Solving Integer Programs with Enumeration Cutting Planes

E. Andrew Boyd[1]

March, 1992

TR92-08

[1]This work was sponsored in part by the National Science Foundation and the Office of Naval Research under NSF grant number DDM-9101578. The author gratefully acknowledges the support of IMSL, Inc.

Report Documentation Page

Form Approved
OMB No. 0704-0188

1. REPORT DATE	2. REPORT TYPE	3. DATES COVERED
MAR 1992		**00-00-1992 to 00-00-1992**

4. TITLE AND SUBTITLE	5a. CONTRACT NUMBER
Solving Integer Programs with Enumeration Cutting Planes	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) **Computational and Applied Mathematics Department ,Rice University,6100 Main Street MS 134,Houston,TX,77005-1892**	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 **unclassified**	b. ABSTRACT **unclassified**	c. THIS PAGE **unclassified**		**24**	

Abstract

A cutting plane technique with applicability to the solution of general integer programs is presented and the computational value of this technique is demonstrated by applying it to a collection of seven difficult integer programs arising from real-world applications. Four of the seven problems are solved to optimality without the aid of branch and bound, and six of the seven problems have the gap between the value of the integer program and its linear programming relaxation closed by over 98%.

1 Introduction

Integer programs can be used to model an incredible wealth of real-world problems in the areas of transportation, logistics, communication network design and routing, scheduling, distributed system design and operation, as well as many, many other areas. Yet, while the list of applications is staggering, the generality of integer programs makes them difficult to solve both in theory and practice.

While specific classes of integer programs have been the focus of considerable research effort, algorithms for general integer programs have not received much recent attention since they can be routinely confounded by relatively small problems. However, general algorithms have begun to receive renewed attention, in part due to the availability of parallel computers which can take advantage of the natural coarse-grain parallelism provided by integer programs, but more importantly due to mathematical techniques that have only come to be appreciated for their computational possibilities in the last decade.

Many authors have contributed to the mathematical techniques which have led to a resurgence of interest in algorithms for general integer programs, but one effort that clearly stands out as a landmark work is the Lanchester prize-winning paper of Crowder, Johnson, and Padberg [6]. This paper used preprocessing and cutting plane techniques to solve a collection of 0/1 integer programs compiled by IBM from real-world client applications. Some of these problems were considered so difficult that at one time it was thought they would never be solvable under any foreseeable advances

in computer technology or mathematical techniques. Using appropriate algorithms, all of these problems can now be solved to optimality on most workstations in times ranging from a few seconds to an hour depending on the size of the problem and the algorithm used to solve it. The techniques used to solve these problems have begun to find their way into research and commercial codes such as Georgia Tech's GT-MIO and IBM's Optimization System Library.

Yet, while the techniques used by Crowder, Johnson, and Padberg work well on a broad collection of 0/1 real-world integer programs, they do not work well on all such problems. The challenge of general integer programming algorithms is to discover a collection of basic techniques which, when taken together, can solve most problems sharing characteristics common to many real-world integer programs.

In this paper we examine a cutting plane technique with applicability to the solution of general integer programs. The cutting planes are based on a methodology suggested by the author in [3]. The proposed *Fenchel cuts* build upon the celebrated dual relationship between separation and optimization and differ from more conventional cuts by focusing directly on the problem of cutting plane generation without reference to an underlying class of facet-inducing cutting planes.

The paper is organized as follows. In Section 2 the theory underlying Fenchel cuts is outlined in sufficient detail to provide a framework for the remaining sections, and the specific method for generating the proposed class of Fenchel cuts is then presented in Section 3. In Section 4 computational results are presented demonstrat-

ing the effectiveness of these cutting planes as a tool for solving real-world integer programming problems. The paper concludes with a discussion of further possible applications of this technique.

2 Fenchel Cuts

The underlying theory of Fenchel cuts is developed in [3]. In this section, we outline the aspects of this theory necessary for the developments presented in this paper.

Consider the arbitrary integer program

$$\begin{array}{lll} & \max & cx \\ (P) & \text{s.t.} & Ax \leq b \\ & & x \text{ integer} \end{array}$$

where $x \in \mathbb{R}^n$. Let $\mathcal{P}_F$ denote some polyhedron containing the feasible integer points for (P) (which we assume is bounded for simplicity of exposition) and let $\hat{x}$ be feasible for the constraints of (P) with the exception of the integrality restriction. Conceptually, $\hat{x}$ can be thought of as a point generated by solving the linear programming relaxation of (P). The cut generation procedure to be described seeks an inequality that is not satisfied by $\hat{x}$ but that contains $\mathcal{P}_F$ and therefore the set of feasible points for (P).

While the cut generation procedure can be described in terms of an arbitrary polyhedron $\mathcal{P}_F$, the applicability of this procedure cannot be fully appreciated without an understanding of how the polyhedron $\mathcal{P}_F$ might arise in practice. As but one

example, the polyhedron $\mathcal{P}_F$ could be the convex hull of feasible integer points for a relaxation of (P) obtained by eliminating some collection of complicating constraints. Polyhedra $\mathcal{P}_F$ arising in this way are found in another popular integer programming technique, namely, Lagrangian relaxation. As in Lagrangian relaxation, the technique described in this section does not require that the polyhedron $\mathcal{P}_F$ be expressed as an explicit collection of linear inequalities, but instead that it is possible to optimize a linear function on $\mathcal{P}_F$. More will be said about a specific class of polyhedra $\mathcal{P}_F$ in the next section.

Letting λ be a vector in $\mathbb{R}^n$ we define $f(\lambda)$ and $v(\lambda)$ as follows.

$$f(\lambda) = \max\{\lambda x : x \in \mathcal{P}_F\}$$

$$v(\lambda) = \lambda\hat{x} - f(\lambda)$$

The following is easily verified directly.

Theorem 1 *There exists a value λ for which $v(\lambda) > 0$ if and only if there exists a hyperplane $\lambda x \leq f(\lambda)$ separating $\hat{x}$ from $\mathcal{P}_F$.*

The practical implication of Theorem 1 is that the question of whether or not there exists a cutting plane separating $\hat{x}$ from $\mathcal{P}_F$ can be answered by investigating whether or not the function $v(\lambda)$ achieves a positive value. For any fixed value of λ the inequality

$$\lambda x \leq f(\lambda)$$

is valid for $\mathcal{P}_F$, and this inequality separates $\mathcal{P}_F$ from $\hat{x}$ if and only if $v(\lambda) > 0$. Due to connections with Fenchel duality such cuts were deemed Fenchel cuts.

The following two theorems, which can be found in [3], make note of important theoretical properties that simplify finding values of λ for which $v(\lambda) > 0$.

Theorem 2 *The function $v(\lambda)$ is piecewise linear and concave. Specifically, $v(\lambda)$ can be expressed as*

$$v(\lambda) = \min\{\lambda\hat{x} - \lambda x^i : x^i \in E(\mathcal{P}_F)\}$$

where $E(\mathcal{P}_F)$ is the set of extreme points of $\mathcal{P}_F$.

Theorem 3 *If $\overline{x} \in \mathcal{P}_F$ satisfies $\overline{\lambda}\overline{x} = f(\overline{\lambda})$ then $(\hat{x} - \overline{x})$ is a subgradient of $v(\lambda)$ at $\overline{\lambda}$.*

The definition of $v(\lambda)$ yields the following observation.

Observation 1 *For any scalar $\omega \geq 0$, $v(\omega\lambda) = \omega v(\lambda)$.*

The immediate implication of this observation is that if $v(\lambda)$ achieves a positive value it achieves a positive value on any full dimensional set containing the origin in its strict interior. In fact, it is not difficult to verify the following observation.

Observation 2 *$v(\lambda)/\|\lambda\|$ is the distance from $\hat{x}$ to the plane $\lambda x = f(\lambda)$ when $\lambda\hat{x} \leq f(\lambda)$ separates $\hat{x}$ and $\mathcal{P}_F$ and the negative of this distance when it does not.*

Thus, solving the maximization problem

$$\begin{aligned} \max \quad & v(\lambda) \\ \text{s.t.} \quad & \|\lambda\| \leq 1 \end{aligned}$$

generates the deepest cut separating a point $\hat{x}$ from $\mathcal{P}_F$. In practice, it is easier to attempt to maximize $v(\lambda)$ on a linear domain. Further, through the appropriate choice of domain it is possible to affect the polyhedral characteristics of the generated cut.

In summary, Fenchel cuts are generated by seeking to maximize the function $v(\lambda)$ on any domain with appropriate characteristics. Any value of λ with $v(\lambda) > 0$ corresponds to a cutting plane, and if the maximum value of $v(\lambda)$ is zero or less then this represents a proof that there exists no cutting plane separating $\hat{x}$ from $\mathcal{P}_F$.

3 Fenchel Cuts from Subproblem Enumeration

At the heart of Fenchel cut generation for a polyhedron $\mathcal{P}_F$ is the need for an oracle which optimizes a linear function on $\mathcal{P}_F$. This can be seen from the fact that the function $v(\lambda)$ is defined in terms of $f(\lambda)$, which is calculated by maximizing λx on $\mathcal{P}_F$. With an optimization oracle it is possible to develop algorithms for maximizing $v(\lambda)$ based on generalized programming or, using the results of Theorem 3, approximate ascent algorithms based on subgradients (see [3]). Assuming an oracle which can parametrically optimize a linear function of $\mathcal{P}_F$ it is possible to apply the simplex algorithm to maximize $v(\lambda)$, and simplex based algorithms tend to be the most efficient in practice (see [4]).

In this section we propose a way to define a subproblem polyhedron $\mathcal{P}_F$ on which a linear function can be parametrically maximized and which provides good cutting

planes. In order to facilitate the discussion, we express the constraints of an arbitrary integer program in the following form.

$$\sum_{j=1}^{p} a_{ij}x_j + \sum_{j=1}^{q} d_{ij}y_j \leq b_i \qquad i = 1, \ldots, m$$

$$0 \leq x \leq 1$$

$$0 \leq y \leq 1$$

$$x, y \text{ integer}$$

Defining the vector $\overline{y}^i$ so that $\overline{y}^i_j = 0$ if $d_{ij} \geq 0$ and $\overline{y}^i_j = 1$ if $d_{ij} < 0$, we consider the polyhedron $\mathcal{P}_F$ defined by the convex hull of feasible integer solutions to the following collection of inequalities.

$$\sum_{j=1}^{p} a_{ij}x_j \leq b_i - \sum_{j=1}^{q} d_{ij}\overline{y}^i_j \qquad i = 1, \ldots, m$$

$$0 \leq x \leq 1$$

$$x \text{ integer}$$

Formally, the polyhedron $\mathcal{P}_F$ of interest resides in the space $\mathbb{R}^{p+q}$ corresponding to the x and y variables. However, it is conceptually simpler to think of $\mathcal{P}_F$ as residing in $\mathbb{R}^p$ and recognizing that when speaking of valid inequalities for $\mathcal{P}_F$ it is intended to mean that such inequalities will be extended to $\mathbb{R}^{p+q}$ by choosing coefficients of 0 for the subspace corresponding to the y variables. By the very choice of the $\overline{y}^i$, it is easy to see that any valid inequality for $\mathcal{P}_F$ must be valid for the original integer program.

If the polyhedron $\mathcal{P}_F$ has some special structure then it may be possible to apply any variety of techniques to generate cutting planes for that structure, while if $\mathcal{P}_F$ has no special structure then the problem of cutting plane generation remains very

difficult in general. However, if the number of feasible integer points contained in $\mathcal{P}_F$ is small enough, solving the optimization problem necessary to generate a Fenchel cut can be accomplished by enumeration; determining λx^k for all integer $x^k \in \mathcal{P}_F$ and choosing the largest value of λx^k. It is easy to see that parametric optimization is no more difficult so that efficient simplex procedures can be used to generate Fenchel cuts. When a Fenchel cut is generated by solving the necessary optimization problem enumeratively we will refer to it as an *enumeration cut.*

In the following section we investigate the potential of this approach by applying it to a collection of integer programs taken from real-world applications.

4 Applying Enumeration Cuts

The cutting planes described in the previous section were tested by applying them to the collection of 0/1 integer programs studied by Crowder, Johnson, and Padberg in [6]. A summary of these problems is shown in Table 1. Foremost among the reasons for choosing these problems is that they come from real-world applications. Such problems display characteristics that are not generally exhibited by randomly generated problems and provide a much more fertile ground for developing practically useful computational techniques. An additional consideration is that the problems are rapidly becoming a standard test set, having been studied previously in [1], [3], [4], [6], and [9]. The problems are available electronically through the MIPLIB library as described in [2] (additional information can also be obtained by sending email

Name	Variables	Constraints	Nonzeros	v_{LP}	v_{IP}
P0033	33	15	98	2520.6	3089.0
P0040	40	23	110	61796.5	62027.0
P0201	201	133	1923	6875.0	7615.0
P0282	282	241	1966	176867.5	258411.0
P0291	291	252	2031	1705.1	5223.8
P0548	548	176	1711	315.3	8691.0
P2756	2756	755	8937	2688.7	3124.0

Table 1: Summary of Problems

to *softlib@rice.edu* with the message *send catalog*). The goal of the computational tests was to determine whether enumeration cuts could be effective in refining the linear programming approximation to an integer program. For this reason, no effort was made to develop a branch and bound shell in which to embed the cutting plane routines.

An outline of the algorithm used to test enumeration cuts is given in Figure 1. While general preprocessing routines were not used (see [9]) two very basic variable fixing routines were included since they had an important impact on the ability to apply enumeration cuts effectively. The coefficient magnitude check consists of determining whether the coefficient of a variable x_j in a constraint $\sum a_i x_i \leq b$ with all $a_i > 0$ satisfies $a_j > b$, and if so setting x_j to 0. While seemingly a very weak check, it can have a cascading effect whenever a variable is fixed using reduced cost variable fixing. Reduced cost variable fixing (see [11, p. 389]) is a technique that fixes an integer variable at its present value in the linear programming optimal solution whenever the magnitude of its reduced cost exceeds the gap between the present linear

Figure 1: **Algorithm KE Cut**

1. Solve the linear programming relaxation of the integer program to obtain an optimal solution $\hat{x}$. If $\hat{x}$ is integral, stop.
2. Fix variables using reduced costs and a coefficient magnitude check.
3. Determine variable sets that may yield valuable enumeration cuts and generate a feasible point list for these sets.
4. For all knapsack constraints and enumeration sets of interest at this iteration, attempt to generate a cutting plane. If no cutting plane can be generated, stop. Otherwise, append the generated cutting planes to the problem formulation and return to step 1.

programming value and the value of any known integer solution. Since the algorithm makes no attempt to generate feasible solutions to the integer program and since it was not at all clear what "arbitrary" gap value should be chosen, the optimal value of the integer program was used for calculating such a gap. While there is a slight performance degradation when a larger gap value is used, the numerical results were relatively insensitive to the specific gap value.

Knapsack cutting planes, which have proven phenomenally successful in the solution of many 0/1 integer programs, were generated using the same Fenchel cutting plane ideas as enumeration cuts, but with the underlying polyhedron $\mathcal{P}_F$ defined by the feasible points for the knapsack polyhedra associated with the individual constraints of the problem. Details regarding the efficient generation of Fenchel cutting planes for knapsack polyhedra can be found in [4]. These cuts were used in addition to enumeration cuts because they tended to interact well with enumeration cuts, providing better results than if just one type of cutting plane had been used.

4.1 Enumeration Cut Implementation

A fundamental question regarding the effective use of enumeration cuts is how the variable set defining the polyhedron $\mathcal{P}_F$ is chosen. Since every variable that is not used in defining $\mathcal{P}_F$ must be set at a value that most weakens each constraint in which it is involved, it is not difficult to choose variable sets that yield large numbers of feasible points and very weak cutting planes. Thus, care must be taken in choosing potentially useful variable sets. In addition, the overwhelmingly important factor determining the speed with which enumeration cuts can be generated is the number of feasible points contained in the polyhedron $\mathcal{P}_F$. If $\mathcal{P}_F$ is defined on a set of k variables, then $\mathcal{P}_F$ could contain as many as 2^k feasible 0/1 vectors, which for practical purposes becomes too large in the neighborhood of $k = 15$. In practice, not all 2^k points will be feasible and polyhedra $\mathcal{P}_F$ based on larger collections of variables can be used. Nonetheless, effective application of enumeration cuts requires the use of reasonably small variable sets.

Variable sets were chosen to correspond to the variables in individual constraints in each problem. In this way, the number of feasible points was limited by the number of feasible points for the constraint itself and could potentially be reduced by any overlapping constraints — constraints whose variable sets have a nonempty intersection with the chosen constraint. The existence of overlapping constraints was particularly important in the case of special ordered set constraints; that is,

constraints that can be written in the form

$$\sum x_i \leq 1$$

with properly complemented variables. In fact, special ordered set constraints proved to be so instrumental that in instances where the variable set defined by the constraint was small enough it was extended to include the variables in overlapping special ordered set constraints. If this extended constraint was too large, the original constraint set was used.

The idea of extending the variable set to include special ordered set variables was motivated by the use of surrogate constraints in the original work of Crowder, Johnson, and Padberg. Surrogate constraints are valid inequalities for an integer program constructed by taking a positive combination of other valid inequalities for that problem. Crowder, Johnson, and Padberg used this technique to construct surrogate constraints consisting of special ordered set constraints combined with one other valid inequality. One of the difficulties with surrogate constraint generation is that there is considerable freedom in the choice of multipliers used to construct the surrogate constraint, and different choices of multipliers can lead to radically different constraints. On the other hand, it is not difficult to argue that the polyhedron defined by the points used in generating an enumeration cut must always be contained in the knapsack polyhedron defined by a surrogate constraint on the same set of variables. Thus, enumeration cuts will always be at least as strong the knapsack cuts from any surrogate constraint, and just as importantly, the use of enumeration cuts takes

the guesswork out of choosing a good set of multipliers for constructing a surrogate constraint.

When enumerating the set of feasible points corresponding to a given variable set, an upper bound was placed on the number of feasible points that would be allowed. If more than 4000 feasible points were generated, the variable set was abandoned under the assumption that too much time would be required to generate enumeration cuts based on a set of this size. In most instances enumeration cuts were generated on much smaller sets, usually containing under 500 points and often containing fewer than 100. In the case of problem P0201 the bound of 4000 was relaxed to 6000. It was, in fact, quite a surprise that such small enumeration sets could be used as effectively as is demonstrated by the computational results.

Since cutting planes generated as valid inequalities for knapsack polyhedra are by nature defined on some subset of the variables in the original knapsack constraint, it follows that enumeration cuts based on the constraint set of the original knapsack polyhedron will always be at least as strong as those based on the cutting plane. However, it was found that in some cases while an original knapsack polyhedron might have too many variables to define an enumeration cut, the set of variables associated with a cutting plane for that polyhedron might be sufficiently small for this purpose. In fact, not only were such sets of appropriate size but they tended to define variable sets which yielded useful enumeration cuts. This phenomenon was particularly evident in the solution of problem P0282, and was discovered quite fortuitously.

4.2 Computational Results

Computational results for Algorithm KE Cut are shown in Tables 2 and 3. For comparison, Tables 4 and 5 present computational results for the same algorithm but with no enumeration cuts; that is, with knapsack cuts and rudimentary variable fixing only. The column labeled *Cuts* represents the total number of cutting planes appended in the course of the algorithm. The columns labeled $v_{LP}^{1.0}$, $v_{LP}^{2.0}$, and $v_{LP}^{3.0}$ give the value of the linear programming relaxation after 1, 2, and 3 minutes, respectively, and the column v_{LP}^{T} gives the value of the linear programming relaxation after T minutes where T is given in the table. The values $\Delta Gap^{1.0}$, $\Delta Gap^{2.0}$, and $\Delta Gap^{3.0}$ represent the percentage by which the gap between v_{LP} and v_{IP} was reduced in 1, 2, and 3 minutes, respectively, with ΔGap^{T} representing the percentage by which this gap was closed in T minutes. All computational tests were performed on a SUN SPARCserver 490.

Since Fenchel cuts are guaranteed to generate a cutting plane if one exists, and since the algorithm was only terminated when no cutting planes could be found, the values v_{LP}^{T} in Tables 4 and 5 represent the provably best gap reduction that can be achieved using only knapsack cutting planes and rudimentary variable fixing. This is true of all problems except P2756 which contained two constraints that were sufficiently large so that an exact separation procedure could not be employed. The difference between the values in Tables 4 and 5 can therefore be fully attributed to the use of enumeration cuts.

Of particular interest is the fact that four of the seven problems were solved to optimality using only cutting planes. Although problem P0040 required the aid of branch and bound in the original study of Crowder, Johnson, and Padberg, it has since been recognized as an extremely easy problem to solve using only a small number of knapsack cutting planes and all subsequent studies have not required branch and bound. In [9], problem P0548 was solved using a combination of knapsack cutting planes and extensive preprocessing techniques. This problem has also been solved using preprocessing techniques by members of the OSL development group at the IBM T. J. Watson Research Center. Given that this problem was at one time considered unsolvable, its solution in nothing more than a few minutes of CPU time without the aid of branch and bound is a remarkable computational achievement for all of these computational studies. However, while problems P0040 and P0548 have been solved without branch and bound in other studies, to the author's knowledge this is the first study in which the problems P0033 and P0282 have been solved without the use of branch and bound. While relatively small, problem P0033 has long been recognized as a challenging problem, and just achieving a strong linear programming relaxation can be a difficult task. Of great surprise to the author was the fact that problem P0282 could be solved to optimality with knapsack and enumeration cuts.

5 Conclusions

In this paper we have examined a cutting plane technique with applicability to the solution of general integer programs. The computational value of this technique was demonstrated by applying it to a collection of problems arising from real-world applications. As a result, four of the seven test problems were solved to optimality without the aid of branch and bound, and six of the seven problems had the gap between the value of the integer program and its linear programming relaxation closed by over 98%. All of the results were achieved in under ten minutes of CPU time on a SUN workstation.

While the results demonstrate that enumeration cuts are valuable computational tools, the cuts themselves represent a very general technique, and the question of instances in which they are effective in solving integer programs deserves further consideration. In this paper it was shown that variable sets defined by problem constraints, both from the original problem and from cutting planes appended to the problem, were computationally useful. However, there are many alternative methods for choosing the variable sets underlying enumeration cuts. For example, one method that was not explored was variable sets defined by pairs of constraints. With these and other techniques, it may yet be possible to do what would have been considered impossible only a decade ago: solve all seven of the test problems without the aid of branch and bound.

Name	v_{LP}	v_{IP}	$v_{LP}^{1.0}$	$v_{LP}^{2.0}$	$v_{LP}^{3.0}$	v_{LP}^{T}	$T\dagger$	Cuts
P0033	2520.60	3089.00	-	-	-	3089.00	.13	57
P0040	61796.50	62027.00	-	-	-	62027.00	.01	2
P0201	6875.00	7615.00	7106.08	7185.00	7185.00	7185.00	3.10	36
P0282	176867.50	258411.00	256234.08	256851.19	257111.88	258411.00	9.67	566
P0291	1705.10	5223.75	-	-	-	5204.17	.75	133
P0548	315.30	8691.00	3463.70	6008.12	8053.00	8691.00	3.73	202
P2756	2688.70	3124.00	2701.75	2912.58	3056.12	3115.60	7.18	495

†minutes on a SUN SPARCserver 490

Table 2: Cut Summary With Enumeration Cuts

Name	v_{LP}	v_{IP}	$\Delta Gap^{1.0}$	$\Delta Gap^{2.0}$	$\Delta Gap^{3.0}$	ΔGap^{T}	$T\dagger$	Cuts
P0033	2520.60	3089.00	-	-	-	100.00%	.13	57
P0040	61796.50	62027.00	-	-	-	100.00%	.01	2
P0201	6875.00	7615.00	31.23%	41.89%	41.89%	41.89%	3.10	36
P0282	176867.50	258411.00	97.33%	98.09%	98.41%	100.00%	9.67	566
P0291	1705.10	5223.75	-	-	-	99.44%	.75	133
P0548	315.30	8691.00	37.59%	67.97%	92.38%	100.00%	3.73	202
P2756	2688.70	3124.00	3.00%	51.43%	84.41%	98.07%	7.18	495

†minutes on a SUN SPARCserver 490

Table 3: Cut Summary With Enumeration Cuts

Name	v_{LP}	v_{IP}	$v_{LP}^{1.0}$	$v_{LP}^{2.0}$	$v_{LP}^{3.0}$	v_{LP}^{T}	$T\dagger$	Cuts
P0033	2520.60	3089.00	-	-	-	3017.50	.15	58
P0040	61796.50	62027.00	-	-	-	62027.00	.01	2
P0201	6875.00	7615.00	7125.00	-	-	7125.00	1.07	50
P0282	176867.50	258411.00	256234.08	256851.19	257261.97	257261.97	3.28	413
P0291	1705.10	5223.75	-	-		5204.17	.74	133
P0548	315.30	8691.00	7248.51	7379.28	7379.28	7379.28	3.41	426
P2756	2688.70	3124.00	2701.75	2912.58	3056.12	3063.75	3.93	417

†minutes on a SUN SPARCserver 490

Table 4: Cut Summary Without Enumeration Cuts

Name	v_{LP}	v_{IP}	$\Delta Gap^{1.0}$	$\Delta Gap^{2.0}$	$\Delta Gap^{3.0}$	ΔGap^{T}	$T\dagger$	Cuts
P0033	2520.60	3089.00	-	-	-	87.42%	.15	58
P0040	61796.50	62027.00	-	-	-	100.00%	.01	2
P0201	6875.00	7615.00	33.78%	-	-	33.78%	1.07	50
P0282	176867.50	258411.00	97.33%	98.09%	98.59%	98.59%	3.28	413
P0291	1705.10	5223.75	-	-	-	99.44%	.74	133
P0548	315.30	8691.00	82.78%	84.34%	84.34%	84.34%	3.41	426
P2756	2688.70	3124.00	3.00%	51.43%	84.41%	86.16%	3.93	417

†minutes on a SUN SPARCserver 490

Table 5: Cut Summary Without Enumeration Cuts

References

[1] Balas, E., S. Ceria, and G. Cornuéjols. 1991. A Lift-and-Project Cutting Plane Algorithm for Mixed 0-1 Programs. Management Science Research Report 576, Graduate School of Industrial Administration, Carnegie Mellon University.

[2] Bixby, R. E., E. A. Boyd, and R. Indovina. March 1992. MIPLIB: A Test Set of Mixed Integer Programming Problems. *SIAM News* **25:2**, 16.

[3] Boyd, E. A. 1990. Fenchel Cutting Planes for Integer Programs. To appear in *Operations Research.*

[4] Boyd, E. A. 1990. Solving Integer Programs with Fenchel Cutting Planes for Knapsack Polyhedra. Technical Report TR90-20, Department of Mathematical Sciences, Rice University.

[5] Boyd, E. A. 1991. On the Convergence of Fenchel Cutting Planes in Mixed-Integer Programming. Technical Report TR91-39, Department of Mathematical Sciences, Rice University.

[6] Crowder, H., E. L. Johnson, and M. W. Padberg. 1983. Solving Large-scale Zero-one Linear Programming Problems. *Operations Research* **31**, 803-834.

[7] Gomory, R. E. 1958. Outline of an Algorithm for Integer Solutions to Linear Programs. *Bulletin of the American Mathematical Society* **64**, 275-278.

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

[9] Hoffman, K. L., and M. W. Padberg. 1991. Improving the LP-Representation of Zero-One Linear Programs for Branch-and-Cut. *ORSA Journal on Computing* **3**, 121-134.

[10] Lemaréchal, C. 1989. Nondifferentiable Optimization. In *Optimization*, G. L. Nemhauser *et. al.* eds., *Handbooks in Operations Research and Management Science* **1**, 529-572. North Holland, New York.

[11] Nemhauser, G. L. and L. A. Wolsey. 1988. *Integer and Combinatorial Optimization*. Wiley and Sons, New York.

[12] Shapiro, J. F. 1971. Generalized Lagrange Multipliers in Integer Programming. *Operations Research* **19**, 68-76.

[13] Shapiro, J. F. 1979. *Mathematical Programming: Structures and Algorithms*. Wiley and Sons, New York.

[14] Wolfe, P. 1976. Finding the Nearest Point in a Polytope. *Mathematical Programming* **11**, 128-149.