

A Family of Inequalities for the Generalized Assignment Polytope

I. R. de Farias, Jr.

Department of Industrial Engineering

State University of New York at Buffalo, 403 Bell Hall, Buffalo, NY 14260-2050

G. L. Nemhauser*

School of Industrial and Systems Engineering

Georgia Institute of Technology, Atlanta, GA 30332-0205

June 26, 2000

Abstract

We present a family of inequalities that are valid for the generalized assignment polytope. Although the inequalities are not facet-defining in general, they define facets of a polytope of a relaxation. We report computational results on the use of the inequalities in a branch-and-cut scheme that demonstrate their effectiveness.

Keywords: integer programming, generalized assignment, branch-and-cut

1 Introduction

Given a set of m jobs and a set of n processors, the *Generalized Assignment Problem* (GAP) consists of finding the cheapest assignment of each job to a single processor that respects the capacity constraints of the processors. GAP has many applications and several algorithms have been proposed for solving it. Cattrysse and van Wassenhove [1] review solution approaches for solving GAP, and, more recently, Savelsbergh [10] proposed a branch-and-price algorithm.

Let $M = \{1, \dots, m\}$ and $N = \{1, \dots, n\}$. Given $i \in M$ and $j \in N$, we denote by $b_j > 0$ the capacity of processor j , $a_{ij} \geq 0$ the capacity of processor j required by job i , and c_{ij} the cost of assigning job i to processor j . We assume that $b_j, a_{ij} \in \mathbb{Z}$, that $m, n \geq 2$, and that GAP is feasible. GAP can be formulated as

*Partially supported by NSF grant DMI-9700285 to the Georgia Institute of Technology.

$$\begin{aligned} \min \quad & \sum_{i \in M} \sum_{j \in N} c_{ij} x_{ij} \\ & \sum_{i \in M} a_{ij} x_{ij} \leq b_j, \quad j \in N, & (1) \\ & \sum_{j \in N} x_{ij} = 1, \quad i \in M, & (2) \\ & x_{ij} \in \{0, 1\}, \quad i \in M, j \in N. & (3) \end{aligned}$$

By modifying the objective function coefficients appropriately, (2) can be replaced by

$$\sum_{j \in N} x_{ij} \leq 1, i \in M, \quad (4)$$

see [7].

Let $S = \{x \in \{0, 1\}^{mn} : x \text{ satisfies (1) } \forall j \in N \text{ and (4) } \forall i \in M\}$ and $PS = \text{conv}(S)$. Because PS is a full-dimensional polytope, we use the inequality formulation of GAP with (4) replacing (2). We call PS the GAP polytope. The LP relaxation of GAP is obtained by replacing (3) with $x_{ij} \in [0, 1] \forall ij \in M \times N$. The set of feasible solutions of the LP relaxation of GAP is denoted by LPS .

In [5, 6], Gottlieb and Rao presented several inequalities that are valid for the GAP polytope. In particular, they proved that the inequalities defining facets of the 0-1 knapsack polytopes that correspond to the individual capacity constraints (1) are facet-defining for the GAP polytope as well. These inequalities - lifted cover inequalities (LCIs) - are automatically generated in several commercial and academic MIP solvers that implement branch-and-cut schemes.

In this paper we introduce a new family of inequalities that are valid for the GAP polytope. The inequalities, in general, are not facet-defining. However, they define facets of the convex hull of the set of the feasible solutions of a relaxation of GAP obtained by allowing the jobs to be partially executed (or not executed at all), while keeping the constraint that a job is assigned to at most one processor. The inequalities have a simple structure, and it is possible to separate them efficiently with a heuristic that is fast in practice. We demonstrate through computational experimentation that the inequalities are effective when incorporated into a branch-and-cut scheme to solve GAP.

The paper is organized as follows. In section 2 we present a family of inequalities that define facets of the convex hull of the set of feasible solutions of the relaxation. In section 3 we report computational results on the use of the inequalities presented in section 2 in a branch-and-cut scheme to solve GAP.

2 A Family of Valid Inequalities

In this section we relax the constraint that the jobs must be fully executed, or even executed at all. We keep the capacity constraint and the constraint that a job is assigned to at most one processor. A formulation for the relaxed problem can be obtained by substituting $x_{ij} \in [0, 1] \forall ij \in M \times N$ for (3), and by adding the following multiple-choice constraint

$$\text{at most one of } x_{i1}, \dots, x_{in} \text{ is positive } \forall i \in M. \quad (5)$$

We call this relaxation the *multiple-choice GAP* (MGAP).

Theorem 1 *MGAP is NP-hard.*

Proof We reduce the partition problem [4] to MGAP.

PARTITION

INSTANCE: Two sets of positive integers $M = \{1, \dots, m\}$ and $S = \{a_1, \dots, a_m\}$.

QUESTION: Does M have a subset M' such that $\sum_{i \in M'} a_i = \sum_{i \in M - M'} a_i$?

Let $n = 2$, $a_{i1} = a_{i2} = a_i \forall i \in M$, $b_1 = b_2 = \frac{1}{2} \sum_{i \in M} a_i$, and consider the MGAP

$$\begin{aligned} \max \quad & \sum_{i \in M} a_i(x_{i1} + x_{i2}) + \sum_{i \in M} (x_{i1} + x_{i2}) \\ & \sum_{i \in M} a_i x_{ij} \leq \frac{1}{2} \sum_{i \in M} a_i, \quad j \in \{1, 2\}, \\ & x_{i1} + x_{i2} \leq 1, \quad i \in M, \\ & x_{ij} \in [0, 1], \quad i \in M, j \in \{1, 2\} \\ & \text{at most one of } x_{i1} \text{ and } x_{i2} \text{ is positive, } i \in M. \end{aligned}$$

Note that the optimal value of the above MGAP is at most $\sum_{i \in M} a_i + m$. Suppose that the optimal value of MGAP is $\sum_{i \in M} a_i + m$. Then, if \tilde{x} is an optimal solution, $\tilde{x}_{i1} + \tilde{x}_{i2} = 1 \forall i \in M$, and thus $\tilde{x}_{ij} \in \{0, 1\} \forall i \in M, j \in \{1, 2\}$. Also, $\sum_{i \in M} a_i \tilde{x}_{i1} = \sum_{i \in M} a_i \tilde{x}_{i2} = \frac{1}{2} \sum_{i \in M} a_i$. Therefore, by taking $M' = \{i \in M : \tilde{x}_{i1} = 1\}$, we have that the solution to PARTITION is *yes*.

Conversely, suppose that the solution to PARTITION is *yes*. This means that $\sum_{i \in M'} a_i = \sum_{i \in M - M'} a_i = \frac{1}{2} \sum_{i \in M} a_i$. Let \hat{x} be given by

$$\hat{x}_{i1} = \begin{cases} 1 & \text{if } i \in M' \\ 0 & \text{if } i \in M - M', \end{cases}$$

and

$$\hat{x}_{i2} = \begin{cases} 1 & \text{if } i \in M - M' \\ 0 & \text{if } i \in M'. \end{cases}$$

It follows that \hat{x} is a feasible solution to MGAP with objective function value $\sum_{i \in M} a_i + m$, and therefore it is an optimal solution. Thus, the solution to PARTITION is *yes* iff the optimal value to MGAP is $\sum_{i \in M} a_i + m$. \square

We denote by PS^1 the convex hull of the set of feasible solutions to MGAP. Next, we present a family of inequalities that are valid and facet-defining for PS^1 . Because $PS \subseteq PS^1$, the inequalities are also valid for PS . The inequalities are a strengthening of the inequalities

of a family that define facets of the convex hull of the set of feasible solutions of an SOS II relaxation of GAP [3]. They can be obtained by fixing some of the variables at 0, and then lifting (1); see [2, 12] for a general definition and result about lifting. To motivate the family of inequalities, we present an example.

Example 1 Let $m = 3$, $n = 3$, $a_{11} = a_{12} = a_{13} = 2$, $a_{21} = a_{22} = a_{23} = 4$, $a_{31} = a_{32} = a_{33} = 5$, and $b_1 = b_2 = b_3 = 5$. Fix all variables at 0, except for x_{11} and x_{21} . The inequality

$$2x_{11} + 4x_{21} \leq 5 \tag{6}$$

is valid and facet-defining for the resulting polytope $PS^1 \cap \{x \in \mathfrak{R}^9 : x_{12} = x_{13} = x_{22} = x_{23} = x_{31} = x_{32} = x_{33} = 0\}$. When $x_{12} > 0$, $x_{11} = 0$. Because $4x_{21} \leq 4$, and $x_{12} \leq 1$, the lifting coefficient of x_{12} is 1. Likewise, the lifting coefficient of x_{13} is 1. Thus, by lifting (6) with respect to x_{12} and x_{13} , we obtain

$$2x_{11} + 4x_{21} + x_{12} + x_{13} \leq 5. \tag{7}$$

In the same way, by lifting (7) with respect to x_{22} and x_{23} , we obtain

$$2x_{11} + 4x_{21} + x_{12} + x_{13} + 3x_{22} + 3x_{23} \leq 5. \tag{8}$$

We now lift (8) with respect to x_{31} . The lifting coefficient of x_{31} , α_{31} , is given by

$$2x_{11} + 4x_{21} + x_{12} + x_{13} + 3x_{22} + 3x_{23} + \alpha_{31}x_{31} \leq 5. \tag{9}$$

By substituting the point $x_{11} = 1$, $x_{31} = \frac{3}{5}$, $x_{22} = 1$, $x_{12} = x_{13} = x_{21} = x_{23} = x_{32} = x_{33} = 0$, which is feasible for $PS^1 \cap \{x \in \mathfrak{R}^9 : x_{32} = x_{33} = 0\}$, in (9), we obtain $\alpha_{31} \leq 0$. Since (8) is valid for $PS^1 \cap \{x \in \mathfrak{R}^9 : x_{32} = x_{33} = 0\}$, $\alpha_{31} = 0$. Likewise, the lifting coefficients of x_{32} and x_{33} are 0, and therefore (8) is facet-defining for PS . \square

Theorem 2 Let $I \subseteq M$ and $t \in N$. Suppose that $\sum_{i \in I} a_{it} > b_t$ and $\sum_{i \in I - \{r\}} a_{it} < b_t$ for some $r \in I$. Then,

$$\sum_{i \in I} a_{it}x_{it} + \sum_{i \in I} (\max\{0, b_t - \sum_{i' \in I - \{i\}} a_{i't}\}) \sum_{j \in N - \{t\}} x_{ij} \leq b_t \tag{10}$$

defines a facet of PS^1 .

Proof We first prove that (10) is valid. Inequality (10) is clearly valid when $x_{ij} = 0 \forall i \in I$ such that $b_t - \sum_{i' \in I - \{i\}} a_{i't} > 0$ and $j \in N - \{t\}$. So, suppose that $x_{uv} > 0$ for some $u \in I$ with $b_t - \sum_{i' \in I - \{u\}} a_{i't} > 0$ and $v \in N - \{t\}$. It follows that

$$\sum_{i \in I} a_{it}x_{it} + \sum_{i \in I} (\max\{0, b_t - \sum_{i' \in I - \{i\}} a_{i't}\}) \sum_{j \in N - \{t\}} x_{ij} =$$

$$\sum_{i \in I - \{u\}} a_{it} x_{it} + \sum_{i \in I - \{u\}} (\max\{0, b_t - \sum_{i' \in I - \{i\}} a_{i't}\}) \sum_{j \in N - \{t\}} x_{ij} + (b_t - \sum_{i' \in I - \{u\}} a_{i't}) x_{uv} \leq$$

$$\sum_{i \in I - \{u\}} a_{it} x_{it} + \sum_{i \in I - \{u\}} a_{it} \sum_{j \in N - \{t\}} x_{ij} + (b_t - \sum_{i' \in I - \{u\}} a_{i't}) x_{uv} \leq \sum_{i \in I - \{u\}} a_{it} + (b_t - \sum_{i' \in I - \{u\}} a_{i't}) = b_t,$$

where the first equality is implied by (5), the first inequality is implied by $a_{it} > b_t - \sum_{i' \in I - \{i\}} a_{i't}$ and $a_{it} \geq 0 \forall i \in I$, and the second inequality is implied by (4).

We now show that (10) is facet-defining. For every $uv \in M \times N$, let $e^{uv} \in \{0, 1\}^{mn}$ be given by

$$e_{ij}^{uv} = \begin{cases} 1 & \text{if } ij = uv \\ 0 & \text{otherwise.} \end{cases}$$

Because $\sum_{i \in I} a_{it} > b_t$, PS^1 has $|I|$ linearly independent points with $x_{uv} = 0 \forall uv \in (M \times N) - (I \times \{t\})$.

Let $u \in I$. Suppose that $\sum_{i \in I - \{u\}} a_{it} < b_t$. Then, $\sum_{i \in I - \{u\}} e^{it} + e^{uv} \in PS^1$ and satisfies (10) at equality $\forall v \in N - \{t\}$. Now, suppose that $\sum_{i \in I - \{u\}} a_{it} \geq b_t$. Then, there exists $\tilde{x} \in PS^1$ such that $\tilde{x}_{ij} = 0 \forall ij \in (M \times N) - (I \times \{t\})$, $\tilde{x}_{ut} = 0$, and \tilde{x} satisfies (10) at equality. But then, $\tilde{x} + e^{uv} \in PS^1$ and satisfies (10) at equality $\forall v \in N - \{t\}$.

Let $u \in M - I$. Because $\sum_{i \in I - \{r\}} a_{it} < b_t$, for $\epsilon > 0$ small enough, $\sum_{i \in I - \{r\}} a_{it} + \epsilon a_{ut} \leq b_t$. Let $v \in N - \{t\}$. Then, $\sum_{i \in I - \{r\}} e^{it} + e^{rv} + \epsilon e^{ut} \in PS^1$ and satisfies (10) at equality. Finally, let $\hat{x} \in PS^1$ be such that $\hat{x}_{ij} = 0 \forall ij \in (M \times N) - (I \times \{t\})$, and \hat{x} satisfies (10) at equality. Then, $\hat{x} + e^{uv} \in PS^1$ and satisfies (10) at equality. Thus, PS^1 has mn linearly independent points that satisfy (10) at equality. \square

3 Computational Experience

We tested the effectiveness of (10) when used as cuts in a branch-and-cut scheme. We used an IBM RS6000/590 to run our tests and MINTO 3.0 [8, 11] as branch-and-cut solver, with CPLEX 6.0 as LP solver. We performed preliminary tests to compare the following alternatives: use only (10); use only LCIs; use (10) and LCIs. The preliminary computation proved the first option to be significantly less practical than the other two. We then compared MINTO with LCIs only against MINTO with LCIs and (10). For each instance we limited the size of the branching tree to 50,000 nodes, and the computational time to 36,000 seconds.

We used the four test data generators A, B, C, and D suggested in [7] to obtain test instances. The A and B data are known to be the easiest, while the D are the hardest ones. For each data type we performed tests on several instance sizes, and for each instance size we performed 5 tests with different instances. The instances are denoted as $X.n.m$, where X denotes the data generator. For example, C.3.30 denotes instances of the C data type with $n = 3$ and $m = 30$.

3.1 Branch-and-Cut Options

We performed preliminary tests with MINTO's pre-processing, node selection, and branching strategies. The alternatives that worked best in the preliminary tests were:

- perform pre-processing and limited probing
- use best bound for node selection
- use a pseudo-cost variable rule for variable branching selection.

We then used these options in our tests. We performed preliminary tests with different cut generation strategies, e.g. introduce (10) at all nodes, at the root node only, etc. As expected, the preliminary tests indicated that the more (10) is used, the smaller the enumeration tree becomes. However, the price paid in computational time grows substantially when (10) is used intensively. The preliminary tests indicated that using (10) in only part of the enumeration tree yields a good balance between tree size reduction and increased computational time at each node. Of all the alternatives we tested, the one that appeared to be the most efficient was:

- use (10) in the first 1,000 nodes
- use (10) between nodes 10,000 and 11,000
- use (10) in all nodes from 20,000 on.

We then used this option in our computations. We tested different options with respect to row management alternatives, i.e. perform no row management, delete constraints with positive slacks every 10 iterations, etc. The alternative that seemed to be best was to delete all LCIs and (10) with positive slacks every 100 nodes, and that was the alternative we adopted in our computations.

3.2 Separation Heuristic

It is easy to show that the separation problem for (10) is NP-hard when we consider any point $x \in \mathfrak{R}^{mn} - PS^1$. This can be done by reducing the subset sum problem to it [4]. On the other hand, it appears that it is more difficult to show that the restricted problem of separating points from $LPS - PS^1$ is in fact NP-hard, although we suspect that it is. Thus, we used a heuristic to solve the separation problem.

The heuristic is based on the following observation. Let \tilde{x} be a point that does not satisfy (5). Let $t \in N$,

$$I_t = \{i \in M : \tilde{x}_{it} > 0\},$$

and suppose that

$$\sum_{i \in I_t} a_{it} \tilde{x}_{it} = b_t. \quad (11)$$

Let $u \in I_t$ be such that

$$b_t - \sum_{i \in I_t - u} a_{it} > 0, \tilde{x}_{ut} > 0, \text{ and } \tilde{x}_{uv} > 0 \text{ for some } v \in N - \{t\}. \quad (12)$$

Then \tilde{x} is cut off by (10) with $I = I_t$.

Suppose that the current node is among those for which (10) is used as cuts, and that the optimal solution \tilde{x} of the current LP relaxation does not satisfy (5). The separation heuristic is as follows. We perform n iterations, from $t = 1$ through $t = n$. At each iteration we first check whether (11) is satisfied. If (11) is satisfied, then we check whether (10) with $I = I_t$ is not satisfied by \tilde{x} (this is equivalent to Condition (12) being satisfied for some $u \in I_t$), in which case the inequality cuts off \tilde{x} , and we add it to the LP relaxation.

If at the end of the separation heuristic we have found at least one inequality that cuts off \tilde{x} , then we solve the new LP relaxation. If the new optimal solution satisfies (5), or the new optimal value is not greater than the objective function value of the best feasible solution found so far, we fathom the current node. Otherwise, we apply the separation heuristic to the optimal solution of the new LP relaxation. On the other hand, if at the end of the separation heuristic we do not find any inequality that cuts off \tilde{x} , we branch.

We illustrate the separation heuristic with the following example.

Example 2 Consider the instance of Example 1. Let \tilde{x} be given by $\tilde{x}_{11} = \tilde{x}_{21} = \tilde{x}_{31} = \tilde{x}_{12} = \tilde{x}_{33} = 0$, $\tilde{x}_{22} = \frac{1}{4}$, $\tilde{x}_{32} = \frac{4}{5}$, $\tilde{x}_{13} = 1$, and $\tilde{x}_{23} = \frac{3}{4}$. For $t = 1$, $2\tilde{x}_{11} + 4\tilde{x}_{21} + 5\tilde{x}_{31} = 0 < 5$.

For $t = 2$, $2\tilde{x}_{12} + 4\tilde{x}_{22} + 5\tilde{x}_{32} = 5$. $I_2 = \{2, 3\}$, and the resulting inequality (10) is $4x_{22} + 5x_{32} + x_{31} + x_{33} \leq 5$. However, this inequality does not cut off \tilde{x} .

Finally, for $t = 3$, $2\tilde{x}_{13} + 4\tilde{x}_{23} + 5\tilde{x}_{33} = 5$. $I_3 = \{1, 2\}$, and the resulting inequality (10) is $2x_{13} + 4x_{23} + x_{11} + x_{12} + 3x_{21} + 3x_{22} \leq 5$. This inequality cuts off \tilde{x} , and we add it to the LP relaxation. \square

3.3 Computational Results

We tested instances A and B of various sizes, including very large instances with over 10,000 variables. Most instances, however were solved at the root node, the majority of time spent with pre-processing and probing. In most cases, MINTO generated at most 3 lifted cover inequalities and inequalities (10). The C and D instances were more interesting. We were able to solve C instances with up to 9,000 variables. MINTO with (10) added was able to solve to proven optimality all C instances tested. MINTO without (10) added was unable to solve to proven optimality 8 C instances among the ones with 4,000 variables or more. The D instances, as expected, were considerably harder. We were able to solve D instances with up to 1,575 variables. MINTO with (10) added was able to solve to proven optimality

all D instances. MINTO without (10) added was unable to solve to proven optimality 3 D instances.

Table 1 gives, for each instance type and size, the average number of nodes and computational time, over the 5 instances tested, of MINTO with LCIs only and with LCIs and (10). Our tests indicate that using (10) can be particularly useful when solving large C instances and the D instances. Table 1 also gives the average number of cuts generated when MINTO generates LCIs only and when MINTO generates LCIs and (10). Because, when (10) is added, the number of nodes is smaller for the largest C instances and the D instances, the number of cuts generated when (10) is added is also smaller.

Overall, with the addition of (10), the total number of nodes was reduced by 53%, and the total CPU time was reduced by 66%. To the best of our knowledge the largest instances we tested are considerably larger than the ones reported in the current literature. For example, Savelsbergh [10] reports computational experience with C and D instances that have up to 1,000 variables. Note also that for the 4 instance dimensions reported on in [10] and by us, our average computational time is smaller.

Acknowledgment We are grateful to an anonymous referee and to the editors for their valuable suggestions.

Table 1: Number of nodes, computational time, and number of cuts

Inst. type	Nodes			Time			Nodes		
	LCI only	LCI & (10)	% Red.	LCI only	LCI & (10)	% Red.	LCI only	LCI & (10)	
								LCI	(10)
C.3.30	62	50	19	0.98	0.94	4	27	21	10
C.3.50	48	37	23	1.21	1.09	10	25	21	13
C.6.30	162	23	86	2.64	1.47	44	21	21	3
C.25.40	1	1	0	3.27	3.44	-5	35	37	9
C.30.40	1	1	0	4.20	4.20	0	25	20	4
C.35.45	1	1	0	4.98	4.95	0	20	20	3
C.20.200	7,106	2,660	63	1,414	562	60	806	492	66
C.20.250	39,935	5,323	87	15,357	2,159	86	2,096	901	76
C.20.300	28,740	5,278	82	14,072	2,791	80	1,956	766	54
C.20.350	29,087	2,380	92	17,738	1,502	92	3,850	789	71
C.20.400	19,715	9,780	50	19,782	8,045	59	3,780	1,797	152
C.20.450	18,183	8,325	54	18,346	8,735	52	2,664	2,069	125
D.3.30	4,521	2,519	44	39	23	41	234	170	107
D.3.50	11,077	6,771	39	148	96	35	827	545	175
D.6.30	30,114	16,041	47	838	438	48	1,109	740	152
D.25.40	41,200	28,411	31	4,717	3,290	30	730	671	112
D.30.40	30,964	26,389	15	3,434	2,852	17	616	608	170
D.35.45	25,871	20,969	19	3,966	3,466	13	685	640	144
TOTAL	286,788	134,959	53	99,868	33,975	66	19,506	10,328	1,446

References

- [1] Cattrysse, D.G., and L.N. van Wassenhove, "A Survey of Algorithms for the Generalized Assignment Problem," *European Journal of Operational Research* 72, 167-174 (1992).
- [2] de Farias, I.R., "A Polyhedral Approach to Combinatorial Complementarity Problems," *Ph.D. Thesis*, School of Industrial and Systems Engineering, Georgia Institute of Technology, Atlanta, GA (1995).
- [3] de Farias, I.R., E.L. Johnson, and G.L. Nemhauser, "A Generalized Assignment Problem with Special Ordered Sets: A Polyhedral Approach," to appear in *Mathematical Programming*.
- [4] Garey, M.R. and D.S. Johnson, *Computers and Intractability: A Guide to the Theory of NP-Completeness*, W.H. Freeman and Company, New York (1979).

- [5] Gottlieb, E.S. and M.R. Rao, “The Generalized Assignment Problem: Valid Inequalities and Facets,” *Mathematical Programming* 46, 31-52 (1990).
- [6] Gottlieb, E.S. and M.R. Rao, “ $(1, k)$ -Configuration Facets for the Generalized Assignment Problem,” *Mathematical Programming* 46, 53-60 (1990).
- [7] Martello, S., and P. Toth, “An Algorithm for the Generalized Assignment Problem,” *Operations Research 1981*, 589-603, Brans, J.P., ed., North-Holland, Amsterdam (1981).
- [8] Nemhauser, G.L., M.W.P. Savelsbergh, and G.C. Sigismondi, “MINTO, a Mixed INTe-ger Optimizer,” *Operations Research Letters* 15, 47-58 (1994).
- [9] Ross, G.T., and R.M. Soland, “A Branch-and-Bound Algorithm for the Generalized Assignment Problem,” *Mathematical Programming* 8, 91-103 (1975).
- [10] Savelsbergh, M.W.P., “A Branch-and-Price Algorithm for the Generalized Assignment Problem,” *Operations Research* 45, 831-841 (1997).
- [11] Savelsbergh, M.W.P., “Functional Description of MINTO, a Mixed INTe-ger Optimizer (version 3.0),” <http://udaloy.isye.gatech.edu/mwps/projects/minto.html>.
- [12] Wolsey, L.A., “Facets and Strong Valid Inequalities for Integer Programs,” *Operations Research* 24, 367-372 (1976).