(2)

AD-A219 013

ON PAGE

Form Approved
OMB No. 0704-0188

1. AGENCY USE ONLY (Leave blank)	2. REPORT DATE	3. REPORT TYPE AND DATES COVERED
		Final Report, 1 Jul 87 thru 31 Oct 89

4. TITLE AND SUBTITLE
LINEAR PROGRAMMING TOOLS FOR INTEGER PROGRAMMING

5. FUNDING NUMBERS
AFOSR-87-0276
61102F 2304/B1

6. AUTHOR(S)
Robert E. Bixby

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
Rice University
Department of Mathematical Sciences
P. O. Box 1892
Houston, TX 77251

8. PERFORMING ORGANIZATION REPORT NUMBER
AFOSR-TR- 90-0313

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)
AFOSR/NM
Building 410
Bolling AFB, DC 20332-6448

10. SPONSORING/MONITORING AGENCY REPORT NUMBER
AFOSR-87-0276

11. SUPPLEMENTARY NOTES

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

12b. DISTRIBUTION CODE

13. ABSTRACT *(Maximum 200 words)*

There has been significant progress in two areas: solution of the maximum-weight cut problem and development of simplex-based tools for integer programming. Codes developed have been widely used to improve solution time.

14. SUBJECT TERMS	15. NUMBER OF PAGES
	3
	16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT	18. SECURITY CLASSIFICATION OF THIS PAGE	19. SECURITY CLASSIFICATION OF ABSTRACT	20. LIMITATION OF ABSTRACT
UNCLASSIFIED	UNCLASSIFIED	UNCLASSIFIED	UL

Final Report on AFOSR Grant #870276

Linear Programming Tools for Integer Programming

Principal Investigator: Robert E. Bixby, Rice University

1 Results on the Maximum–weight Cut Problem

The motivation for this work has been the need for a practical procedure to solve the maximum–weight cut problem (MCP) in undirected graphs. Our primary focus has been on problems arising from considerations in statistical mechanics. These problems are typically posed on grid graphs and some natural variants. Though heuristic methods such as simulated annealing can often provide optimal or near optimal cuts, they are required to be provably optimal. Our algorithm takes the availability of good cuts into account. In short, given a cut, we change the sign of the edge–weights on the cut and attempt to find a positive weight cut with respect to the new edge–weights. If we find such a cut, we take its symmetric difference with the original cut and repeat the procedure with the new (and better) cut. If not, the given cut is optimal. The procedure for finding the positive weight cut requires us to optimize over the cone of the cut–polytope. This optimization involves approximating the cone by known facet–defining inequalities and solving the associated Linear Programming problem.

Generating good initial cuts is crucial to the success of our procedure. Presently we heuristically generate cuts with weight within 10% of optimal. These are obtained by applying neighborhood–exchange heuristics to random cuts. We are working on generating *optimal* cuts in a significant number of instances, perhaps using simulated annealing.

Our next consideration is actually solving the long sequences of generated LPs. We use CPLEX as our linear optimizer. In our algorithm, the LPs are quite dense and the simplex iterations highly degenerate: the simplex procedure begins with the 0–solution and stays at 0 till unboundedness is detected. To help deal with the degeneracy, we solve the dual LPs. Since the dual right–hand–side is no longer 0, the major problem with the dual is occasional stalling, which CPLEX can deal with. The dual formulation is made feasible by adding a column of all 1's. The corresponding variable x_0 is given cost–coefficient 1. The resulting objective function (Min x_0) value provides us with a global potential indicator: as we approach optimality, it goes down to 0.

The non–zero entries in the LPs correspond to the supports of simple cycle inequalities. Since the cycles can be long, the resulting LPs are quite dense. Typical densities are around 15%. We minimize this problem by generating the shortest possible violated cycles and by deleting superfluous columns. At the end of every call to CPLEX, all columns with non–zero reduced–costs are deleted. In addition, since most non–basic columns usually have zero reduced–cost, after every few calls to CPLEX, we delete *all* non–basic columns. The column deletion scheme is the most time–saving step in the implementation. Deleted columns are stored in a buffer, so that they can be re–examined for violation. The column buffering is an important part of our implementation, since checking violations in the buffer is typically much faster than generating violated inequalities from scratch.

The cycle generation procedure proceeds in three steps. First we examine the buffer for violated cycles. If none are found, we use a fast heuristic to generated cycles. If the heuristic fails, an exact method based on Dijkstra's shortest–path algorithm finds all violated cycles. Both the heuristic and the exact method process the graph in a breadth–first manner, and thus generate shortest violated cycles. We are presently looking for heuristics to identify facet–defining inequalities corresponding to homeomorphs of K_5. These arise in considering toroidal grids.

As input to our cutting plane algorithm, we have primarily used planar grid graphs. In addition, we have considered graphs made up of planar grids with a universal vertex. The grids have ± 1 edge–weights. Presently we are considering toroidal grids (with $\pm$ 1 edge–weights) and planar grids with edge–weights drawn from a Gaussian distribution. These cases correspond to different models

in statistical mechanics.

Preliminary computational results have been encouraging, e.g. we can now routinely solve the MCP on 900 node grids with ± 1 edge-weights. We have used the MIPS M-120 and SUN 3/50 computers for the C-language code development. We present timings obtained on the MIPS at optimization level O2. The column headings in the table are the following:

Graph	Planar grid or planar grid with universal node (grid+1).
$\|V\|$	Number of nodes.
$\|E\|$	Number of edges.
Heuristic	Value of heuristically generated initial cut.
Optimal	Value of optimal cut.
Time	Total Time to optimality in seconds (from initial cut).
Stages	Number of stages (intermediate cuts generated).
LPs	Number of LPs solved.
Columns	Number of columns generated in the dual.
Proof	Time to prove optimality (from optimal cut).

Summary of Preliminary Computational Results

Graph	$\|V\|$	$\|E\|$	Heuristic	Optimal	Time	Stages	LPs	Columns	Proof
grid	100	180	73	73	2.0	1	13	218	1.6
grid	100	180	56	58	5.0	3	46	334	2.2
grid	225	420	131	141	34.6	6	111	1039	4.4
grid	225	420	142	155	25.9	5	76	797	6.4
grid	400	740	229	250	162.4	8	262	2371	33.2
grid	400	760	253	270	91.2	8	184	1490	27.8
grid	900	1740	565	615	1905.3	18	834	8649	176.0
grid	900	1740	557	614	1992.6	17	971	8335	225.8
grid+1	401	1160	18886	20279	1392.2	12	983	7444	941.2
grid+1	401	1160	535	542	693.4	7	1139	6003	212.4

The results are to be part of a Ph. D. thesis by Sanjay Saigal expected to be completed by June 1990.

2 Tools for Integer Programming

The purpose of this work has been to develop simplex-based tools for integer programming. One of our principal goals has been to place a usable linear programming "black box", called CPLEX, in the hands of several people who are doing important integer programming research. These people include Francisco Barahona at the University of Waterloo; W.C. Cook at Bell Communications Research; Karla Hoffman at George Mason University; George Nemhauser of Georgia Tech; Manfred Padberg of New York University; Laurence Wolsey of CORE in Belgium; and, very recently, Herbert Scarf of Yale University.

In several of the cases mentioned above, the availability of a fast, robust LP solver has been essential to progress. For example, Karla Hoffman and Manfred Padberg have continued the work on sparse 0/1 problems begun by Crowder, Johnson and Padberg. Improvements by factors of up to 100 and more in the LP solution times have allowed them to change the order of magnitude of the sizes of problems that they can study. In addition, their recent work on some very difficult set-partitioning models may well not have been possible without the availability of CPLEX.

The conceptual design used by CPLEX as an integer-programming tool has been further developed in an attempt to develop a standard interface and to simplify the demands on the integer programming calling routines.

In the standard (or more standard) design, an LP is loaded into a system, solved, information retrieved, the problem possibly changed, and the process is then repeated. In the CPLEX design,

once the problem has initially loaded, the management of the problem data is essentially taken over by the CPLEX tool. Thus, for example, if a problem is modified by adding additional columns, then this is done through function calls, without explicitly reloading the entire problem. In the process, the basis factorization from the previous optimization is retained. If the optimizer is then called, it begins not only by using the old basis (which is standard) but without having to refactor the basis. If, as frequently happens, the number of iterations in this reoptimization is very small, then total solution time can be improved by a factor of 2. Further improvements along these lines are envisioned.

There have been numerous improvements in the performance of the basic optimizer, several of which are directly related to integer programming. The most notable of these is a method for dealing with *stalling*, or long sequences of degenerate iterations. This phenomenon is particularly prevalent in combinatorial optimization problems, many of which exhibit "strong integrality" in their LP relaxations.

There are a number of well known devices for dealing with stalling, and degeneracy in general. The device that has been introduced into CPLEX is a very simple one that does not change the basic algorithm, but rather changes the problem. As such, it is a general-purpose method that can be employed in conjunction with any optimizer, and can thus be used in existing systems. It works as follows: When stalling is encountered, a pseudo-random perturbation of the nontrivial upper and lower bounds of each structural variable is introduced. The perturbation is of the form lower-bound-j - ϵX, upper-bound-j + ϵX, where epsilon is in the range 10^{-3} to 10^{-8}, and X is a random variable uniformly distributed on (0,1). Problem feasibility is preserved, but at optimality the perturbation must be removed. If the basis that has been found is feasible, then it is optimal; otherwise, additional iterations are required. Remarkably, in most cases, the basis optimal for the perturbed problem is optimal for the unperturbed problem (for properly chosen ϵ).

A paper is currently being written describing the basis features of the CPLEX linear optimizer. Following completion of that paper, a second paper will be written (with E.A. Boyd) describing the special features for integer programming.

Additional work that is currently under way includes the following:

1. (with Russell Rushmeier) Development of a parallel implementation of branch-and-bound on an appropriate distributed memory architecture.

2. (with Matthew Saltzman) Basis recovery methods for interior-point methods.