_{ij}^k - \sum_{j \in V} x_{ji}^k = b_i^k \quad i \in V\}$$

and with m global constraints (24). In this case the separation problem is a min-cost flow problem. Under the similarity assumption, the time to solve the separation problem is $T = O(nmlog^2 n)$. For a more refined definition of T see, for example, Ahuja et. al. [1].

As a result, applying theorem 2 we can solve the multicommodity flow problem in

$$O([Kmnlog^2 n + M(m+K)](m+K)L) = O([Knm^2 log^2 n + m^{3.38}]L).$$

The comparison for multicomodity flows is more complex because of another algorithm of Vaidya [28]. He considered the problem in which each commodity consists of a single source and a single sink. The resulting running time is $O(K^{2.5}n^2\sqrt{m}L)$. This running time dominates ours in some cases and is dominated by ours in others. For example, for $K = n^2$ Vaidya's running time is $O(n^7\sqrt{m}L)$. However, our algorithm runs in $O([n^2m^2log^2 n + m^{3.38}]L)$ time in this case, since the problem can be converted into a n-commodity flow problem. In this case, our algorithm dominates Vaidya's for all values of m, and is increasingly better as the graph becomes sparser.

3.5 Network flows with side constraints

Typical problems met in practice involve network flow problems with some additional side constraints. Consider for example a min-cost flow problem with K additional constraints. Using these K constraints as the global constraints in the formulation of problem (P_1) in section 2.1 and the network flow polytope

$$\{x_{ij} |\ x_{ij} \geq 0,\ \sum_{j \in V} x_{ij} - \sum_{j \in V} x_{ji} = b_i \quad i \in V\}$$

as the polytope S we can apply theorem 1 to solve the problem in $O([mnlog^2 n + M(K)]KL)$, where $T = mnlog^2 n$ is an upper bound on the complexity to solve the network flow problem under the similarity assumption (see [1]). For K constant we see that the complexity of solving the problem *with K side constraints* is exactly the same as the complexity of solving the problem *with only one side constraint*, i.e., $O((mnlog^2 n)L)$.

If one applied an interior point algorithm for this problem, it would lead to an $O(m^3L)$ running time.

3.6 The network design problem

The fixed charge network design problem, which is a fundamental discrete choice design problem, is useful for a variety of applications in transportation, distribution, communication, and several other problem settings that make basic cost tradeoffs between operating costs and fixed costs for providing network facilities (see Magnanti and Wong [24]). The problem is described as follows. We are given a set of nodes N, a set of uncapacitated arcs A and a set K of commodities. For each $k \in K$, one unit of flow of commodity k must be sent from its origin $O(k)$ to its destination $D(k)$. Each arc has two types of cost: a per unit flow cost depending on the commodity and a fixed charge for using the arc. The problem is to select a subset of arcs that minimizes the sum of the routing costs and fixed charge costs.

The importance of the network design problem stems from its wide applicability and flexibility. As noted in Magnanti and Wong [24], it contains a number of well-known network optimization problems as special cases including the shortest path, minimum spanning tree, uncapacitated plant location, traveling salesman and Steiner tree problems.

There are a number of IP formulations for the problem. For a review, see Magnanti and Wong [24]. Balakrishnan et al. [3] propose the following multicommodity flow formulation, which contains two types of variables, one modeling discrete design choices and the other continuous flow decisions. Let y_{ij} be a binary variable that indicates whether ($y_{ij} = 1$) or not ($y_{ij} = 0$) arc $\{i, j\}$ is chosen as part of the network's design. Let x_{ij}^k denote the flow of commodity k on the directed arc (i, j). Note that (i, j) and (j, i) denote directed arcs with opposite orientations corresponding to the undirected arc $\{i, j\}$. Even though arcs in the formulation are undirected, we refer to the directed arcs (i, j) and (j, i) because the flows are directed. The formulation is the following.

$$Min \quad \sum_{k \in K} \sum_{\{i,j\} \in A} (c_{ij}^k x_{ij}^k + c_{ji}^k x_{ji}^k) + \sum_{\{i,j\} \in A} F_{ij} y_{ij} \tag{27}$$

subject to

$$\sum_{j \in V:(j,i) \in A} x_{ji}^k - \sum_{j \in V:(i,j) \in A} x_{ij}^k = \begin{cases} -1 & i = O(k) \\ 1 & i = D(k) \\ 0 & \text{otherwise} \end{cases} \tag{28}$$

$$(DP_1) \qquad x_{ij}^k \leq y_{ij}, \quad x_{ji}^k \leq y_{ij} \qquad \{i,j\} \in A,\ k \in K \tag{29}$$

$$x_{ij}^k \geq 0 \qquad \{i,j\} \in A,\ k \in K$$

$$y_{ij} \in \{0,1\} \qquad (i,j) \in A.$$

In this formulation each arc $\{i,j\}$ has a nonnegative fixed design cost F_{ij} and c_{ij}^k is the nonnegative cost for routing commodity k on the directed arc (i,j). Constraints (28) imposed upon each commodity k are the usual network flow conservation equations. The "forcing" constraints (29) state that if $y_{ij} = 0$, i.e., arc $\{i,j\}$ is not included in the design, then the flow of every commodity k on this arc must be zero in both directions, and if arc $\{i,j\}$ is included in the design, i.e., $y_{ij} = 1$, the arc flow is unlimited. Directed network design problems are formulated in a very similar manner.

Although the network design problem is a hard discrete optimization problem, in the last decade researchers have proposed several computationally successful approaches for solving it. Magnanti et al. [22] propose a Benders decomposition approach, and Balakrishnan et al. [3] propose a dual ascent heuristic which has solved large instances of network design problems to within 2%-5% of optimality. In both these cases the authors judge the algorithm's effectiveness by comparing solutions generated to the LP relaxation of their formulations. It is therefore important to solve the LP relaxation efficiently.

We treat the forcing variables y_{ij} in the network design formulation (27) as the complicating variables in the sense of section 2.5. If the y_{ij} are known then the problem decomposes to K shortest paths problems. Therefore, applying theorem 5, the LP relaxation of (27) can be solved in

$$O([K(m + nlogn) + M(m)]mL),$$

where m is the number of complicating variables y_{ij}, and $O(m + nlogn)$ is the time to solve a single shortest path problem. Note that for the problems considered in [3] $K = n^2$ and $m = O(n)$ and thus our algorithm takes $O([n^2(m + nlogn)]mL)$.

For purposes of comparison if one solves the LP relaxation of (27) using an interior point algorithm, it will lead to an $O(K^3m^3L)$ running time.

3.7 Facility location problems

We consider the well known p-median problem in facility location (see for example [25]). We are interested in solving the LP relaxation of the following formulation of the problem:

$$Min \sum_{i \in V} c_{ij} x_{ij} + \sum_{j \in V} y_j \tag{30}$$

subject to

$$\sum_{j \in V} y_j = p \tag{31}$$

$$\sum_{j \in V} x_{ij} = 1 \quad i \in V \tag{32}$$

$$x_{ij} \leq y_j \quad i, j \in V \tag{33}$$

$$x_{ij} \geq 0, \ 0 \leq y_j \leq 1 \tag{34}$$

$$y_j \in \{0, 1\} \tag{35}$$

The interpretation is that y_j is 1 if node j is assigned to a facility and 0 otherwise and x_{ij} is 1 if node i is assigned to a facility j and 0 otherwise. The LP relaxation of (30) has been found to be very close to the IP solution. For this reason almost all algorithmic approaches to the problem compute first the LP relaxation.

In order to solve the LP relaxation we observe that the p-median problem is of the type of problem (P_1) with constraints (31) and (32) being the global constraints and (33) and (34) being the polytope S. We can solve the separation problem over S in $T(n) = O(n^2)$ ($|V| = n$), since we can solve it in closed form in one pass. Therefore, applying theorem 1 we can solve the LP relaxation of the p-median problem in $O(n^2nL + M(n)nL) = O(n^{3.38}L)$. For comparison purposes, if one applied an interior point algorithm directly to solve the LP relaxation of (30), it would take $O(n^6L)$ iterations since there are $O(n^2)$ variables.

For uncapacitated location problems (see for example [25]) exactly the same approach as in the case of the p-median problem leads to an $O(n^{3.38}L)$ algorithm for the solution of the LP relaxation.

3.8 The K-polymatroid intersection problem

Consider K polymatroids $M_i = (N, f_i)$ where f_i is a submodular set function. For example, if f_i is the rank function of a matroid, the problem reduces to the K-matroid intersection problem. Given costs c_j for all $j \in N$, the weighted K-polymatroid intersection problem is described by the mathematical programming problem:

$$Max \sum_{j \in N} c_j x_j \tag{36}$$

subject to

$$\sum_{j \in S} x_j \leq f_1(S) \quad \forall S \subset N \tag{37}$$

...

$$\sum_{j \in S} x_j \leq f_K(S) \quad \forall S \subset N \tag{38}$$

$$x_j \geq 0$$

Classical problems in combinatorial optimization can be modelled in that way. For example the maximum spanning tree ($K = 1$ and $f_1(S) = |S| - 1$, maximum bipartite matching ($K = 2$ and $f_i(S)$ are the rank functions of two partitioned matroids). Our goal is to solve (36) using the techniques of section 2.3.

Let $S_i = \{x \geq 0 | \sum_{j \in S} x_j \leq f_i(S) \quad \forall S \subset N\}$. Using the cost splitting method of section 2.3 and applying theorem 3 we can solve problem (36) in $O([KT(n) + M(Kn)]KnL)$, where $T(n)$ is the number of arithmetic operations to solve the optimization problem over one polymatroid, which, as it is well known (see for example Nemhauser and Wolsey [25], p.689), can be solved by the greedy algorithm as follows:

1. Sort $c_1 \geq c_2 \geq ... \geq c_k > 0 \geq c_{k+1} \geq ... \geq c_n$.
2. Let $S^j = \{1, \ldots, j\}$ with $S^0 = \emptyset$.
3. $x_j = f(S^j) - f(S^{j-1})$ for $j \leq k$ and $x_j = 0$ for $j > k$.

For purposes of comparison, an alternative approach is to apply Vaidya's algorithm to (36) directly. The separation problem is to decide whether a given $x_0 \in S_i$. Such an approach leads to a $O([KA(n)+M(n)]nL)$ arithmetic operations where $A(n)$ is the number of arithmetic operations to solve the separation problem $x_0 \in S_i$. Indeed, $A(n) = O(nL[T(n)+M(n)])$ if we use Vaidya's algorithm. Overall this approach leads to $O(K(nL)^2[T(n)+M(n)])$ arithmetic operations. Alternatively in order to solve the separation problem, one could use the strongly polynomial algorithm of Cunningham [9], but unfortunately the running time would not be as good.

Overall, we expect that our approach will work better, in problems in which the separation problem is much harder than the optimization problem.

3.9 Network flow problems on graphs with almost special structure

Consider a network flow problem on the network $G = (N, A \cup B)$ such that $G_A = (N, A)$ is a graph of special structure (for example planar, tree, unbalanced bipartite, etc.) for which the related network flow problem can be solved faster than in a general graph. For example, for network flow problems on unbalanced bipartite graphs see [2]. Assume that $|B| = K$. The network flow problem can be formulated as follows:

$$z = Min\ c_A x_A + c_B x_B \tag{39}$$

subject to

$$N_A x_A + N_B x_B = b \tag{40}$$

$$0 \leq x_A \leq u_A \tag{41}$$

$$0 \leq x_B \leq u_B \tag{42}$$

where N_A, N_B is the arc incidence matrix corresponding to the arcs in A and in B respectively.

We first observe that problem (39) is of the type of problem (P_5) in section 2.5. Applying the variable splitting algorithm of section 2.5, we obtain that problem (39) can be solved

in $O([T_A(n,m)+M(K)]KL)$ where $T_A(n,m)$ is the time to solve the network flow problem on $G_A=(N,A)$, which is a graph of special structure. Because of the special structure of G_A, $T_A(n,m)$ will be smaller than the time to solve the problem on general graphs. For K constant the running time becomes $O(T_A(n,m)L)$. For comparison purposes we will denote the running time on general graphs as $O(T_G(n,m))$.

3.10 The multiple item capacitated lot sizing problem

Consider the multiple item capacitated lot sizing problem, where x_{jt}, y_{jt} and s_{jt} represent the production, setup and storage variables for item j in period t, $d_{jt}, c_{jt}, f_{jt}, h_{jt}$ are the demand, production cost, setup cost and storage cost for item j in period t and Q_t represents the amount of the resource available in period t:

$$Min \sum_{j=1}^{K}\sum_{t=1}^{T}[c_{jt}x_{jt}+f_{jt}y_{jt}+h_{jt}s_{jt}] \tag{43}$$

subject to

$$x_{jt}+s_{j,t-1}=d_{jt}+s_{jt} \quad j=1,\dots K,\ t=1,\dots T \tag{44}$$

$$s_{j0}=s_{jT}=0 \quad j=1,\dots K \tag{45}$$

$$x_{jt}\le(\sum_{r=t}^{T}d_{jr})y_{jt} \quad j=1,\dots K,\ t=1,\dots T \tag{46}$$

$$\sum_{j=1}^{K}x_{jt}\le Q_t \quad t=1,\dots T \tag{47}$$

$$x_{jt}, s_{jt}\ge 0 \quad y_{jt}\in\{0,1\}, \tag{48}$$

The multiple item capacitated lot sizing problem is NP-hard. If we relax constraints (47) the problem decomposes into K single item capacitated lot sizing problems, each of which can be solved by a dynamic programming algorithm, that has $O(T)$ running time. Applying the algorithm of section 2.4 we can find the value of the Lagrangean dual in

$$O([KT+M(T)]TL)=O([KT^2+T^{3.38}]L).$$

Note that the value of the Lagrangean dual can be strictly better that the value of the LP relaxation, since the subproblem does not have the integrality property. To the best of our knowledge we do not know any other polynomial time approach for the problem. For comparison, the solution of the LP relaxation of (43), which gives a weaker bound than the Lagrangean dual, takes $O(K^3T^3L)$ using an interior point approach.

3.11 Comparisons with the previously known fastest method

In order to facilitate the comparison of our methods with the previously fastest known methods we include table III. We assumed that we could use fast matrix multiplication. The problems refer to the LP relaxation or the Lagrangean dual.

Table III: Comparisons

Problem	Our running time	Best previuosly known
Held-Karp	$O(n^{3.38}L)$	$O(n^{6.76}L)$
Steiner tree, k-connected	$O(n^{3.38}L)$	$O(n^{6.76}L)$
K-vehicle routing	$O([n^{3.38}+n^3K]L)$	$O(K^{3.38}n^{6.76}L)$
K-multicommodity flow	$O([Knm^2log^2n+m^{3.38}]L)$	$O(K^{2.5}n^2\sqrt{m}L)$
Network flows with constraints	$O([Knmlog^2n+K^{3.38}]L)$	$O(m^3L)$
Network design	$O([K(m+nlogn)+m^{2.38}]mL)$	$O(K^3m^3L)$
Facility location problems	$O(n^{3.38}L)$	$O(n^6L)$
K-polymatroid intersection	$O([KT(n)+M(Kn)]KnL)$	$O(K(nL)^2[T(n)+M(n)])$
Network flows on $G(V,A\cup B)$	$O(T_A(n,m)L)$	$O(T_G(n,m))$
capacitated lot sizing	$O([KT^2+T^{3.38}]L)$	?

4 Concluding remarks

The previous section was simply an indication of the variety of different applications our method has in combinatorial optimization. One can certainly find other applications in other areas of combinatorial optimization (for example scheduling and sequencing). Our technique can also be used to solve stronger Lagrangean duals using Lagrangean decomposition.

Although our techniques lead to the fastest known algorithms for several problems from a worst-case perspective by a significant margin, we are not certain whether our techniques can be competive from a practical standpoint with the classical methods to solve the Lagrangean dual, like subgradient optimization. The practicality of our algorithm critically depends on whether Vaidya's algorithm is a practical algorithm, and to our knowledge Vaidya's algorithm has not been tested in practice. We believe, however, that perhaphs a combination of our techniques with the classical methods to solve the Lagrangean dual can potentially lead to practical algorithms as well.

Acknowledgements

The research of D. Bertsimas was partially supported by the National Science Foundation research initiation grant DDM-9010332 and and DDM-9014751. The research of J. Orlin was partially supported by the Air Force Office of Scientific Research AFOSR-88-0088, NSF grant DDM-8921835 and grants from UPS, Prime Computer Corp. and Draper Laboratories.

References

[1] R. Ahuja, A. Goldberg, J. Orlin and R. Tarjan, "Finding minimum-cost flows by double scaling", to appear in *SIAM journal of computing*, (1991).

[2] R. Ahuja, J. Orlin, C. Stein and R. Tarjan, "Improved algorithms for bipartite network flow", "Improved algorithms for bipartite network flows", Sloan working paper 3218-90-MS, November 1990.

[3] A. Balakrishnan, T.L. Magnanti and R.T. Wong , "A dual-ascent procedure for large-scale uncapacitated network design", *Operations Research*, 5, 716-740, (1989).

[4] F. Barahona, M. Grötschel, M. Jünger and G. Reinelt, "An application of combinatorial optimization to statistical physics and circuit layout design", *Operations Research*, 36, 493-513, (1988).

[5] I. Barany, T.J. Van Roy and L.A. Wolsey, "Uncapacitated lot sizing: the convex hull of solutions", *Mathematical Programming Study*, 22, 32-43, (1984).

[6] D. Coppersmith and S. Winograd, "Matrix multiplication via arithmetic progressions", *Proc. 19th Annual ACM Symp. Theory of Computing*, (1987), 1-6.

[7] H. Crowder, E.L. Johnson and M.W. Padberg, "Solving large scale zero-one linear programming problems", *Operations Research*, 31, 803-834 (1983).

[8] H. Crowder and M.W. Padberg, "Solving large scale symmetric traveling salesman problems to optimality", *Management Science*, 26, 495-509 (1980).

[9] W. Cunningham, "Testing membership in matroid polyhedra", Journal of Combinatorial Theory (B), 36, (1984), 161-188.

[10] G.D. Eppen and R.K. Martin, "Solving multi-item capacitated lot sizing problems using variable redefinition", *Operations Research*, 35, 832-848 (1987).

[11] M. Fisher, "The Lagrangean relaxation method for solving integer programming problems", *Management Science*, 27, 1-18 (1981).

[12] Geoffrion A., "Lagrangean relaxation and its uses in integer programming", *Math. Progr. Study*, 2, 82-114, (1974).

[13] M.X. Goemans and D. J. Bertsimas, "Survivable networks, linear programming relaxations and the parsimonious property", Operations Research Center technical report, MIT , OR 225-90, sumbitted for publication, (1990).

[14] M. Grötschel, M. Jünger and G. Reinelt, "A cutting plane algorithm for the linear ordering problem", *Operations Research*, 32, 1195-1220 (1984).

[15] M. Grötschel, L. Lovász and A. Schrijver, *Geometric Algorithms and Combinatorial Optimization*, Springer-Verlag, 1988.

[16] M. Held and R.M. Karp, "The traveling-salesman problem and minimum spanning trees", *Operations Research*, 18, 1138-1162 (1970).

[17] M. Held and R.M. Karp, "The traveling-salesman problem and minimum spanning trees: Part II", *Mathematical Programming*, 1, 6-25 (1971).

[18] M. Held, P. Wolfe and H. Crowder, "Validation of subgradient optimization", *Mathematical Programming*, 6, 62-88, (1974).

[19] E.L. Johnson, M.M. Kostreva and H.H. Suhl, "Solving 0-1 integer programming problems arising from large scale planning models", *Operations Research*, 33, 802-820 (1985).

[20] L. G. Khachian, "A polynomial algorithm for linear programming", *Soviet Mathematics Doklady*, 20, 191-194, (1979).

[21] J. Leung, T. L. Magnanti and R. Vachani, "Facets and algorithms for capacitated lot sizing", to appear in *Mathematical Programming* (1989).

[22] T.L. Magnanti, P. Mireault and R.T. Wong, "Tailoring Benders decomposition for uncapacitated network design", *Mathematical Programming Study*, 26 ,112-154 (1986).

[23] T.L. Magnanti and R. Vachani, "A strong cutting plane algorithm for production scheduling with changeover costs", working paper OR173-87, Operations Research Center, revised March 1989.

[24] T.L. Magnanti and R.T. Wong, "Network design and transportation planning: Models and algorithms", *Transportation Science*, 18, 1-56 (1984).

[25] G.L. Nemhauser and L.A. Wolsey, *Integer and combinatorial optimization*, John Wiley & Sons (1985).

[26] M.W. Padberg and S. Hong, "On the symmetric traveling salesman problem: a computational study", *Mathematical Programming Study*, 12, 78-107 (1980).

[27] M.W. Padberg and G. Rinaldi, "Optimization of a 532-city symmetric traveling salesman problem by branch and cut", *Operations Research Letters*, 6, 1-7 (1987).

[28] P. Vaidya, "Speeding-up linear programming using fast matrix multiplication", Proceedings of the 30th Symposium on the Foundations of Computer Science, 332-337, (1989).

[29] P. Vaidya, "A new algorithm for minimizing convex functions over convex sets", AT&T Bell Laboratories (1990).