We propose an approximate lifting procedure for general integer programs. This lifting procedure uses information from multiple constraints of the problem formulation and can be used to strengthen formulations and cuts for mixed integer programs. In particular, we demonstrate how it can be applied to improve Gomory’s fractional cut which is central to Glover’s primal cutting plane algorithm. We show that the resulting algorithm is finitely convergent. We also present numerical results that illustrate the computational benefits of the proposed lifting procedure.

Key words: Integer Programming, Primal Cutting Plane Algorithm

Date: May 20, 2006.

1. Introduction

Most integer programs (IPs) can be successfully solved using a combination of cutting and branching techniques; see Nemhauser and Wolsey (1988), Johnson et al. (2000), and Marchand et al. (2002) for a description of these methods. Cutting plane approaches use inequalities that are satisfied by all integer solutions of the problem but not by some solutions of its linear programming (LP) relaxation. We refer to these inequalities as cuts or cutting planes. There are three main types of cutting plane algorithms for solving integer programs: dual, all-integer and primal.

Dual cutting plane methods proceed by solving LP relaxations of the IP and by iteratively generating cuts to achieve integer feasibility. Dual cutting plane methods are very commonly used in the solution of IPs and have been successfully implemented in commercial software. All-integer cutting plane algorithms maintain a solution that is integer at all times and generate cuts until primal feasibility is achieved. Finally primal cutting plane methods mimic primal simplex by maintaining a primal and integer feasible solution at all times and by generating cuts to prevent simplex moves that create fractional solutions. We
refer to Garfinkel and Nemhauser (1972) and Nemhauser and Wolsey (1988) for a more detailed discussion of these different types of algorithms.

Primal algorithms have obvious advantages over dual and all-integer algorithms. First, a feasible integer solution to the problem is available at each step of the algorithm. Therefore, even if the algorithm is stopped before an optimal solution is reached, a feasible integer solution is available. Second, because primal cutting plane algorithms are similar to primal simplex, they can be used to re-optimize IPs after their objectives are changed.

Primal cutting plane algorithms were first introduced by Ben-Israel and Charnes (1962). The first provably convergent primal cutting plane algorithm was proposed by Young (1965). Young (1968) and Glover (1968) concurrently presented simplified versions of this algorithm that are still finitely convergent. Arnold and Bellmore (1974a,b,c) and Young (1975) proposed various improvements on these algorithms. Although research on primal cutting plane algorithms has been dormant since the 1970’s, some attempts on reviving the approach were recently undertaken; see Sharma and Sharma (1997) and Letchford and Lodi (2002, 2003). We mention that variants of the primal algorithm can be implemented without the use of cuts. An exposition of these algebraic approaches can be found in Aardal et al. (2002). More recent works include Haus et al. (2003), Henk et al. (2003) and Köppe and Weismantel (2003).

The strength of cutting planes added to a formulation, as well as the strength of the formulation itself, are important factors in making any cutting plane algorithm successful. This simple observation has motivated the study of the polyhedral structure of many specific integer and mixed integer programs and has also lead to the introduction of various methods for generating strong cutting planes for unstructured integer and mixed integer programs. This observation is also very relevant for primal cutting plane algorithms where it seems that good computational results can only be obtained when strong cuts are used; see Padberg and Hong (1980) for a discussion of primal cuts for the Traveling Salesman Problem and see Letchford and Lodi (2002) for a discussion of research directions to improve the computational performance of primal cutting plane algorithms.

In this paper we describe an approximate lifting method that generates strong cuts for unstructured integer and mixed integer programs by considering all rows of the problem formulation. Although this technique can be applied in conjunction with all types of cutting plane methods, we focus on its use with primal algorithms. In Section 2, we describe our approximate lifting technique and show that it yields a natural cut improvement procedure.
We also describe strategies to improve the computational efficiency of the method. In Section 3, we show that our cut improvement procedure can be used inside Glover’s primal cutting plane algorithm. In particular, we show that Gomory’s fractional cut, which is central to Glover’s algorithm, can be improved using our approach in a way that yields a finitely convergent algorithm. In Section 4, we present computational results obtained with our variant of Glover’s primal algorithm and show that our cut improvement procedure outperforms more elaborate techniques. Finally we summarize the contributions of this research in Section 5.

2. Lifting for IPs with multiple constraints

Lifting is the process of constructing strong cuts for an MIP from strong cuts for a related MIP with a lesser number of variables; see Padberg (1973, 1975), Balas (1975), Hammer et al. (1975), Wolsey (1976, 1977), Balas and Zemel (1984), Ceria et al. (1998), Gu et al. (1999, 2000), Richard et al. (2003a,b), Atamtürk (2003, 2004), Rajan (2004), and Agra and Constantino (2007) for descriptions of the technique as well as some examples of its application. Because lifting is a general tool, it can be used for problems with multiple constraints; see among many others Gu et al. (1999) for single node flow models and see Zeng and Richard (2006) for knapsack models with cardinality constraints. However, when the problems studied have multiple constraints and no specific structure, lifting is typically harder to apply; see Martin and Weismantel (1998) and Kaparis and Letchford (2008).

In this section, we consider the problem of generating lifting coefficients for unstructured IPs with multiple constraints. Although this problem is difficult in general, we describe an efficient method to obtain approximate lifting coefficients. In Section 2.1, we describe the general problem of computing the lifting coefficient of an integer variable. In Section 2.2, we propose a method to obtain approximate solutions to this problem. This method requires the solution of a single linear program. Finally, in Section 2.3, we show that if several coefficients must be sequentially lifted in a given inequality, the amount of computation required by our lifting approach can be reduced by performing appropriate sequences of primal and dual simplex pivots.

Although the lifting procedure we present is general and can be used to build valid inequalities, we use it only for improving the coefficients of cutting planes obtained using other schemes, i.e., we use the procedure only for preprocessing cuts. A survey of general
ideas for preprocessing is given in Savelsbergh (1994). Other references include Guignard and Spielberg (1981), Johnson et al. (1981), Dietrich and Escudero (1990), Hoffman and Padberg (1991), and Atamtürk et al. (2000).

2.1. Multiple constraint lifting problem

Consider the general IP

$$\begin{align*}
\min & \quad c^T x \\
\text{s.t.} & \quad Ax \leq b \\
& \quad x \in \mathbb{Z}^m
\end{align*}$$ (1)

where $A \in \mathbb{Q}^{n \times m}$, $b \in \mathbb{Q}^{n \times 1}$, and $c \in \mathbb{Q}^{m \times 1}$ and consider an inequality

$$\sum_{j \in M} \alpha_j x_j \leq r$$ (2)

where $M = \{1, \ldots, m\}$ that is valid for (1). Our goal is to strengthen the coefficient $\alpha_m$ of (2) using lifting. More precisely, to determine if an improved coefficient can be found for $x_m$, we will fix the variable $x_m$ to a specific value $k$ and lift it back into the inequality using information from multiple constraints. To simplify the discussion, we assume in the remainder of this paper that $\alpha_j \in \mathbb{Z} \ \forall j \in M$ and that $r \in \mathbb{Z}$. Although this assumption may not always be satisfied in practice, most of the results of this section still apply when it is not. We will specifically indicate the results that depend on the integrality of the $\alpha_j$'s in the following sections.

Assume we fix $x_m$ to $k \in \mathbb{Z}$. Substituting $r - \alpha_m k$ with $r'$, (2) reduces to

$$\sum_{j \in M^*} \alpha_j x_j \leq r'$$ (3)

where $M^* = \{1, \ldots, m-1\}$. To lift $x_m$ back into (3), we compute a value $\beta$ for which

$$\sum_{j \in M^*} \alpha_j x_j + \beta(x_m - k) \leq r'$$ (4)

is a valid inequality for the convex hull of feasible solutions to (1). It is easily verified that admissible values for $\beta$ satisfy the following relations

$$\beta \leq \bar{\beta}^u := \min_{x_m \geq k+1} \frac{1}{x_m-k} \left( r' - \sum_{j \in M^*} \alpha_j x_j \right)$$

s.t.

$$\begin{align*}
\sum_{j \in M^*} A_j x_j & \leq b - A_m x_m \\
x_m & \geq k + 1 \\
x_j & \in \mathbb{Z}_+ \ \forall j \in M
\end{align*}$$ (5)
\[
\beta \geq \beta_l := \max \frac{1}{k-x_m} \left( \sum_{j \in M^*} \alpha_j x_j - r' \right) \\
\text{s.t.} \quad \sum_{j \in M^*} A_j x_j \leq b - A_m x_m \\
x_m \leq k - 1 \\
x_j \in \mathbb{Z}_+ \forall j \in M
\]

where we denote the \( j \)th column of \( A \) in (1) by \( A_j \).

If the problem corresponding to \( \hat{\beta}^u \) (resp. \( \hat{\beta}^l \)) is infeasible then we define \( \hat{\beta}^u = \infty \) (resp. \( \hat{\beta}^l = -\infty \)). We observe that (5) and (6) are difficult problems to solve in general. Therefore finding the values of \( \hat{\beta}^l \) and \( \hat{\beta}^u \) is typically computationally prohibitive. We note that in much of the lifting literature, the above problems are solved for a single-constraint relaxation of (1).

In this work, we propose to solve relaxations of (5) and (6) in which the integrality restrictions of the variables \( x_j \) are relaxed for \( j \in M^* \), but the integrality restriction on \( x_m \) is maintained. After setting \( z_m = x_m - k \), \( w_m = k - x_m \) and \( b' = b - A_m k \), we obtain

\[
\beta \leq \beta^u := \min \frac{1}{z_m} \left( r' - \sum_{j \in M^*} \alpha_j x_j \right) \\
\text{s.t.} \quad \sum_{j \in M^*} A_j x_j \leq b' - A_m z_m \\
z_m \geq 1, \ z_m \in \mathbb{Z} \\
x_j \geq 0 \ \forall j \in M^*
\]

and

\[
\beta \geq \beta^l := \max \frac{1}{w_m} \left( \sum_{j \in M^*} \alpha_j x_j - r' \right) \\
\text{s.t.} \quad \sum_{j \in M^*} A_j x_j \leq b' + A_m w_m \\
w_m \geq 1, \ w_m \in \mathbb{Z} \\
x_j \geq 0 \ \forall j \in M^*
\]

Note that since \( \hat{\beta}^u \geq \beta^u \) and \( \hat{\beta}^l \leq \beta^l \), it is not always the case that \( \beta^l \leq \beta^u \) even if \( \hat{\beta}^l \leq \hat{\beta}^u \).

The bounds \( \beta^u \) and \( \beta^l \) we propose in (7) and (8) to derive lifting coefficients are obtained by solving nonlinear continuous relaxations of the initial lifting problems (5) and (6). When \( r' \) and \( \alpha_j \) are integers for \( j \in M^* \) these bounds can be strengthened using the following rounding operations

\[
\beta \leq \beta^u := \min \frac{1}{z_m} \left( r' - \sum_{j \in M^*} \alpha_j x_j \right) \\
\text{s.t.} \quad \sum_{j \in M^*} A_j x_j \leq b' - A_m z_m \\
z_m \geq 1, \ z_m \in \mathbb{Z} \\
x_j \geq 0 \ \forall j \in M^*
\]

and

\[
\beta \geq \beta^l := \max \frac{1}{w_m} \left( \sum_{j \in M^*} \alpha_j x_j - r' \right) \\
\text{s.t.} \quad \sum_{j \in M^*} A_j x_j \leq b' + A_m w_m \\
w_m \geq 1, \ w_m \in \mathbb{Z} \\
x_j \geq 0 \ \forall j \in M^*
\]
since the actual variables \( x_j \) are integers for \( j \in M^* \). One way to compute the bounds \( \beta^u \) and \( \beta^l \) of (9) and (10) is to solve a sequence of LPs each corresponding to a different value of the integer variables \( z_m \) and \( w_m \). To this end, we define \( \beta^u \) and \( \beta^l \) as

\[
\beta^u_i := \min \frac{1}{i} \left[ r' - \sum_{j \in M^*} \alpha_j x_j \right]
\text{s.t.} \quad \sum_{j \in M^*} A_j x_j \leq b' - A_m i
\quad x_j \geq 0 \ \forall j \in M^*
\]  

(11)

and

\[
\beta^l_i := \max \frac{1}{i} \left| \sum_{j \in M^*} \alpha_j x_j - r' \right|
\text{s.t.} \quad \sum_{j \in M^*} A_j x_j \leq b' + A_m i
\quad x_j \geq 0 \ \forall j \in M^*
\]  

(12)

It is clear that, if there exists a value of \( \beta \) that satisfies (9) and (10), it can be computed as

\[
\max \{ \beta^l_i \mid i \geq 1, i \in \mathbb{Z}_+ \} \leq \beta \leq \min \{ \beta^u_i \mid i \geq 1, i \in \mathbb{Z}_+ \}.
\]

This shows that it is possible to obtain \( \beta \) by solving one linear program for each value of \( i \). We next show that \( \beta \) can be obtained by solving no more than two linear programs. The result drastically reduces computation in the case when \( x_m \) is a general integer variable. The result is also useful for 0–1 programs as slacks are typically general integer variables.

### 2.2. Efficient solution of the multiple constraint lifting problem

In this section we study some properties of the value functions of LPs related to the computation of lifting coefficients. We then use these properties to determine the values of \( \beta^u \) and \( \beta^l \) and also the values of \( \beta^\mu \) and \( \beta^\ell \). We first introduce the following definition.

**Definition 1** Let \( S^+ = \{ \lambda \in \mathbb{R} \mid \exists x \ \text{s.t.} \ \sum_{j \in M^*} A_j x_j + A_m \lambda \leq b', x_j \geq 0 \ \forall j \in M^* \} \) and \( S^- = \{ \lambda \in \mathbb{R} \mid -\lambda \in S^+ \} \). We define \( \Gamma^+: S^+ \rightarrow \mathbb{R} \) to be the function

\[
\Gamma^+(\lambda) = r' + \min \ \frac{1}{i} \left[ \sum_{j \in M^*} (-\alpha_j) x_j \right]
\text{s.t.} \quad \sum_{j \in M^*} A_j x_j + A_m \lambda \leq b'
\quad x_j \geq 0 \ \forall j \in M^*.
\]  

(13)

Similarly we define \( \Gamma^-: S^- \rightarrow \mathbb{R} \) to be the function

\[
\Gamma^-(\lambda) = -r' + \max \ \frac{1}{i} \left| \sum_{j \in M^*} \alpha_j x_j \right|
\text{s.t.} \quad \sum_{j \in M^*} A_j x_j - A_m \lambda \leq b'
\quad x_j \geq 0 \ \forall j \in M^*.
\]  

(14)

For a given \( \lambda \in S^+ \) (resp. \( \lambda \in S^- \)), we say that \( \Gamma^+(\lambda) \) (resp. \( \Gamma^-(\lambda) \)) is bounded if the LP in (13) (resp. (14)) has an optimal solution.
In Definition 1, \( S^+ \) represents the set of values of \( x_m \) for which the linear program \( \sum_{j \in M^*} A_j x_j + A_m x_m \leq b', x_j \geq 0, \forall j \in M^* \) has feasible solutions. Similarly, \( S^- \) represents the set of values of \( x_m \) for which the linear program \( \sum_{j \in M^*} A_j x_j - A_m x_m \leq b', x_j \geq 0, \forall j \in M^* \) has feasible solutions. It is easy to verify that \( S^+ \) and \( S^- \) are convex and that \( \Gamma^+(\lambda) = -\Gamma^-(\lambda) \). It is also easy to derive that

\[
\bar{\beta}^u = \begin{cases} 
\min \left\{ \frac{\Gamma^+(j)}{j} \mid j \in S^+, j \geq 1, j \in \mathbb{Z} \right\} & \text{if } S^+ \cap \{ j \in \mathbb{Z}, j \geq 1 \} \neq \emptyset, \\
\infty & \text{otherwise}
\end{cases}
\]

\[
\bar{\beta}^l = \begin{cases} 
\max \left\{ \frac{\Gamma^-(j)}{j} \mid j \in S^-, j \geq 1, j \in \mathbb{Z} \right\} & \text{if } S^- \cap \{ j \in \mathbb{Z}, j \geq 1 \} \neq \emptyset, \\
-\infty & \text{otherwise}
\end{cases}
\]

and that

\[
\beta^u = \begin{cases} 
\min \left\{ \frac{[\Gamma^+(j)]}{j} \mid j \in S^+, j \geq 1, j \in \mathbb{Z} \right\} & \text{if } S^+ \cap \{ j \in \mathbb{Z}, j \geq 1 \} \neq \emptyset, \\
\infty & \text{otherwise}
\end{cases}
\]

\[
\beta^l = \begin{cases} 
\max \left\{ \frac{[\Gamma^-(j)]}{j} \mid j \in S^-, j \geq 1, j \in \mathbb{Z} \right\} & \text{if } S^- \cap \{ j \in \mathbb{Z}, j \geq 1 \} \neq \emptyset, \\
-\infty & \text{otherwise}
\end{cases}
\]

In the remainder of this paper, we assume that \( \Gamma^+(\lambda) \) and \( \Gamma^-(\lambda) \) are bounded for all \( \lambda \in S^+ \) and \( \lambda \in S^- \) respectively. This assumption is typically verified in practice. In particular, it can be proven to hold when either the problem (1) is feasible and \( A \) and \( b \) are rational, or when LPR is bounded, where \( LPR = \{ x \in \mathbb{R}^m_+ \mid Ax \leq b \} \); see Proposition 13 in Appendix.

Since we assumed that \( \Gamma^+(\lambda) \) is bounded, we can define \( \mu^+(\lambda) \) (resp. \( \mu^-(\lambda) \)) to be the set of all optimal dual solutions corresponding to \( \Gamma^+(\lambda) \) (resp. \( \Gamma^-(\lambda) \)). Next we present a classical result based on linear programming duality that makes the computation of \( \beta \) simple.

**Proposition 2** The function \( \Gamma^+ \) is piecewise linear and convex over \( S^+ \). Furthermore if \( y \in \mu^+(\lambda) \), then \( -y^T A_m \) is a subgradient of the function \( \Gamma^+ \) at \( \lambda \). \( \Box \)

We next present a similar result for \( \Gamma^- \). Note that we say that a vector \( v_x \) is a supergradient of a concave function \( f \) at a point \( x \) of the domain of \( f \) if \( f(y) - f(x) \leq v_x (y - x) \) for any \( y \) in the domain of \( f \).

**Proposition 3** The function \( \Gamma^- \) is piecewise linear and concave over \( S^- \). Furthermore if \( y \in \mu^-(\lambda) \), then \( y^T A_m \) is a supergradient of the function \( \Gamma^- \) at \( \lambda \). \( \Box \)
Proposition 4 presents an efficient method for computing $\bar{\beta}^u$ and $\bar{\beta}^l$. Its proof, given in the appendix, is based on the convexity of $\Gamma^+$ (resp. the concavity of $\Gamma^-$), which was shown in Proposition 2 (resp. Proposition 3).

**Proposition 4** Assume that $x_m$ is fixed to a value $k$ for which there exists at least one point of the linear programming relaxation of (1) with $x_m = k$ (i.e., $0 \in S^+$). If $\Gamma^+(0) \leq 0$ and

1. $1 \in S^+$ then $\bar{\beta}^u = \Gamma^+(1)$.

2. $1 \in S^-$ then $\bar{\beta}^l = \Gamma^-(1)$.

Therefore, if $\Gamma^+(0) \leq 0$, only one linear program must be solved in order to obtain an upper (or lower) bound of $\beta$.

The condition $\Gamma^+(0) \leq 0$ is satisfied if (3) is tight for some point of the LP relaxation of $
\sum_{j \in M} A_j x_j \leq b', \; x_j \in \mathbb{Z}_+, \forall j \in M^*$, or if a point $x^*$ of the LP relaxation of $\sum_{j \in M} A_j x_j \leq b', \; x_j \in \mathbb{Z}_+, \forall j \in M^*$ is violated by (3). Therefore, the condition $\Gamma^+(0) \leq 0$ implies that we attempt to improve a cut that already cuts off some part of the LP relaxation or is at least tight for the LP relaxation.

Next we study the derivation of the bounds $\beta^u$ and $\beta^l$ which are stronger than the bounds $\bar{\beta}^u$ and $\bar{\beta}^l$ derived in Proposition 4. These bounds assume that the original cut had only integral coefficients. Proposition 5 presents conditions under which the derivation of these bounds is easy.

**Proposition 5** If $1 \in S^+$ and $\exists y \in \mu^+(1)$ such that $y^T A_m \leq -[\Gamma^+(1)]$, then $\beta^u = \beta^u_1$.

**Proof** By definition of $\beta^u_1$ and $\Gamma^+$, we have $\beta^u_1 = \frac{\lceil \Gamma^+(0) \rceil}{i}$. Let $y \in \mu^+(1)$. For $\delta \in \mathbb{Z}_+$ such that $\delta + 1 \in S^+$ we conclude from Proposition 2 that

$$\Gamma^+(1 + \delta) \geq \Gamma^+(1) - \delta y^T A_m.$$  

(15)

It follows that

$$\lceil \Gamma^+(1 + \delta) \rceil \geq \lceil \Gamma^+(1) - \delta y^T A_m \rceil \geq \lceil \Gamma^+(1) \rceil - \lceil \delta y^T A_m \rceil$$  

(16)

since the ceiling function is subadditive and monotone. Because $y^T A_m \leq -[\Gamma^+(1)]$ we obtain, $\lceil \delta y^T A_m \rceil \leq \lceil -\delta \Gamma^+(1) \rceil = -\delta \lceil \Gamma^+(1) \rceil$ or

$$-\lceil \delta y^T A_m \rceil \geq \delta \lceil \Gamma^+(1) \rceil.$$  

(17)
From (16) and (17) we conclude that $\left\lceil \Gamma^+(1 + \delta) \right\rceil - \left\lfloor \Gamma^+(1) \right\rfloor \geq \delta \left\lfloor \Gamma^+(1) \right\rfloor$, i.e.,

$$\frac{\left\lceil \Gamma^+(1 + \delta) \right\rceil}{1 + \delta} \geq \left\lfloor \Gamma^+(1) \right\rfloor.$$  

(18)

This implies that $\bar{\beta}_j^i \geq \bar{\beta}_i^i \forall j \geq 1$, which proves the result. □

The following result can be proven similarly.

**Proposition 6** If $1 \in S^-$ and $\exists w \in \mu^-(1)$ such that $w^T A_m \leq \left\lfloor \Gamma^-(1) \right\rfloor$, then $\beta^l = \bar{\beta}_1^l$. □

Next we combine the results of Propositions 4, 5 and 6 to obtain a general procedure to compute approximate lifting coefficients. The underlying idea is that the results of Propositions 5 and 6 should be applied whenever the coefficients of the original cut are integral and there exists an adequate subgradient in the subdifferential of $\Gamma^+$. When these assumptions are not satisfied, the weaker result of Proposition 4 is used. Note here that we use $\bar{\beta}_u^i$ (resp. $\bar{\beta}_l^i$) to denote the optimal value of (7) (resp. (8)) when $z_m = 1$ (resp. $w_m = 1$).

**Proposition 7** Assume that $x_m$ is fixed at a value $k$ for which there exists at least one point of the linear programming relaxation of (1) with $x_m = k$, and (3) is violated by a point $x^*$ of the LP relaxation of $\sum_{j \in M^*} A_j x_j \leq b', x_j \in \mathbb{Z}_+ \forall j \in M^*$. Define

$$U = \begin{cases} \infty & \text{if } 1 \notin S^+ \\ \bar{\beta}_u^i & \text{if } 1 \in S^+ \text{ and } \exists y \in \mu^+(1) \text{ with } y^T a_k \leq -\left\lceil \Gamma^+(1) \right\rceil \\ \bar{\beta}_i^i & \text{otherwise} \end{cases}$$

(19)

and

$$L = \begin{cases} -\infty & \text{if } 1 \notin S^- \\ \bar{\beta}_l^i & \text{if } 1 \in S^- \text{ and } \exists w \in \mu^-(1) \text{ with } w^T a_k \leq \left\lfloor \Gamma^-(1) \right\rfloor \\ \bar{\beta}_i^i & \text{otherwise.} \end{cases}$$

(20)

Then $U$ and $L$ are valid upper and lower bounds for $\beta$ if $U \geq L$. □

Proposition 7 significantly reduces the computational burden of the lifting procedure of Section 2.2 since it implies that it is sufficient to solve the linear programs corresponding to $\Gamma^+(1)$ and $\Gamma^-(1)$ to obtain bounds on the lifting coefficients. However, there are still two possible difficulties with this scheme:

1. If the bounds $U$ and $L$ contradict each other, then approximate lifting is not possible.
2. The result does not guarantee that an integer value is found for the lifting coefficient, i.e., Proposition 7 does not guarantee that \( \lfloor \bar{\beta}_u \rfloor \geq \lceil \bar{\beta}_l \rceil \). This is because it is possible that \( p < \bar{\beta}_l < \bar{\beta}_u < p + 1 \) for some integer \( p \). Then \( \lfloor \bar{\beta}_u \rfloor = p < p + 1 = \lceil \bar{\beta}_l \rceil \).

Although it is typically possible to achieve integral coefficients in a cut by scaling, there are some circumstances where it may be necessary to obtain an integer value \( \beta \) without scaling. This is the case, for example, in primal cutting plane algorithms as we will see in Section 3.

One possible way to overcome these difficulties is to choose \( k \) to be the lower or upper bound of \( x_m \). In particular when \( k \) is chosen to be the lower (resp. the upper bound) on \( x_m \), only one lifting problem needs to be solved, as setting \( \beta = U \) (resp. \( \beta = L \)) provides a valid lifting coefficient. We obtain in this case that the lifted coefficient are integral as shown in the following corollary (since \( \bar{\beta}_u \) is integral).

**Corollary 8** Suppose that there exists at least one point of the linear programming relaxation of (1) with \( x_m = 0 \). If \( x_m \) is fixed at 0 and (3) is violated by a point \( x^* \) of the LP relaxation of \( \sum_{j \in M^*} A_j x_j \leq b, x_j \in \mathbb{Z}_+ \forall j \in M^* \), then the inequality \( \sum_{j \in M^*} \alpha_j x_j + \beta x_m \leq r \) is valid if

\[
\beta \leq \begin{cases} 
\infty & \text{if } 1 \notin S^+ \\
\bar{\beta}_u & \text{if } 1 \in S^+ \text{ and } \exists y \in \mu^+(1) \text{ with } y^T a_k \leq -[\Gamma^+(1)] \\
\bar{\beta}_u - 1 & \text{otherwise.}
\end{cases}
\]  

(21)

Observe that in Corollary 8, \( U = \infty \) implies that the column corresponding to \( x_m \) can be dropped from the problem formulation as the only integer value \( x_m \) takes is 0.

We end this section by observing that the lifting method proposed here is designed for integer variables and that a straightforward extension of the idea does not work for continuous variables in general. To illustrate this remark, consider the simple case where a continuous variable \( y_m \) is lifted from its lower bound, which we assume is 0. After relaxing the integrality restrictions of the integer variables \( x_j \) for \( j \in M^* \), we obtain the following lifting problem (similar to the definition of \( \bar{\beta}_u \))

\[
\inf \frac{1}{y_m} \left( r - \sum_{j \in M^*} \alpha_j x_j \right) \\
\text{s.t.} \quad \begin{align*}
\sum_{j \in M^*} A_j x_j & \leq b - A_m y_m \\
y_m & > 0 \\
x_j & \geq 0 \forall j \in M^*
\end{align*}
\]  

(22)
Since we are strengthening an existing cutting plane, such as the Gomory cut, we may assume that $\Gamma^+(0) < 0$. This implies that for arbitrarily small values of $y_m$, the objective function of (22) typically tends towards $-\infty$, which does not yield an useful coefficient. To obtain useful coefficients, we could propose to either work with a variant of the lifting problem where integrality of the variables $x_j$ for $j \in M^*$ is not relaxed or to use in (22) a lower bound on the value of $y_m$ obtained from the exact lifting problem. However both of these options require significant modifications to our algorithm.

2.3. Lifting multiple integer variables

For practical purposes, improving all coefficients of a given inequality using the schemes presented in Proposition 7 or in Corollary 8 can be time-consuming as it requires the solution of a different LP for each coefficient. This computational burden may be reduced by studying the sequence of linear programs that are solved. Assume that we wish to sequentially improve the coefficients corresponding to two different variables $x_u$ and $x_v$ in a valid inequality. The LP that has to be solved to determine the improved coefficient of $x_u$ is

$$\min \ - \sum_{j \in M \setminus \{u\}} \alpha_j x_j$$
$$s.t. \ \sum_{j \in M \setminus \{u\}} A_j x_j + A_u \leq b$$
$$x_j \geq 0 \quad \forall j \in M \setminus \{u\}$$

while the LP that has to be solved to improve the coefficient of $x_v$ is

$$\min \ - \sum_{j \in M \setminus \{u,v\}} \alpha_j x_j + \alpha^*_u x_u$$
$$s.t. \ \sum_{j \in M \setminus \{v\}} A_j x_j + A_v \leq b$$
$$x_j \geq 0 \quad \forall j \in M \setminus \{v\}$$

where $\alpha^*_u$ is the improved coefficient of $x_u$ obtained from (23).

Because of their similarity, it is possible to obtain an optimal solution to (24) from an optimal solution of (23) using the following sequence of primal and dual simplex pivots. We sketch the sequence of pivoting operations required; see Chvátal (1983) (page 161) for details.

First, starting from an optimal basis of (23) we obtain an optimal basis for the intermediate problem

$$\min \ - \sum_{j \in M \setminus \{u\}} \alpha_j x_j + \alpha^*_u x_u$$
$$s.t. \ \sum_{j \in M} A_j x_j \leq b - A_u$$
$$x_j \geq 0 \quad \forall j \in M$$

by introducing $x_u$ as a nonbasic variable in the optimal tableau of (23) and by performing primal simplex pivots. Second, using an optimal basis of (25) we obtain an optimal basis
of (24). To do so, we observe that a feasible basis to the dual of (24) can be obtained from an optimal dual basis of (25). Starting from this dual feasible basis for (24), we obtain the optimal basis for (24) using dual simplex pivots.

3. Modified Glover’s algorithm

In this section we use the general cut improvement procedure designed in Section 2 to strengthen Glover’s primal cutting plane algorithm. We first present in Section 3.1 a brief overview of Glover’s algorithm. We then apply in Section 3.2 our cut strengthening procedure to improve Gomory’s fractional cut, which is central to Glover’s algorithm. We show that the resulting primal algorithm is finitely convergent.

3.1. Review of Glover’s algorithm

In 1968, Glover introduced a simplified version of Young’s (1965) algorithm that we present in Table 1. A very good exposition of the details of this algorithm is given in Garfinkel and Nemhauser (1972).

<table>
<thead>
<tr>
<th>Table 1: Glover’s algorithm</th>
</tr>
</thead>
</table>

1. Initialize the algorithm with a simplex tableau in which all entries are integer.
   - Row 0 of the tableau is the row of reduced costs.
   - Column 0 of the tableau is the column of right-hand-sides.

2. Identify a special row (See Definition 9). Add one if necessary.


4. If the pivot element is greater than or equal to 2, go to Step 5.
   Else, do a normal simplex pivot and go to Step 8.

5. Select the source row for cut generation using Rule 2.

6. Generate Gomory’s fractional cut. Add it to the bottom of the tableau.
   Pivot.

7. If the new basic variable is a slack variable, drop the pivot row.

8. If all reduced costs are of the desired sign; STOP.
   Else go to Step 3.
Glover’s algorithm is similar to primal simplex algorithm. It maintains a primal and integer feasible solution of the IP problem inside a simplex tableau. If a nonbasic variable can be entered in the basis that improves the objective value of the current solution, a pivot is made provided that the new solution is still integer. If the new solution is not integer, a cut is added to prevent the move. The cut that is added in Glover’s algorithm is Gomory’s fractional cut. It is constructed in the following fashion. Consider the source row \( r \) of the simplex tableau, 
\[
x_{b(r)} + \sum_{j \in N} a_{rj} x_j = a_{ro}
\]
where \( N \) is the set of nonbasic variables in the tableau and \( x_{b(r)} \) is the basic variable of row \( r \). Let \( x_k \) be the entering variable and assume that \( a_{rk} > 1 \). Then Gomory fractional cut that is added to prevent the fractional move is of the form
\[
s + \sum_{j \in N} \left\lfloor \frac{a_{rj}}{a_{rk}} \right\rfloor x_j = \left\lfloor \frac{a_{ro}}{a_{rk}} \right\rfloor,
\]
where \( s \) is a slack variable. Glover (1968) proved that under a strict discipline for selecting the entering variable and for selecting the source row for cut generation, the algorithm of Table 1 converges in a finite number of steps. The main requirements to ensure convergence in Glover’s algorithm are presented next. We consider first the selection of the entering variable. In order to define this rule, Glover uses a special row that is identified in Step 2 of the algorithm. The following discussion is from Garfinkel and Nemhauser (1972).

**Definition 9** For \( u, v \in \mathbb{R}^n \), we use \( u \prec^L v \) to describe that \( u \) lexicographically precedes \( v \). The special row (whose index we denote by \( * \)) in a tableau must satisfy the following properties

1. If \( a_j \prec^L \bar{0}, j \in N \), then \( a_{*j} \geq 1 \).

2. Define \( N_* = \{ j | j \in N, a_{*j} \geq 1 \} \), \( n_j = a_j/a_{*j} \) for \( j \in N \) and \( n_k = \text{lexmin}_{l \in N_*} \{ n_l \} \). If \( a_{*j} \leq -1 \), then \( n_j \prec^L n_k \)

where \( a_j \) is the column corresponding to variable \( x_j \) in the simplex tableau.

If the problem studied does not contain a special row, a redundant constraint can be added to the problem to serve as one. For example, the inequality \( \sum_{j \in N} x_j \leq M \), where \( M \) is an integer number chosen sufficiently large for the inequality to be valid is a special row. In Glover’s algorithm, the entering variable selection criteria is based on the special row in the following fashion.
Rule 1 The variable $x_k$ selected to enter the basis is such that $n_k = \text{lexmin}_{l \in N_\ast \{n_l\}}$.

Observe first that a column satisfying Rule 1 always exists. This is because whenever the current solution is not proven to be optimal, there exists a column with negative reduced cost, i.e. there is a column with $a_{0j} < 0$, or $a_j \prec^L 0$. Hence, from the definition of the special row, $a_{*j} \geq 1$. Thus, $N_\ast \neq \emptyset$. Next we describe the rule that Garfinkel and Nemhauser (1972) used for source row selection. We will use this rule in this paper.

Rule 2 The source row to generate a cut is selected as follows:

1. If $a_{*k} > a_{*0}$, choose the special row as the source row.

2. If $a_{*k} \leq a_{*0}$ and there exists a row $i \neq 0$ with $a_{ik} > 0$ and $\frac{a_{ik}}{a_{*k}} > a_{i0}$ choose the topmost of these rows (say $i = q$) as the source. After pivoting, if $\frac{a_{qk_1}}{a_{*k_1}} > a_{i0}$, use $q$ as the source for $t$ consecutive cuts, until
   \begin{equation}
   \frac{a_{qk_t}}{a_{*k_t}} \leq a_{q0} < \frac{a_{qk_{t-1}}}{a_{*k_{t-1}}},
   \end{equation}
   When (27) applies, return to 1. for the next iteration.

3. It neither 1. nor 2. is satisfied, choose any row $q$ with $a_{qk} \geq 1$ and $\left[\frac{a_{q0}}{a_{qk}}\right] = \left[\frac{a_{r0}}{a_{rk}}\right] \leq \frac{a_{r0}}{a_{rk}}$ (where $x_{b(r)}$ is the leaving variable) as the source row.

3.2. Applying lifting to improve Glover’s algorithm

Our idea is to strengthen Gomory’s fractional cut (26) which is central to Glover’s algorithm by using information from multiple rows of the tableau simultaneously. Note that one straightforward way to improve Gomory’s fractional cut is to use Gomory’s mixed integer cut (GMIC). However, GMIC introduces fractional entries in the tableau that are not compatible with Glover’s algorithm. In fact, adding cuts with only integer coefficients is crucial to the proof of finite convergence of the resulting algorithm. Our technique has the advantage that it strengthens Gomory’s fractional cut without introducing fractional coefficients and therefore can easily be incorporated into Glover’s algorithm. It is applied as follows. We fix one of the non-basic variables other than the entering variable to 0. Let $l$ be the index of this variable. Observe that because $x_l$ is a nonbasic variable, there exists
at least one point of the linear programming relaxation of (1) with \( x_l = 0 \). Using rounding, we obtain

\[
\sum_{j \in \mathcal{N} \setminus \{l\}} \left\lfloor \frac{a_{rj}}{a_{rk}} \right\rfloor x_j \leq \left\lfloor \frac{a_{r0}}{a_{rk}} \right\rfloor
\]

which is valid for the convex hull of integer solutions to

\[
\sum_{j \in \mathcal{B} \cup \mathcal{N} \setminus \{l\}} a_j x_j = a_0, \quad x \in \mathbb{Z}_+^{B \cup N - 1}.
\]

It is easy to verify that (28) is violated by a fractional vertex of the LP relaxation of (29). Therefore we can use Corollary 8 to compute a value of \( \beta \) such that

\[
\sum_{j \in \mathcal{N} \setminus \{l\}} \left\lfloor \frac{a_{rj}}{a_{rk}} \right\rfloor x_j + \beta x_l \leq \left\lfloor \frac{a_{r0}}{a_{rk}} \right\rfloor
\]

is valid for the original problem. We use the maximum of \( \beta \) and \( \left\lfloor \frac{a_{rl}}{a_{rk}} \right\rfloor \) as the coefficient for \( x_l \). We record this result in the following proposition.

**Proposition 10** Let \( \tilde{\beta} = \max \left\{ \beta, \left\lfloor \frac{a_{rl}}{a_{rk}} \right\rfloor \right\} \) where \( \beta \) is computed for (29) using Corollary 8 on the cut (28). Then

\[
\sum_{j \in \mathcal{N} \setminus \{l\}} \left\lfloor \frac{a_{rj}}{a_{rk}} \right\rfloor x_j + \tilde{\beta} x_l \leq \left\lfloor \frac{a_{r0}}{a_{rk}} \right\rfloor
\]

is a valid inequality for (1). \( \square \)

We use Proposition 10 repeatedly to improve the coefficients of all the different variables. In each use of Proposition 10, (28) is updated to reflect the changes in coefficients of the previously lifted variables. We then used the resulting inequality in place of Gomory’s fractional cut inside Glover’s algorithm to prevent fractional pivots. We refer to this new algorithm as modified Glover’s algorithm (MGA).

Garfinkel and Nemhauser (1972) give a proof that Glover’s algorithm is finitely convergent. Although their proof is given for the particular case of Gomory’s fractional cut, it is easily seen that it generalizes to other families of cuts provided that three conditions are satisfied. These conditions are presented in the following proposition.

**Proposition 11** Consider a family of cutting planes with integer coefficients. Assume that, given any fractional row of a simplex tableau, it is possible to generate a violated cutting plane in the family. Let \( k, k_1, \ldots \) be the sequence of indices of entering variables. Then Glover’s algorithm based on this family of cuts is finitely convergent if

1. The properties of the special row are maintained after a cut is added and pivots are performed.

2. If \( a_{sk} > a_{s0} \), then for some finite number \( t \), \( a_{skt}^t \leq a_{s0}^t \) on consecutive applications of Part 1 of Rule 2 where \( a_{ij}^t \) represent the \((i,j)^{th}\) entry of the tableau obtained after adding \( t \) cuts and pivoting.
3. If \( a_{qk} > 0 \) and \( \frac{a_{qk}}{a_{*k}} > a_{q0} \), then there exists some finite number \( t \), so that \( \frac{a_{qk}}{a_{*k}} \leq a_{q0} \) on consecutive applications of Part 2 of Rule 2.

We now apply the result of Proposition 11 to show that our improved variant of Glover’s algorithm is finitely convergent. We refer to the appendix for a proof.

**Proposition 12** MGA converges in a finite number of steps.

4. **Computational experiments**

In this section we computationally test the algorithms described in Section 3. First, we describe in Section 4.1 the test instances on which we carry our computational evaluation. We then describe in Section 4.2 the computing environment on which the evaluation is performed. Next, in Section 4.3 we present four different strategies we considered for deciding the order in which the coefficients of the Gomory fractional cut are improved. This order is significant as the amount of improvement obtained for a cut coefficient depends on its order in the improvement sequence. Finally, in Section 4.4 we describe the results of our computational experiment aimed at evaluating (i) the importance of the lifting order in the cut improvement algorithm, (ii) the impact of using restart strategies discussed in Section 2.3 in reducing cut improvement times and (iii) the quality of modified Glover’s algorithm as compared to the original Glover’s algorithm and also the more sophisticated algorithms of Letchford and Lodi (2002).

4.1. **Test instances**

We evaluate our algorithms on three sets of instances with different characteristics.

The first set of instances we use is composed of 0-1 multi-dimensional knapsack problems that were obtained from Letchford and Lodi (2002). These problems are known to be hard for their sizes. Briefly, these problems are constructed with \( m = 5, 10, 15, 20 \) and 25 variables and \( n = 5 \) and 10 constraints. Because five instances are created for each combination of parameters \( m \) and \( n \), this test set contains fifty instances. The objective is to maximize and the objective coefficients are generated uniformly between 1 and 10. The instances with 5 constraints are dense and each coefficient of the constraint matrix is generated uniformly between 1 and 10. For instances with 10 constraints, each entry in the coefficient matrix is either 0 with probability 0.5 or is generated uniformly between
1 and 10 with probability 0.5. The right-hand-sides of constraints are computed as half the sum of the coefficients in each constraint. We add the special row $\sum_{i=1}^{m} x_i \leq m$ to these problems, where the $x_i$’s are the original binary variables of the problem. Because the point $(0, \ldots, 0)$ is always feasible in these randomly generated problems, we will always start primal algorithms from this solution.

The second set of instances we use is composed of general integer multi-dimensional knapsack problems. The reason that we consider these instances is that we want to test our algorithm with problems that naturally have general integer variables (the only general integer variables that the first set of instances contains are the slack variables.) Because there does not seem to exist a standard set of instances of general integer problems that are solved using primal cutting plane algorithms, we generate our own set of 50 new random problems. To generate these problems, we use the same specifications for the objective function, constraints and right-hand-sides as in Letchford and Lodi (2002), except that variables are allowed to assume general integer values. The special row for these problems is $\sum_{i=1}^{m} x_i \leq \tilde{m}$, where the $x_i$’s are the original variables of the problem and $\tilde{m}$ is obtained by solving the LP $\max \{ \sum_{i=1}^{m} x_i \mid Ax \leq b \}$ and then by rounding down the optimal value. Again as the point $(0, \ldots, 0)$ is always feasible in these randomly generated problems, we always starts primal algorithms from the solution $(0, \ldots, 0)$.

The last set of problems we consider is composed of the two smallest 0-1 integer problems from MIPLIB (Bixby et al. 1992). These problems ($p0033$ and $p0040$) have relatively few variables and therefore should be amenable to solution by primal cutting plane algorithms. In particular, $p0033$ is a binary IP with 16 rows and 33 columns and $p0040$ is a binary IP with 23 rows and 40 columns. The special row in each of these cases is generated as $\sum_{i=1}^{m} x_i \leq m$, where the $x_i$’s are the original binary variables of the problem. Because there is no obvious feasible solution for these problems, we generated twenty different starting primal feasible solutions by optimizing these problems over different objectives. We then ran Glover’s algorithm and MGA starting from these solutions.

4.2. Implementation details

We implement MGA and Glover’s algorithm in C since Glover’s algorithm involves very specific pivoting rules. In the spirit of the original finitely convergent algorithm of Glover, we handle upper bounds on variables explicitly, i.e. upper bounds on variables are treated as additional constraints. Both algorithms are stopped if the maximum entry in the tableau
exceeds 1,000,000 due to the increased likeliness that the algorithm will run into numerical errors. Further, MGA is stopped if more than 500 lifted cuts are added without augmenting the solution. Finally MGA and Glover’s algorithm are stopped if they run for more than 10 minutes without augmenting the solution.

The coefficient improvement procedure described in Corollary 8 is implemented in two ways. In the first approach, all linear programs whose solutions are needed to obtain improved coefficients are solved independently. In the second approach, all linear programs needed to obtain improved coefficients in a given cut are solved using the sequence of primal and dual simplex pivoting steps presented in Section 2.2. In both approaches, the linear programs are solved using CPLEX 10.1. The second approach was implemented using the advanced basis feature in CPLEX and by enforcing the type of simplex algorithm used.

4.3. Coefficient improvement sequence

One advantage of primal algorithms is that, through the knowledge of reduced cost it is possible to evaluate directions in which the objective function value improves. Note also that when coefficients are being improved sequentially, the coefficient improvement of a variable considered earlier in the sequence is likely to be larger than the coefficient improvement of a variable improved later. Therefore we decide to order the variables for coefficient improvement as follows. For each of the nonbasic variable we compute the ratio of the current reduced cost and the entry in the special row. We call these ratios normalized reduced costs. We then sort variables in increasing order of normalized reduced costs (i.e., from most negative to most positive) and improve the coefficients in this order. The details of the procedure are described next.

1. We generate Gomory’s fractional cut.

2. We perform a tentative pivot operation after adding Gomory’s fractional cut. We obtain the new reduced costs.

3. We divide the reduced cost of a variable (after the pivot operation), if negative, by the entry in the special row. Note that the special row has the property that the entry in the special row is strictly positive if the reduced cost is negative.

4. We sort the resulting vector of normalized reduced costs in increasing order.
5. We apply the coefficient improvement procedure in the above order using Proposition 10.

To determine whether this lifting sequence is significant in the overall algorithm, we compare computational results obtained using the above sequence with the following three lifting sequences.

1. *Increasing reduced cost sequence:* In this variant, variables are ordered in increasing magnitude of reduced cost instead of normalized reduced costs. The motivation behind selecting this sequence for lifting is to test whether small changes in the lifting sequence produce drastic changes in the results, i.e., to test if the algorithm behaves robustly with respect to small changes in the lifting order.

2. *Random sequence:* In this variant, the coefficient improvement sequence is chosen randomly (without regard to the sign or magnitude of the reduced costs). The motivation behind selecting this sequence for lifting is to test whether there is any significant effect in choosing the lifting sequence based on reduced costs or whether any lifting sequence produces similar results.

3. *Decreasing normalized reduced cost sequence:* In this variant, we reverse the order suggested by the normalized reduced costs. The motivation behind considering this order is that it is conceivable that the lifting sequence is important, but the normalized reduced cost based lifting sequence is a poor order. In such a case, the reverse order might work well.

### 4.4. Experimental results

In this section, we present the results of our computational tests. First, we describe in Tables 2 and 3 the performance of different primal algorithms on the first set of instances, which were obtained from Letchford and Lodi (2002). We present the results obtained by Glover’s algorithm and by our improved version of Glover’s algorithm (MGA) with the four different lifting sequences described in Section 4.3. We also present the results obtained by Letchford and Lodi (2002) under the name ‘Improved Young’s Algorithm’ (IYA). For rows corresponding to variants of MGA, the column labeled ‘Gom. Cuts’ gives the average number of cuts that could not be improved using the procedure of Section 2. For the row corresponding to IYA, this entry represents the number of cuts added that were not lifted.
cover inequalities; this data is from Letchford and Lodi (2002). For rows corresponding to MGA, the column labeled ‘Imp. Cuts’ gives the average number of Gomory fractional cuts that were strengthened using our procedure and then added to the formulation. For the row corresponding to IYA, this entry corresponds to the number of lifted cover inequalities added to the problem. The column labeled ‘% Gap’ represents the average percentage gap between the value of the best solution found and the actual optimal value of the IP as obtained by CPLEX 10.1.

Observe that MGA was able to obtain optimal solutions and prove their optimality for most problems while Glover’s algorithm rarely even reached an optimal solution as the size of the problem increased. We note that except for 2 instances with 10 rows and 25 columns, all the instances were solved using relatively small number of cuts using MGA in the case of normalized reduced cost. Comparing this with the computational results of Letchford and Lodi (2002), we observe that with respect to proving optimality, MGA outperforms IYA, while with respect to reaching an optimal solution, MGA outperforms IYA except for instances with 10 rows and 25 columns. Considering all instances, MGA with normalized reduced costs reached and proved optimality in a total of 48 instances, while IYA reached optimal solution in 48 instances and proved optimality in 35 instances. Moreover the total number of cuts used by MGA and IYA are 1698 and 9236 respectively. We note here that to the best of our knowledge, the results obtained by IYA (Letchford and Lodi (2002)) are the best computational results for unstructured integer programs using primal cutting plane algorithms. This result is particularly striking as IYA uses strong cuts such as lifted cover cuts and also uses primal heuristics while MGA uses only improved Gomory fractional cuts.

An interesting statistic is that out of a total of 1698 cuts added in all of the instances (for the case of normalized reduced cost), only 64 cuts could not be improved. Thus, approximately 97% of the cuts were improved using the algorithm of Section 2.

Using Tables 2 and 3, we can analyze the strength of various lifting sequences. There is little difference between the use of normalized reduced cost and reduced costs to determine the lifting sequence. This suggests that the algorithm performs similarly for similar lifting orders. MGA with random order reached an optimal solution for 45 instances and proved optimality for 41 instances. This clearly shows that although the lifting method based on multiple constraints generates strong coefficients for the cuts, the lifting order is also significant. Note also that MGA based on a random lifting order outperforms IYA for the following cases: 5 rows and 15, 20, 25 columns. Finally MGA with a reverse lifting sequence
Table 2: Performance on 0 – 1 instances with 5 rows

<table>
<thead>
<tr>
<th>Columns</th>
<th>Type</th>
<th>Reached</th>
<th>Proved</th>
<th>Imp. Cuts</th>
<th>Gom. Cuts</th>
<th>% Gap</th>
</tr>
</thead>
<tbody>
<tr>
<td>5</td>
<td>Glover’s Algorithm</td>
<td>5</td>
<td>5</td>
<td>-</td>
<td>17.2</td>
<td>0.00</td>
</tr>
<tr>
<td></td>
<td>MGA (Normalized Reduced Cost)</td>
<td>5</td>
<td>5</td>
<td>1.6</td>
<td>0.2</td>
<td>0.00</td>
</tr>
<tr>
<td></td>
<td>Improved Young’s Algorithm</td>
<td>5</td>
<td>5</td>
<td>4.4</td>
<td>1.4</td>
<td>0.00</td>
</tr>
<tr>
<td></td>
<td>MGA (Reduced Cost)</td>
<td>5</td>
<td>5</td>
<td>1.4</td>
<td>0.2</td>
<td>0.00</td>
</tr>
<tr>
<td></td>
<td>MGA (Random)</td>
<td>5</td>
<td>5</td>
<td>2.2</td>
<td>0.2</td>
<td>0.00</td>
</tr>
<tr>
<td></td>
<td>MGA (Reverse)</td>
<td>5</td>
<td>5</td>
<td>2.2</td>
<td>0.2</td>
<td>0.00</td>
</tr>
<tr>
<td>10</td>
<td>Glover’s Algorithm</td>
<td>4</td>
<td>3</td>
<td>-</td>
<td>301.8</td>
<td>1.20</td>
</tr>
<tr>
<td></td>
<td>MGA (Normalized Reduced Cost)</td>
<td>5</td>
<td>5</td>
<td>6.4</td>
<td>0.0</td>
<td>0.00</td>
</tr>
<tr>
<td></td>
<td>Improved Young’s Algorithm</td>
<td>5</td>
<td>5</td>
<td>8.6</td>
<td>5</td>
<td>0.00</td>
</tr>
<tr>
<td></td>
<td>MGA (Reduced Cost)</td>
<td>5</td>
<td>5</td>
<td>6.4</td>
<td>0.0</td>
<td>0.00</td>
</tr>
<tr>
<td></td>
<td>MGA (Random)</td>
<td>5</td>
<td>5</td>
<td>6.6</td>
<td>0.0</td>
<td>0.00</td>
</tr>
<tr>
<td></td>
<td>MGA (Reverse)</td>
<td>5</td>
<td>5</td>
<td>7.0</td>
<td>0.0</td>
<td>0.00</td>
</tr>
<tr>
<td>15</td>
<td>Glover’s Algorithm</td>
<td>2</td>
<td>1</td>
<td>-</td>
<td>44261.8</td>
<td>3.97</td>
</tr>
<tr>
<td></td>
<td>MGA (Normalized Reduced Cost)</td>
<td>5</td>
<td>5</td>
<td>11.2</td>
<td>1.2</td>
<td>0.00</td>
</tr>
<tr>
<td></td>
<td>Improved Young’s Algorithm</td>
<td>5</td>
<td>4</td>
<td>16.6</td>
<td>95.6</td>
<td>0.00</td>
</tr>
<tr>
<td></td>
<td>MGA (Reduced Cost)</td>
<td>5</td>
<td>5</td>
<td>10.8</td>
<td>1.2</td>
<td>0.00</td>
</tr>
<tr>
<td></td>
<td>MGA (Random)</td>
<td>5</td>
<td>5</td>
<td>11.6</td>
<td>1.2</td>
<td>0.00</td>
</tr>
<tr>
<td></td>
<td>MGA (Reverse)</td>
<td>5</td>
<td>5</td>
<td>45.8</td>
<td>1.8</td>
<td>0.00</td>
</tr>
<tr>
<td>20</td>
<td>Glover’s Algorithm</td>
<td>2</td>
<td>0</td>
<td>-</td>
<td>622</td>
<td>4.13</td>
</tr>
<tr>
<td></td>
<td>MGA (Normalized Reduced Cost)</td>
<td>5</td>
<td>5</td>
<td>32</td>
<td>2.2</td>
<td>0.00</td>
</tr>
<tr>
<td></td>
<td>Improved Young’s Algorithm</td>
<td>5</td>
<td>2</td>
<td>34.2</td>
<td>388</td>
<td>0.00</td>
</tr>
<tr>
<td></td>
<td>MGA (Reduced Cost)</td>
<td>5</td>
<td>5</td>
<td>27.2</td>
<td>2</td>
<td>0.00</td>
</tr>
<tr>
<td></td>
<td>MGA (Random)</td>
<td>5</td>
<td>5</td>
<td>75.2</td>
<td>0.8</td>
<td>0.00</td>
</tr>
<tr>
<td></td>
<td>MGA (Reverse)</td>
<td>5</td>
<td>1</td>
<td>415.2</td>
<td>2.2</td>
<td>0.00</td>
</tr>
<tr>
<td>25</td>
<td>Glover’s Algorithm</td>
<td>0</td>
<td>0</td>
<td>-</td>
<td>64135.4</td>
<td>5.00</td>
</tr>
<tr>
<td></td>
<td>MGA (Normalized Reduced Cost)</td>
<td>5</td>
<td>5</td>
<td>35.2</td>
<td>1</td>
<td>0.00</td>
</tr>
<tr>
<td></td>
<td>Improved Young’s Algorithm</td>
<td>4</td>
<td>2</td>
<td>26.6</td>
<td>439.6</td>
<td>0.57</td>
</tr>
<tr>
<td></td>
<td>MGA (Reduced Cost)</td>
<td>5</td>
<td>5</td>
<td>32.4</td>
<td>1.2</td>
<td>0.00</td>
</tr>
<tr>
<td></td>
<td>MGA (Random)</td>
<td>5</td>
<td>5</td>
<td>192.8</td>
<td>1.8</td>
<td>0.00</td>
</tr>
<tr>
<td></td>
<td>MGA (Reverse)</td>
<td>5</td>
<td>4</td>
<td>217.8</td>
<td>2.6</td>
<td>0.00</td>
</tr>
</tbody>
</table>
Table 3: Performance on 0 − 1 instances with 10 rows

<table>
<thead>
<tr>
<th>Columns</th>
<th>Type</th>
<th>Reached</th>
<th>Proved</th>
<th>Imp. Cuts</th>
<th>Gom. Cuts</th>
<th>% Gap</th>
</tr>
</thead>
<tbody>
<tr>
<td>5</td>
<td>Glover’s Algorithm</td>
<td>5</td>
<td>5</td>
<td>-</td>
<td>22.8</td>
<td>0.00</td>
</tr>
<tr>
<td></td>
<td>MGA (Normalized Reduced Cost)</td>
<td>5</td>
<td>5</td>
<td>1.4</td>
<td>2.4</td>
<td>0.00</td>
</tr>
<tr>
<td></td>
<td>Improved Young’s Algorithm</td>
<td>5</td>
<td>5</td>
<td>5.0</td>
<td>0.0</td>
<td>0.00</td>
</tr>
<tr>
<td></td>
<td>MGA (Reduced Cost)</td>
<td>5</td>
<td>5</td>
<td>1.4</td>
<td>2.4</td>
<td>0.00</td>
</tr>
<tr>
<td></td>
<td>MGA (Random)</td>
<td>5</td>
<td>5</td>
<td>1.4</td>
<td>2.4</td>
<td>0.00</td>
</tr>
<tr>
<td></td>
<td>MGA (Reverse)</td>
<td>5</td>
<td>4</td>
<td>1.6</td>
<td>2.4</td>
<td>0.00</td>
</tr>
<tr>
<td>10</td>
<td>Glover’s Algorithm</td>
<td>5</td>
<td>4</td>
<td>-</td>
<td>97.4</td>
<td>0.00</td>
</tr>
<tr>
<td></td>
<td>MGA (Normalized Reduced Cost)</td>
<td>5</td>
<td>5</td>
<td>5.4</td>
<td>1.4</td>
<td>0.00</td>
</tr>
<tr>
<td></td>
<td>Improved Young’s Algorithm</td>
<td>5</td>
<td>5</td>
<td>9.0</td>
<td>1.2</td>
<td>0.00</td>
</tr>
<tr>
<td></td>
<td>MGA (Reduced Cost)</td>
<td>5</td>
<td>5</td>
<td>5.4</td>
<td>1.4</td>
<td>0.00</td>
</tr>
<tr>
<td></td>
<td>MGA (Random)</td>
<td>5</td>
<td>5</td>
<td>9</td>
<td>1.6</td>
<td>0.00</td>
</tr>
<tr>
<td></td>
<td>MGA (Reverse)</td>
<td>5</td>
<td>4</td>
<td>22.4</td>
<td>1.4</td>
<td>0.00</td>
</tr>
<tr>
<td>15</td>
<td>Glover’s Algorithm</td>
<td>2</td>
<td>0</td>
<td>-</td>
<td>143306.6</td>
<td>7.26</td>
</tr>
<tr>
<td></td>
<td>MGA (Normalized Reduced Cost)</td>
<td>5</td>
<td>5</td>
<td>22.2</td>
<td>0.6</td>
<td>0.00</td>
</tr>
<tr>
<td></td>
<td>Improved Young’s Algorithm</td>
<td>5</td>
<td>4</td>
<td>12.4</td>
<td>81.2</td>
<td>0.00</td>
</tr>
<tr>
<td></td>
<td>MGA (Reduced Cost)</td>
<td>5</td>
<td>5</td>
<td>28.4</td>
<td>0.8</td>
<td>0.00</td>
</tr>
<tr>
<td></td>
<td>MGA (Random)</td>
<td>5</td>
<td>4</td>
<td>169.6</td>
<td>1.6</td>
<td>0.00</td>
</tr>
<tr>
<td></td>
<td>MGA (Reverse)</td>
<td>4</td>
<td>3</td>
<td>218.2</td>
<td>3</td>
<td>0.93</td>
</tr>
<tr>
<td>20</td>
<td>Glover’s Algorithm</td>
<td>0</td>
<td>0</td>
<td>-</td>
<td>13716.4</td>
<td>14.20</td>
</tr>
<tr>
<td></td>
<td>MGA (Normalized Reduced Cost)</td>
<td>5</td>
<td>5</td>
<td>35.8</td>
<td>1.4</td>
<td>0.00</td>
</tr>
<tr>
<td></td>
<td>Improved Young’s Algorithm</td>
<td>5</td>
<td>2</td>
<td>30.6</td>
<td>312.6</td>
<td>0.00</td>
</tr>
<tr>
<td></td>
<td>MGA (Reduced Cost)</td>
<td>5</td>
<td>5</td>
<td>77.0</td>
<td>1.0</td>
<td>0.00</td>
</tr>
<tr>
<td></td>
<td>MGA (Random)</td>
<td>4</td>
<td>2</td>
<td>358.8</td>
<td>1.6</td>
<td>0.31</td>
</tr>
<tr>
<td></td>
<td>MGA (Reverse)</td>
<td>3</td>
<td>0</td>
<td>422.8</td>
<td>2.2</td>
<td>0.60</td>
</tr>
<tr>
<td>25</td>
<td>Glover’s Algorithm</td>
<td>1</td>
<td>0</td>
<td>-</td>
<td>5661.0</td>
<td>6.60</td>
</tr>
<tr>
<td></td>
<td>MGA (Normalized Reduced Cost)</td>
<td>3</td>
<td>3</td>
<td>175.6</td>
<td>2.4</td>
<td>0.88</td>
</tr>
<tr>
<td></td>
<td>Improved Young’s Algorithm</td>
<td>4</td>
<td>1</td>
<td>40.2</td>
<td>335.0</td>
<td>0.22</td>
</tr>
<tr>
<td></td>
<td>MGA(Reduced Cost)</td>
<td>3</td>
<td>3</td>
<td>245.0</td>
<td>1.4</td>
<td>1.76</td>
</tr>
<tr>
<td></td>
<td>MGA(Random)</td>
<td>2</td>
<td>0</td>
<td>465.4</td>
<td>1.8</td>
<td>3.52</td>
</tr>
<tr>
<td></td>
<td>MGA (Reverse)</td>
<td>1</td>
<td>0</td>
<td>456.0</td>
<td>1.6</td>
<td>3.72</td>
</tr>
</tbody>
</table>
Table 4: Running time of different implementations of MGA and Glover’s Algorithm

<table>
<thead>
<tr>
<th>Type</th>
<th>Rows</th>
<th>Columns</th>
<th>Lifting Time</th>
<th>Total Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>Glover’s Algorithm</td>
<td>5</td>
<td>15</td>
<td>-</td>
<td>13.60</td>
</tr>
<tr>
<td>MGA (With Primal-Dual pivots)</td>
<td>5</td>
<td>15</td>
<td>0.07</td>
<td>0.07</td>
</tr>
<tr>
<td>MGA (Without Primal-Dual pivots)</td>
<td>5</td>
<td>15</td>
<td>0.10</td>
<td>0.11</td>
</tr>
<tr>
<td>Glover’s Algorithm</td>
<td>5</td>
<td>20</td>
<td>-</td>
<td>0.44</td>
</tr>
<tr>
<td>MGA (With Primal-Dual pivots)</td>
<td>5</td>
<td>20</td>
<td>0.26</td>
<td>0.27</td>
</tr>
<tr>
<td>MGA (Without Primal-Dual pivots)</td>
<td>5</td>
<td>20</td>
<td>0.52</td>
<td>0.53</td>
</tr>
<tr>
<td>Glover’s Algorithm</td>
<td>5</td>
<td>25</td>
<td>-</td>
<td>120.54</td>
</tr>
<tr>
<td>MGA (With Primal-Dual pivots)</td>
<td>5</td>
<td>25</td>
<td>0.42</td>
<td>0.44</td>
</tr>
<tr>
<td>MGA (Without Primal-Dual pivots)</td>
<td>5</td>
<td>25</td>
<td>0.94</td>
<td>0.95</td>
</tr>
<tr>
<td>Glover’s Algorithm</td>
<td>10</td>
<td>15</td>
<td>-</td>
<td>120.96</td>
</tr>
<tr>
<td>MGA (With Primal-Dual pivots)</td>
<td>10</td>
<td>15</td>
<td>0.07</td>
<td>0.07</td>
</tr>
<tr>
<td>MGA (Without Primal-Dual pivots)</td>
<td>10</td>
<td>15</td>
<td>0.10</td>
<td>0.11</td>
</tr>
<tr>
<td>Glover’s Algorithm</td>
<td>10</td>
<td>20</td>
<td>-</td>
<td>5.58</td>
</tr>
<tr>
<td>MGA (With Primal-Dual pivots)</td>
<td>10</td>
<td>20</td>
<td>0.38</td>
<td>0.40</td>
</tr>
<tr>
<td>MGA (Without Primal-Dual pivots)</td>
<td>10</td>
<td>20</td>
<td>0.66</td>
<td>0.68</td>
</tr>
<tr>
<td>Glover’s Algorithm</td>
<td>10</td>
<td>25</td>
<td>-</td>
<td>3.19</td>
</tr>
<tr>
<td>MGA (With Primal-Dual pivots)</td>
<td>10</td>
<td>25</td>
<td>5.11</td>
<td>5.31</td>
</tr>
<tr>
<td>MGA (Without Primal-Dual pivots)</td>
<td>10</td>
<td>25</td>
<td>10.42</td>
<td>10.60</td>
</tr>
</tbody>
</table>

does not perform well. An optimal solution was obtained for 43 instances while optimality was proven for only 31 instances. However, we observe that the performance of MGA based on the reverse order is strictly better than the performance of Glover’s algorithm indicating that our lifting procedure is strong.

We now analyze the running time of MGA with respect to two of its main characteristics: (1) the division of the running time between the actual algorithm and the lifting procedure, and (2) the effect of adopting the sequence of primal-dual simplex pivots presented in Section 2.3. Table 4 presents the time required by Glover’s algorithm and MGA with normalized reduced cost for the set of larger instances (i.e. with more that 10 columns). All times are in CPU seconds. We observe in Table 4 that the running time for Glover’s algorithm appears inconsistent. The reason that the running time decreases with an increase in the problem size seems to be that, the stopping criterion which specifies that entries in the simplex tableaux cannot exceed 1,000,000 is reached sooner. We also observe that MGA spends approximately 96% of the time in the lifting procedure. We finally observe that when MGA was implemented with the sequence of primal-dual pivots of Section 2.3 there was an improvement in the running time of MGA of approximately 50%.

As mentioned before, the slack variables in MGA are general integer variables. It is
Table 5: Comparison of Glover’s algorithm and MGA for general integer instances

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>5</td>
<td>5</td>
<td>5</td>
<td>5</td>
<td>1.8</td>
<td>1.2</td>
<td>0.00</td>
<td>5</td>
<td>5</td>
<td>4.2</td>
<td>0.00</td>
</tr>
<tr>
<td>5</td>
<td>10</td>
<td>5</td>
<td>5</td>
<td>5.6</td>
<td>2.8</td>
<td>0.00</td>
<td>5</td>
<td>5</td>
<td>24</td>
<td>0.00</td>
</tr>
<tr>
<td>5</td>
<td>15</td>
<td>5</td>
<td>5</td>
<td>46.2</td>
<td>5.6</td>
<td>0.00</td>
<td>3</td>
<td>2</td>
<td>794.2</td>
<td>1.39</td>
</tr>
<tr>
<td>5</td>
<td>20</td>
<td>5</td>
<td>5</td>
<td>68</td>
<td>11.4</td>
<td>0.00</td>
<td>3</td>
<td>3</td>
<td>3927</td>
<td>5.13</td>
</tr>
<tr>
<td>5</td>
<td>25</td>
<td>5</td>
<td>5</td>
<td>96</td>
<td>9.2</td>
<td>0.00</td>
<td>1</td>
<td>0</td>
<td>3479.8</td>
<td>9.11</td>
</tr>
<tr>
<td>10</td>
<td>5</td>
<td>5</td>
<td>5</td>
<td>2</td>
<td>2.2</td>
<td>0.00</td>
<td>5</td>
<td>5</td>
<td>10</td>
<td>0.00</td>
</tr>
<tr>
<td>10</td>
<td>10</td>
<td>5</td>
<td>5</td>
<td>7.6</td>
<td>2.2</td>
<td>0.00</td>
<td>3</td>
<td>3</td>
<td>317.6</td>
<td>4.10</td>
</tr>
<tr>
<td>10</td>
<td>15</td>
<td>5</td>
<td>5</td>
<td>66.6</td>
<td>7.4</td>
<td>0.00</td>
<td>1</td>
<td>0</td>
<td>1753.6</td>
<td>10.49</td>
</tr>
<tr>
<td>10</td>
<td>20</td>
<td>4</td>
<td>3</td>
<td>205</td>
<td>15.8</td>
<td>8.82</td>
<td>2</td>
<td>1</td>
<td>2723.6</td>
<td>16.28</td>
</tr>
<tr>
<td>10</td>
<td>25</td>
<td>3</td>
<td>3</td>
<td>312</td>
<td>15.4</td>
<td>15.60</td>
<td>1</td>
<td>1</td>
<td>11336.4</td>
<td>28.00</td>
</tr>
</tbody>
</table>

Therefore interesting to determine how frequently general integer variables are lifted in a 0-1 program. We find that in all the experiments on 0-1 instances approximately 62% of the variables lifted are not binary. This is not surprising since with the addition of each new cut the number of general integer slack variables increases.

We now present the results we obtained with MGA on the second set of instances that have general integer variable in their formulations. We compare in Table 5 the results obtained by Glover’s algorithm and MGA. We observe that MGA performs significantly better than Glover’s algorithm, thus illustrating the advantage of having a lifting method that works for general integer variables.

The results for MIPLIB instances p0040 and p0033 are presented in Tables 6 and 7 respectively. Because these instances were larger we limited the total number of cuts added to 50,000. For p0040, Glover’s algorithm reaches the optimal solution 6 times and proves optimality only 2 times. On the other hand, MGA always reaches and proves optimality. For p0033, a problem known to be more difficult than p0040, Glover’s algorithm never reaches the optimal solution. MGA however proves optimality 8 times and reaches the optimal solution 13 times out of 20 different starting solutions. Clearly, MGA is much stronger and has better numerical properties than a straightforward implementation of Glover’s algorithm. Also we observe that as the starting solution is chosen closer to the optimal solution, the performance of MGA improves.
### Table 6: Results of Modified Glover’s Algorithm on p0040

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>64062</td>
<td>62027</td>
<td>Y</td>
<td>Y</td>
<td>2</td>
<td>0</td>
<td>62051</td>
<td>N</td>
<td>N</td>
<td>341</td>
</tr>
<tr>
<td>64331</td>
<td>62027</td>
<td>Y</td>
<td>Y</td>
<td>2</td>
<td>0</td>
<td>62228</td>
<td>N</td>
<td>N</td>
<td>198</td>
</tr>
<tr>
<td>65056</td>
<td>62027</td>
<td>Y</td>
<td>Y</td>
<td>2</td>
<td>0</td>
<td>62113</td>
<td>N</td>
<td>N</td>
<td>502</td>
</tr>
<tr>
<td>65271</td>
<td>62027</td>
<td>Y</td>
<td>Y</td>
<td>2</td>
<td>0</td>
<td>62051</td>
<td>N</td>
<td>N</td>
<td>261</td>
</tr>
<tr>
<td>65298</td>
<td>62027</td>
<td>Y</td>
<td>Y</td>
<td>2</td>
<td>0</td>
<td>62187</td>
<td>N</td>
<td>N</td>
<td>269</td>
</tr>
<tr>
<td>65578</td>
<td>62027</td>
<td>Y</td>
<td>Y</td>
<td>2</td>
<td>0</td>
<td>62076</td>
<td>N</td>
<td>N</td>
<td>533</td>
</tr>
<tr>
<td>65664</td>
<td>62027</td>
<td>Y</td>
<td>Y</td>
<td>2</td>
<td>0</td>
<td>62027</td>
<td>Y</td>
<td>N</td>
<td>271</td>
</tr>
<tr>
<td>65671</td>
<td>62027</td>
<td>Y</td>
<td>Y</td>
<td>2</td>
<td>0</td>
<td>62170</td>
<td>N</td>
<td>N</td>
<td>94</td>
</tr>
<tr>
<td>65828</td>
<td>62027</td>
<td>Y</td>
<td>Y</td>
<td>2</td>
<td>0</td>
<td>62076</td>
<td>N</td>
<td>N</td>
<td>152</td>
</tr>
<tr>
<td>66011</td>
<td>62027</td>
<td>Y</td>
<td>Y</td>
<td>2</td>
<td>0</td>
<td>62101</td>
<td>N</td>
<td>N</td>
<td>50000</td>
</tr>
<tr>
<td>66093</td>
<td>62027</td>
<td>Y</td>
<td>Y</td>
<td>2</td>
<td>0</td>
<td>62076</td>
<td>N</td>
<td>N</td>
<td>230</td>
</tr>
<tr>
<td>66184</td>
<td>62027</td>
<td>Y</td>
<td>Y</td>
<td>2</td>
<td>0</td>
<td>62027</td>
<td>Y</td>
<td>Y</td>
<td>11</td>
</tr>
<tr>
<td>66476</td>
<td>62027</td>
<td>Y</td>
<td>Y</td>
<td>2</td>
<td>0</td>
<td>62116</td>
<td>N</td>
<td>N</td>
<td>295</td>
</tr>
<tr>
<td>66517</td>
<td>62027</td>
<td>Y</td>
<td>Y</td>
<td>2</td>
<td>0</td>
<td>62027</td>
<td>Y</td>
<td>Y</td>
<td>15</td>
</tr>
<tr>
<td>66579</td>
<td>62027</td>
<td>Y</td>
<td>Y</td>
<td>2</td>
<td>0</td>
<td>62027</td>
<td>Y</td>
<td>Y</td>
<td>14</td>
</tr>
<tr>
<td>66612</td>
<td>62027</td>
<td>Y</td>
<td>Y</td>
<td>2</td>
<td>0</td>
<td>62142</td>
<td>N</td>
<td>N</td>
<td>72</td>
</tr>
<tr>
<td>66675</td>
<td>62027</td>
<td>Y</td>
<td>Y</td>
<td>2</td>
<td>0</td>
<td>62094</td>
<td>N</td>
<td>N</td>
<td>740</td>
</tr>
<tr>
<td>67090</td>
<td>62027</td>
<td>Y</td>
<td>Y</td>
<td>2</td>
<td>0</td>
<td>62076</td>
<td>N</td>
<td>N</td>
<td>371</td>
</tr>
<tr>
<td>67311</td>
<td>62027</td>
<td>Y</td>
<td>Y</td>
<td>2</td>
<td>0</td>
<td>62027</td>
<td>Y</td>
<td>N</td>
<td>165</td>
</tr>
<tr>
<td>67312</td>
<td>62027</td>
<td>Y</td>
<td>Y</td>
<td>2</td>
<td>0</td>
<td>62076</td>
<td>N</td>
<td>N</td>
<td>200</td>
</tr>
<tr>
<td>65980</td>
<td>62027</td>
<td>100%</td>
<td>100%</td>
<td>2</td>
<td>0</td>
<td>62088</td>
<td>25%</td>
<td>15%</td>
<td>2736.7</td>
</tr>
</tbody>
</table>

### Table 7: Results of Modified Glover’s Algorithm on p0033

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>3188</td>
<td>3089</td>
<td>Y</td>
<td>Y</td>
<td>71</td>
<td>0</td>
<td>3188</td>
<td>N</td>
<td>N</td>
<td>50000</td>
</tr>
<tr>
<td>3278</td>
<td>3089</td>
<td>Y</td>
<td>Y</td>
<td>38</td>
<td>0</td>
<td>3278</td>
<td>N</td>
<td>N</td>
<td>1150</td>
</tr>
<tr>
<td>3298</td>
<td>3089</td>
<td>Y</td>
<td>N</td>
<td>87</td>
<td>2</td>
<td>3298</td>
<td>N</td>
<td>N</td>
<td>252</td>
</tr>
<tr>
<td>3302</td>
<td>3302</td>
<td>N</td>
<td>N</td>
<td>102</td>
<td>1</td>
<td>3302</td>
<td>N</td>
<td>N</td>
<td>959</td>
</tr>
<tr>
<td>3438</td>
<td>3089</td>
<td>Y</td>
<td>Y</td>
<td>83</td>
<td>1</td>
<td>3188</td>
<td>N</td>
<td>N</td>
<td>531</td>
</tr>
<tr>
<td>3447</td>
<td>3089</td>
<td>Y</td>
<td>Y</td>
<td>35</td>
<td>2</td>
<td>3188</td>
<td>N</td>
<td>N</td>
<td>107</td>
</tr>
<tr>
<td>3537</td>
<td>3089</td>
<td>Y</td>
<td>Y</td>
<td>51</td>
<td>2</td>
<td>3537</td>
<td>N</td>
<td>N</td>
<td>50000</td>
</tr>
<tr>
<td>3550</td>
<td>3095</td>
<td>N</td>
<td>N</td>
<td>79</td>
<td>1</td>
<td>3550</td>
<td>N</td>
<td>N</td>
<td>617</td>
</tr>
<tr>
<td>3593</td>
<td>3089</td>
<td>Y</td>
<td>Y</td>
<td>57</td>
<td>1</td>
<td>3593</td>
<td>N</td>
<td>N</td>
<td>505</td>
</tr>
<tr>
<td>3602</td>
<td>3089</td>
<td>Y</td>
<td>Y</td>
<td>105</td>
<td>3</td>
<td>3602</td>
<td>N</td>
<td>N</td>
<td>302</td>
</tr>
<tr>
<td>3660</td>
<td>3095</td>
<td>N</td>
<td>N</td>
<td>77</td>
<td>1</td>
<td>3660</td>
<td>N</td>
<td>N</td>
<td>432</td>
</tr>
<tr>
<td>3671</td>
<td>3089</td>
<td>Y</td>
<td>N</td>
<td>92</td>
<td>5</td>
<td>3412</td>
<td>N</td>
<td>N</td>
<td>818</td>
</tr>
<tr>
<td>3672</td>
<td>3089</td>
<td>Y</td>
<td>N</td>
<td>96</td>
<td>1</td>
<td>3338</td>
<td>N</td>
<td>N</td>
<td>222</td>
</tr>
<tr>
<td>3707</td>
<td>3095</td>
<td>N</td>
<td>N</td>
<td>85</td>
<td>1</td>
<td>3707</td>
<td>N</td>
<td>N</td>
<td>129</td>
</tr>
<tr>
<td>3712</td>
<td>3089</td>
<td>Y</td>
<td>N</td>
<td>135</td>
<td>1</td>
<td>3712</td>
<td>N</td>
<td>N</td>
<td>86</td>
</tr>
<tr>
<td>3716</td>
<td>3302</td>
<td>N</td>
<td>N</td>
<td>88</td>
<td>3</td>
<td>3716</td>
<td>N</td>
<td>N</td>
<td>209</td>
</tr>
<tr>
<td>3807</td>
<td>3089</td>
<td>Y</td>
<td>N</td>
<td>78</td>
<td>2</td>
<td>3807</td>
<td>N</td>
<td>N</td>
<td>1163</td>
</tr>
<tr>
<td>3826</td>
<td>3089</td>
<td>Y</td>
<td>Y</td>
<td>94</td>
<td>2</td>
<td>3826</td>
<td>N</td>
<td>N</td>
<td>50000</td>
</tr>
<tr>
<td>3910</td>
<td>3095</td>
<td>N</td>
<td>N</td>
<td>86</td>
<td>2</td>
<td>3910</td>
<td>N</td>
<td>N</td>
<td>431</td>
</tr>
<tr>
<td>4059</td>
<td>3550</td>
<td>N</td>
<td>N</td>
<td>123</td>
<td>2</td>
<td>4059</td>
<td>N</td>
<td>N</td>
<td>65</td>
</tr>
<tr>
<td>3598</td>
<td>3134</td>
<td>65%</td>
<td>40%</td>
<td>83.1</td>
<td>1.65</td>
<td>3543</td>
<td>0%</td>
<td>0%</td>
<td>7898.9</td>
</tr>
</tbody>
</table>

25
5. Conclusion

In this paper, we present a general lifting procedure for unstructured integer programs that considers several constraints of the problem formulation. This procedure uses linear programming duality concepts to derive approximate lifting coefficients using information from multiple constraints in an integer program and can be used to significantly strengthen cutting planes in IP. The time required to improve a single cut coefficient is basically that of solving an LP. Furthermore the computational effort to strengthen all the coefficients in a cut can be significantly reduced using appropriate sequences of primal and dual simplex pivots. We proved that the use of strengthened Gomory’s fractional cuts within the framework of Glover’s algorithm yields finite convergence.

We then empirically tested our cut improvement procedure within the framework of Glover’s algorithm. The results we obtained show the benefits of the proposed strengthening approach. An interesting statistic that emerged from the experiments is that 97% of cuts could be improved using the procedure. Note also that unlike the GMIC where improvements of cut coefficients are by fractional amounts, the improvements of coefficients in our procedure are by integer amounts. Because the procedure we proposed appears to converge faster than both Glover’s algorithm and the algorithm developed by Letchford and Lodi, which uses strong cuts like covers and also uses heuristics, we believe that our computational experiments indicate that considering information from multiple constraints is very helpful in the generation of stronger cuts. Furthermore we believe it indicates that the lifting procedure we proposed is a simple and effective method to use multiple constraint information especially if no prior information is available about the structure of the problem.

Although our cut improvement method was used here specifically to improve Gomory’s fractional cut for pure integer programs, it is very general and can be used in many different contexts. First, it can be used to improve inequalities other than Gomory’s fractional cut including cover cuts and clique inequalities. Second, it can be used in the context of preprocessing to improve the formulation of general integer programs. We believe that this is a significant practical advantage of this procedure as most preprocessing tools currently used in commercial software are designed only for binary variables. Third, the approach can be adapted to improve coefficients of integer variables for mixed-integer programs. Fourth, this method can be adapted to strengthen cuts in a dual cutting plane setting. Finally, the
primal algorithm presented in this paper may be useful for improving existing heuristics for finding good feasible solutions for IPs within branch-and-bound trees.

The work presented in this paper opens the way for four directions of future research. First, a larger computational study is needed to verify the benefits of the approach especially with dual cutting planes. Second, a deeper investigation of primal cutting plane algorithms incorporating the suggestions by Letchford and Lodi (2002) with lifting is necessary. Third, sophisticated heuristics to decide which cuts to strengthen and which variables to improve should be designed. Finally, as discussed in Section 2.2, the lifting procedure should be expanded to yield useful coefficients for continuous variables.

**Appendix**

**Proposition 4** Assume that $x_m$ is fixed to a value $k$ for which there exists at least one point of the linear programming relaxation of (1) with $x_m = k$ (i.e., $0 \in S^+$). If $\Gamma^+(0) \leq 0$ and

1. $1 \in S^+$ then $\bar{\beta}^u = \Gamma^+(1)$.
2. $1 \in S^-$ then $\bar{\beta}^l = \Gamma^-(1)$.

**Proof** We first prove that $\bar{\beta}^u = \Gamma^+(1)$. Observe that since we assumed that there exists a feasible point to the LP relaxation of (1) with $x_m = k$ then $0 \in S^+$. Furthermore, because we established in Proposition 2 that $\Gamma^+$ is convex, we obtain that

$$\frac{1}{j} \Gamma^+(j) + \left(1 - \frac{1}{j}\right) \Gamma^+(0) \geq \Gamma^+(1) \forall j \in S^+, \text{ for } j \geq 1.$$ 

Since $\Gamma^+(0) \leq 0$ by assumption, we obtain that $\frac{\Gamma^+(j)}{j} \geq \Gamma^+(1)$ for $j \geq 1$. Observe now that $\bar{\beta}^u = \min \left\{ \frac{\Gamma^+(j)}{j} \mid j \in S^+, j \geq 1, j \in \mathbb{Z} \right\}$, hence proving the result. \hfill \Box

**Proposition 12** MGA converges in a finite number of steps.

**Proof** We verify conditions (1), (2) and (3) of Proposition 11. For (1), it follows from Garfinkel and Nemhauser (1972) that the two properties of the special row are maintained in successive iterations for any valid cut that has integer coefficients for all variables and has coefficient of 1 for the entering variable.

For (2), we observe first that for $j \neq k$, $a_j^1 = a_j - h_j a_k$ where $h_j$ is the coefficient of $x_j$ in the improved Gomory fractional cut. (Note that $h_0$ is the right-hand-side of the improved Gomory cut and is equal to $\left\lfloor \frac{a_0}{a_k} \right\rfloor$). Therefore if $a_{+k} > a_{+0}$, we obtain $a_{+0}^1 = a_{+0}$.
since \( h_0 = \left\lfloor \frac{a_{00}}{a_{0k}} \right\rfloor = 0 \). Since \( a^1_{s0} = a_{s0} \geq 0 \) and \( a^1_{sk} = a_{sk} - h_{k1}a_{sk} \), if \( a^1_{sk} > a^1_{s0} \), we obtain \( a_{sk} - h_{k1}a_{sk} \geq 0 \). Hence,

\[
0 \leq \frac{a^1_{sk}}{a_{sk}} = a^1_{sk} - h_{k1} \leq \frac{a^1_{sk}}{a_{sk}} - \left\lfloor \frac{a^1_{sk}}{a_{sk}} \right\rfloor < 1.
\]

The integrality of \( a^1_{sk} \) and \( a_{sk} \) then implies that \( a^1_{sk} \leq a_{sk} - 1 \) (It also implies that \( h_{k1} = \left\lfloor \frac{a^1_{sk}}{a_{sk}} \right\rfloor \) in this case). Thus there exists an integer \( t \) such that \( a^1_{sk} \leq a^1_{s0} \) since \( a^1_{s0} = a_{s0} \). Thus the algorithm is finitely convergent.

For (3), again it is easy to verify that \( a^1_{q0} = a_{q0} \) and \( a^1_{qk} \leq a_{qk} - 1 \). Thus there exists \( t \) such that \( a^1_{qk} \leq a_{q0} \). Since \( k \) is the entering variable, it follows from the property of the special row that \( a^1_{sk} \geq 0 \). Therefore, \( \frac{a^1_{qk}}{a^1_{sk}} \leq a_{q0} \).

**Proposition 13** Assume that (1) has at least one feasible solution and assume that \( A \) and \( b \) are rational. Then \( \Gamma^+ \) is bounded for all \( \lambda \in S^+ \) and \( \Gamma^- \) is bounded for all \( \lambda \in S^- \).

**Proof** Let \((\bar{x}, \bar{\lambda}) \in (\mathbb{Z}_+^{m-1}, \mathbb{Z}_+)\) be a point that satisfies \( \sum_{j=1}^{m-1} A_jx_j + Am\lambda \leq b \). Clearly, \( \bar{\lambda} - k \in S^+ \). We first prove that \( \Gamma^+(\bar{\lambda} - k) \) is bounded. Assume by contradiction that \( \Gamma^+(\bar{\lambda} - k) \) is not bounded. Then there exists a rational vector \( \eta \in \mathbb{Q}^{m-1} \) that is an extreme ray of the polytope \( \sum_{j=1}^{m-1} A_jx_j \leq b' - Am(\bar{\lambda} - k) = b - A_m\bar{\lambda} \) with \(-\sum_{j=1}^{m-1} \alpha_j\eta_j < 0 \). As \( \eta \) is rational, \( \exists M \in \mathbb{Z}_+ \) such that \( \bar{x} + M\eta \in \{x \in \mathbb{Z}_+^{m-1} \mid \sum_{j=1}^{m-1} A_jx_j \leq b - A_m\bar{\lambda} \} \) and \( \sum_{j=1}^{m-1} -\alpha_j(\bar{x} + M\eta)_j < -r + A_m\bar{\lambda} \). This is the required contradiction as \( \sum_{j=1}^{m} \alpha_jx_j \leq r \) is a valid inequality for (1). Now assume by contradiction that \( \Gamma^+(\lambda) \) is unbounded for some \( \lambda \in S^+ \). This implies that the dual of the linear program defining \( \Gamma^+ \) is infeasible. This is the required contradiction, since the feasible region of the dual does not depend on \( \lambda \). Finally, since \( \Gamma^-(-\lambda) = -\Gamma^+(-\lambda) \), \( \Gamma^- \) is also bounded for all \( \lambda \in S^- \).

**Acknowledgments**

The authors would like to thank Adam Letchford and Andrea Lodi for providing them with their tests instances. They also would like to thank anonymous referees for their helpful and constructive comments about an earlier version of this paper. This research was supported by NSF Grant DMI-03-48611.
References


