INFORMS Journal on Computing
Vol. 21, No. 4, Fall 2009, pp. 641–649
ISSN 1091-9856 | EISSN 1526-5528 | 09 | 2104 | 0641

DOI 10.1287/ijoc.1090.0324
© 2009 INFORMS

Numerically Safe Gomory Mixed-Integer Cuts

William Cook
Industrial and Systems Engineering, Georgia Institute of Technology,
Atlanta, Georgia 30332, bico@isye.gatech.edu

Sanjeeb Dash, Ricardo Fukasawa
IBM T. J. Watson Research Center, Yorktown Heights, New York 10598
{sanjeebd@us.ibm.com, rfukasaw@us.ibm.com}

Marcos Goycoolea
School of Business, Universidad Adolfo Ibáñez, Peñalolen, 7941169 Santiago, Chile,
marcos.goycoolea@uai.cl

We describe a simple process for generating numerically safe cutting planes using floating-point arithmetic and the mixed-integer rounding procedure. Applying this method to the rows of the simplex tableau permits the generation of Gomory mixed-integer cuts that are guaranteed to be satisfied by all feasible solutions to a mixed-integer programming problem (MIP). We report on tests with the MIPLIB 3.0 and MIPLIB 2003 test collections as well as with MIP instances derived from the TSPLIB traveling salesman library.

Key words: integer programming; cutting plane; Gomory cuts
History: Accepted by Karen Aardal, Area Editor for Design and Analysis of Algorithms; received March 2008; revised October 2008, February 2009; accepted March 2009. Published online in *Articles in Advance* June 29, 2009.

1. Introduction

Mixed-integer programming (MIP) is a fundamental tool in operations research. An MIP problem has the form

$$\begin{aligned}&\text{Minimize} \quad c^T x\\ &\text{subject to} \quad Ax \geq d, \quad l \leq x \leq u, \; x_j \text{ integer } (j \in I),\end{aligned} \tag{1}$$

where A is an $m \times n$ matrix, d is an m-vector, c, l, u are n-vectors, x is an n-vector of variables, and $I \subseteq \{1, \ldots, n\}$. We assume all input data are rational and that some of the variable bounds can be infinite. Introductions to MIP theory and applications can be found in Nemhauser and Wolsey (1988), Wolsey (1998), and Bertsimas and Weismantel (2005).

The past decade has seen rapid improvements in the quality of software for the solution of general MIP problems. Bixby et al. (2000, 2004) report that a critical factor in this progress has been the increased use of cutting planes, or *cuts*, to improve the linear programming (LP) relaxations of MIP models. An important milestone in this context is the study of Balas et al. (1996), renewing interest in the computational application of Gomory mixed-integer cuts (Gomory 1960). Indeed, among the varieties of cutting planes in current use, Bixby et al. (2004) rank Gomory cuts as the class having the greatest overall impact on the performance of MIP solvers.

Gomory cuts are obtained via a process known as mixed-integer rounding (MIR) applied to the rows of an optimal simplex tableau (Wolsey 1998). It is important to note that both the MIR process and the linear algebra required to construct a simplex tableau row are subject to rounding errors when computed in floating-point arithmetic, the standard platform adopted in MIP software. This numerical difficulty can, in practice, lead to the generation of cutting planes that are not valid for a given MIP instance. Indeed, in a careful study, Margot (2009) documents the frequent occurrence of infeasible cuts in applications of the MIR procedure. A consequence of this numerical difficulty is that MIP algorithms can become unstable if Gomory cuts are added in multiple rounds, where cuts become a part of the LP relaxation used to produce further cuts. For this reason, commercial MIP solvers limit the repeated application of Gomory cuts, even in cases where significant progress in the objective bound of the LP relaxation is being made.

We say that a cutting-plane generation process in a specified model of arithmetic is *safe* if it is guaranteed to produce inequalities that are valid for a given MIP problem. In this paper, our notion of a valid inequality coincides with the standard notion of an inequality satisfied by all feasible solutions of an MIP when computations to evaluate the inequality are performed in infinite precision.

Neumaier and Shcherbina (2004) introduce the notion of safe cuts and use a combination of interval arithmetic and directed rounding to derive a safe version of the MIR procedure for floating-point computation. In that paper, they also show how to compute safe dual bounds and to obtain safe infeasibility certificates. However, their approach relies on the fact that all variables are bounded and that these bounds are all reasonable (for instance, not the theoretical exponential bound that can be imposed in any MIP), and uses interval arithmetic, which makes it not so straightforward to implement.

In this paper we present an alternative implementation of safe MIR cuts in floating-point arithmetic, and we demonstrate its application in generating Gomory cuts in general MIP instances. We report on computational studies with the MIPLIB 3.0 and MIPLIB 2003 test collections (Bixby et al. 1998, Achterberg et al. 2006, respectively) and with MIP instances derived from the TSPLIB traveling salesman library (Reinelt 1991). In the TSPLIB tests, our method uses the Concorde TSP solver (Applegate et al. 2006) and safe Gomory cuts to establish lower bounds on the lengths of tours.

Our computer code is based on standard IEEE floating-point arithmetic (IEEE 1985), and the ideas used in the implementation are relatively simple and easy to replicate. The source code is available as an Online Supplement (available at http://joc.pubs.informs.org/ecompanion.html).

The rest of this paper is organized as follows. In §2 we define the MIR procedure for generating a cut from a mixed-integer knapsack set. General properties of floating-point arithmetic are discussed in §3, together with simple procedures for safe row aggregation and safe substitution of slack variables. In §4 we present the safe MIR procedure and extend this to a safe procedure for the more general complemented-MIR cuts. In §5 we present the results of our computational study. Final remarks are presented in §6.

2. MIR Inequalities

The sets of rational numbers, real numbers, and integers are denoted by $\mathbb{Q}$, $\mathbb{R}$, and $\mathbb{Z}$, respectively. For $t \in \mathbb{R}$, $\lfloor t \rfloor$ denotes the greatest integer less than or equal to t, $\lceil t \rceil$ denotes the smallest integer greater than or equal to t, and $\hat{t}$ denotes $t - \lfloor t \rfloor$.

Let n be a positive integer and let N denote the set $\{1, \dots, n\}$. Consider $a \in \mathbb{Q}^n$, $b \in \mathbb{Q}$, and a partition (I, C) of N, where I and C are sets of indices of integer and continuous variables, respectively. A single-row relaxation of an MIP model can take the form

$$K = \left\{ x \in \mathbb{R}^n \colon \sum_{j \in N} a_j x_j \geq b,\ x \geq 0,\ x_j \in \mathbb{Z}\ \forall j \in I \right\}. \tag{2}$$

The defining inequality of K can, for example, be one of the original MIP constraints or, more generally, a nonnegative linear combination of the original constraints.

Let $S = \{j \in I \colon \hat{a}_j \leq \hat{b}\}$. The *MIR inequality*

$$\sum_{j \in S} (\hat{a}_j + \hat{b}\lfloor a_j \rfloor) x_j + \sum_{j \in I \setminus S} (\hat{b} + \hat{b}\lfloor a_j \rfloor) x_j + \sum_{j \in C} \max\{a_j, 0\} x_j \geq \hat{b}\lceil b \rceil \tag{3}$$

is valid for K and can therefore be considered for use as a cutting plane to improve the LP relaxation of the original MIP model (Wolsey 1998).

Defining $f(q_1, q_2) := \min\{\hat{q}_1, \hat{q}_2\} + \hat{q}_2 \lfloor q_1 \rfloor$ and $h(q) := \max\{q, 0\}$, the MIR inequality (3) can be written as

$$\sum_{i \in I} f(a_i, b) x_i + \sum_{i \in C} h(a_i) x_i \geq \hat{b}\lceil b \rceil. \tag{4}$$

This compact notation will be convenient in our discussion of safe versions of the inequality.

Let l, u be vectors with n components such that for each $j \in C$, $l_j \in \mathbb{Q}$ and $u_j \in \mathbb{Q} \cup \{+\infty\}$, and for each $j \in I$, $l_j \in \mathbb{Z}$ and $u_j \in \mathbb{Z} \cup \{+\infty\}$. Assume $0 \leq l_j \leq u_j$ for each $j \in N$ and consider the set

$$K_B = \left\{ x \in \mathbb{R}^n \colon \sum_{j \in N} a_j x_j \geq b,\ l_j \leq x_j \leq u_j\ \forall j \in N \text{ and } x_j \in \mathbb{Z}\ \forall j \in I \right\}. \tag{5}$$

The lower and upper bounds on the variables lead to a slightly more general form of the MIR inequality, called the *complemented-MIR* (c-MIR) inequalities (Marchand and Wolsey 2001). The name derives from the fact that the c-MIR inequalities are obtained by complementing variables.

Let U, L be disjoint subsets of N such that $u_j < \infty$ for every $j \in U$. For each $j \in U$, define $x_j^u = u_j - x_j$. Also, define $x_j^l = x_j - l_j$ for each $j \in L$. Observe that x_j^u and x_j^l are both bounded below by zero for $x \in K_B$ and are integer-valued when $j \in I$. Substituting x_j for $j \in U$ by $u_j - x_j^u$, and x_j for $j \in L$ by $x_j^l + l_j$, one obtains the inequality

$$\sum_{j \in U} (-a_j) x_j^u + \sum_{j \in L} a_j x_j^l + \sum_{j \notin U \cup L} a_j x_j \geq b - \sum_{j \in U} a_j u_j - \sum_{j \in L} a_j l_j, \tag{6}$$

with all variables being nonnegative and variables with index $j \in I$ being integral. Let $r = b - \sum_{j \in U} a_j u_j - \sum_{j \in L} a_j l_j$. Applying the MIR procedure to this inequality and substituting back yields the following c-MIR inequality for K_B:

$$-\sum_{U \cap I} f(-a_j, r) x_j + \sum_{I \setminus U} f(a_j, r) x_j - \sum_{U \cap C} h(-a_j) x_j + \sum_{C \setminus U} h(a_j) x_j \geq R, \tag{7}$$

where

$$R = \hat{r}\lceil r\rceil - \sum_{U\cap I} f(-a_j, r)u_j + \sum_{L\cap I} f(a_j, r)l_j - \sum_{U\cap C} h(-a_j)u_j + \sum_{L\cap C} h(a_j)l_j.$$

Marchand and Wolsey (2001) have demonstrated the effectiveness of c-MIR cuts in practical computations with MIP test instances. Further computational results on MIR and c-MIR cuts are reported in Balas and Saxena (2008), Dash et al. (2008, 2009), and Goycoolea (2006).

If the defining inequality of K_B is taken as a row of an optimal simplex tableau for the LP relaxation of an MIP instance, then the c-MIR cut is a form of the Gomory mixed-integer cut. Our computational study will focus on Gomory cuts, but the numerically safe methods will be presented in the general context of the MIR and c-MIR procedures.

3. Floating-Point Arithmetic

The validity of MIR and c-MIR inequalities, for the sets K and K_B, relies on the correctness of arithmetic calculations, which cannot be guaranteed when floating-point arithmetic is used. Moreover, if the defining inequality for K or K_B is obtained by aggregating several of the original constraints from an MIP instance, then care must be taken to ensure that the set itself is a valid relaxation. To discuss an approach for dealing with these issues, we give a brief description of the floating-point arithmetic platform.

3.1. The Model of Floating-Point Arithmetic

A floating-point number consists of a sign, exponent, and mantissa. Standard IEEE double-precision floating-point arithmetic allots 11 bits for the exponent and 52 bits for the mantissa (Goldberg 1991). In our discussion we assume only that the number of bits assigned to the exponent and mantissa are fixed to some known values.

With the above assumption, let $\mathbb{M} \subseteq \mathbb{Q}$ be the set of floating-point-representable rational numbers. Note that $\mathbb{M}$ is finite. If $q \in \mathbb{M}$, then both $-q$ and $\hat{q}$ are members of $\mathbb{M}$. Also, if $a, b \in \mathbb{M}$, then $\min\{a, b\}$ and $\max\{a, b\}$ are members of $\mathbb{M}$. The set $\mathbb{M}$ does not have much structure; it is not a monoid since associativity for addition does not hold.

We will refer to rational inequalities whose coefficients can be represented as members of $\mathbb{M}$ as $\mathbb{M}$-*representable* inequalities.

3.2. Approximating Real Functions

Given a function $f\colon \mathbb{R}^n \to \mathbb{R}$, we say that a function $f^{\mathrm{up}}\colon \mathbb{M}^n \to \mathbb{M}$ upper approximates f in $\mathbb{M}$ if

$$f^{\mathrm{up}}(x) \ge f(x) \quad \text{if } x \in \mathbb{M}^n,$$

and we say that a function $f^{\mathrm{dn}}\colon \mathbb{M}^n \to \mathbb{M}$ lower approximates f in $\mathbb{M}$ if

$$f^{\mathrm{dn}}(x) \le f(x) \quad \text{if } x \in \mathbb{M}^n.$$

Note that the basic arithmetic operations $+$ and $*$ are functions from $\mathbb{R}^2 \to \mathbb{R}$. We denote, respectively, by $\overline{a+b}$ and $\underline{a+b}$ the value of the upper and lower approximation of $a+b$ for any $a, b, a+b \in \mathbb{M}$, and use a similar notation for subtraction and multiplication. (In the C programming language (ISO/IEC 1999), IEEE floating-point rounding conventions can be set to produce upper and lower approximations using the `fesetround` function. Upper approximations are obtained by first calling the function with the argument `FE_UPWARD` and then carrying out the usual arithmetic operation. Similarly, lower approximations are obtained using the `FE_DOWNWARD` argument.)

Let $k \ge 3$ be an integer and let $a_i \in \mathbb{M}$, $\pi_i \in \mathbb{M}$, for $i = 1, \ldots, k$. We define upper and lower approximations for $\sum_{i=1}^k \pi_i a_i$ by

$$\overline{\sum_{i=1}^{k} \pi_i a_i} := \overline{\left(\overline{\sum_{i=1}^{k-1} \pi_i a_i}\right) + \overline{\pi_k a_k}}$$

and

$$\underline{\sum_{i=1}^{k} \pi_i a_i} := \underline{\left(\underline{\sum_{i=1}^{k-1} \pi_i a_i}\right) + \underline{\pi_k a_k}},$$

respectively. It is important to notice that any ordering of $a_1, \ldots, a_k$ will yield an upper and lower approximation of $\sum_{i=1}^k \pi_i a_i$, but different orders may yield different results. Thus, the operation is not commutative.

In addition, note that these upper and lower approximations may not be correct if an overflow or underflow occurs, and whenever these exceptions are raised, we consider the result as not being defined.

3.3. Safe Row Aggregation

Consider a polyhedron

$$P = \{x \in \mathbb{R}^n\colon Ax \ge d,\ l \le x \le u\},$$

with $A \in \mathbb{M}^{m\times n}$, $d \in \mathbb{M}^m$, $u \in \{\mathbb{M} \cup \{+\infty\}\}^n$, $l \in \mathbb{M}^n$, and $0 \le l \le u$. Given a set of multipliers $\lambda \in \mathbb{M}^m$, with $\lambda \ge 0$, the inequality $\lambda^T Ax \ge \lambda^T d$ is satisfied by all $x \in P$, but this inequality may not be $\mathbb{M}$-representable. Furthermore, when we compute $\lambda^T A$ and $\lambda^T d$ in floating-point arithmetic, the resulting inequality may not even be valid for all points in P. Nonetheless, as $x \ge 0$, we have that

$$\sum_{j\in N} \left(\overline{\sum_{i=1}^{m} \lambda_i a_{ij}}\right) x_j \ge \underline{\sum_{i=1}^{m} \lambda_i d_i}$$

is an $\mathbb{M}$-representable valid inequality. We can perform the safe aggregation of systems of inequalities in this manner.

The same concept can be applied to equality systems $Ax = d$, but in this case the multipliers λ need not be restricted to nonnegative values. However, in this case, we are not guaranteed to obtain a valid $\mathbb{M}$-representable equality. Instead, we need to relax the equalities to inequalities and then aggregate them using the multipliers λ to obtain a valid $\mathbb{M}$-representable inequality. Obviously, we need to relax the equalities to have a sense that matches the desired sense of the $\mathbb{M}$-representable inequality. For example, if we want to obtain a $\geq\mathbb{M}$-representable inequality and the multiplier of a given constraint is negative, we need to relax the equation to $\leq$. This means that, given an equality system and multipliers λ, one can obtain two $\mathbb{M}$-representable valid inequalities: one in $\leq$ form and one in $\geq$ form.

3.4. Safe Substitution of Slack Variables

Suppose $P \subseteq \mathbb{R}_+^{n+1}$ is a polyhedron such that any point $(x, s) \in P$ satisfies the following $\mathbb{M}$-representable constraints:

$$\sum_{j\in N} c_j x_j - s = d, \tag{8}$$

$$\sum_{j\in N} \pi_j x_j + \rho s \geq \pi_o. \tag{9}$$

We wish to eliminate s from (9) while maintaining its validity (here, s can be thought of as a nonnegative slack variable).

This can be done via safe aggregation. Adding ρ times (8) to (9) gives the valid inequality

$$\sum_{j\in N} (\pi + \rho c_j) x_j \geq \pi_o + \rho d.$$

Therefore, the $\mathbb{M}$-representable inequality

$$\sum_{j\in N} (\overline{\pi + \rho c_j}) x_j \geq \underline{\pi_o + \rho d}$$

is valid for P, and we have safely removed s from (9).

4. Safe MIR Inequalities

For the single-row relaxation (2) in §2, we described the MIR inequality

$$\sum_{i\in I} f(a_i, b) x_i + \sum_{i\in C} h(a_i) x_i \geq \hat{b}\lceil b \rceil,$$

where $f(q_1, q_2) := \min\{\hat{q}_1, \hat{q}_2\} + \hat{q}_2 \lfloor q_1 \rfloor$ and $h(q) := \max\{q, 0\}$. To obtain a safe version of this cutting plane in floating-point arithmetic, define the upper approximation $f^{\text{up}}\colon \mathbb{M} \times \mathbb{M} \to \mathbb{M}$ of f as

$$f^{\text{up}}(q_1, q_2) := \overline{(\min\{\hat{q}_1, \hat{q}_2\} + \overline{\hat{q}_2 \lfloor q_1 \rfloor})}$$

and define $h^{\text{up}}(q) := h(q)$ (no rounding is needed).

As $x \geq 0$, we have

$$\sum_{i\in I} f^{\text{up}}(a_i, b) x_i + \sum_{i\in C} h^{\text{up}}(a_i) x_i \geq \sum_{i\in I} f(a_i, b) x_i + \sum_{i\in C} h(a_i) x_i \geq \hat{b}\lceil b \rceil \geq \underline{\hat{b}\lceil b \rceil}.$$

Therefore,

$$\sum_{i\in I} f^{\text{up}}(a_i, b) x_i + \sum_{i\in C} h^{\text{up}}(a_i) x_i \geq \underline{\hat{b}\lceil b \rceil} \tag{10}$$

is an $\mathbb{M}$-representable valid inequality for K, which we call a *safe MIR* inequality.

Recall that before writing the c-MIR inequality (7), we first complemented and shifted variables to obtain the inequality (6). In this context, it is easy to see that given an $\mathbb{M}$-representable inequality $\sum_{j\in N} a_j x_j \geq b$, and $\mathbb{M}$-representable bounds u_j for $j \in U$, and l_j for $j \in L$, the inequality

$$\sum_{j\in U} (-a_j) x_j^u + \sum_{j\in L} a_j x_j^l + \sum_{j\notin U\cup L} a_j x_j \geq \underline{b - \overline{\sum_{j\in U} a_j u_j} - \overline{\sum_{j\in L} a_j l_j}}, \tag{11}$$

is a valid $\mathbb{M}$-representable inequality. This is because

$$\overline{\sum_{j\in U} a_j u_j} \geq \sum_{j\in U} a_j u_j \quad \text{and} \quad \overline{\sum_{j\in L} a_j l_j} \geq \sum_{j\in L} a_j l_j.$$

To obtain safe c-MIR inequalities for K_B, we simply start with a valid $\mathbb{M}$-representable inequality, perform safe variable complementation and shifting as in (11), apply the safe MIR inequality (10), and then perform safe complementation and shifting to get a valid $\mathbb{M}$-representable inequality in the original variables.

5. Computational Study

In our study we consider two scenarios where safe MIR procedures can potentially be useful. The first set of tests is concerned with the use of safe cuts in cases that demand accurate bounds. We use the traveling salesman problem (TSP) as a case study, applying multiple rounds of safe Gomory cuts to improve LP relaxations generated by TSP-specific methods.

The second tests consider the repeated application of Gomory cuts for general MIP instances. Our tests aim to give an indication of the possible improvements in LP bounds that can be obtained by adopting the safe methods, where multiple rounds of cuts can be added without generating invalid cuts. Moreover, we test if our measures to generate safe cuts have a significant impact on the empirical strength of the cuts.

Note that as an alternative to our safe MIR approach, the computations could be carried out entirely in exact rational arithmetic using the LP solver of Applegate et al. (2007) and rational versions of the

MIR procedure. On large instances such an approach is considerably more time-consuming, however, because of the complexity of rational computations on the difficult LP instances that are created (see, for instance, Espinoza 2006). Moreover, the approach adopted here demonstrates the feasibility of including safe MIR methods in standard floating-point–based software.

5.1. Generating Gomory Cuts

Given an MIP instance (1), our cut-generation process uses the simplex algorithm to solve the current LP relaxation, which is then strengthened by adding a *round* of Gomory cuts.

Let x^* be the current basic optimal solution. Each round consists of the following steps.

Step 1. *Slack variables*. Add a slack variable to each row of the model to obtain equality constraints, and consider all slacks as nonnegative continuous variables with no upper bounds.

Step 2. *Variable complementation*. Define the index sets $U = \{j \in N: u_j - x^*_j \leq x^*_j - l_j\}$ and $L = \{j \in N: u_j - x^*_j > x^*_j - l_j\}$.

Step 3. *Rank the fractional variables*. The integer variables x_j that take on a fractional value x^*_j in the current basis are ranked in nondecreasing value of $|\hat{x}^*_j - 0.5|$.

Step 4. *Compute the cuts*.

(a) Process the selected variables in the ranked order.

(b) Safely compute the tableau row corresponding to the current selected variable using aggregation based on the multipliers obtained from the row of the basis inverse given by the LP solver.

(c) For each tableau row safely compute the c-MIR inequality using the sets U and L as defined above, after scaling by 1, 2, 3, as proposed in Cornuéjols et al. (2003).

(d) Only generate c-MIR cuts that are violated by at least 0.001.

(e) Stop this procedure if 500 violated cuts are generated.

Step 5. *Substitute slacks*. Safely substitute slacks in the cuts.

Step 6. *Scale the cuts*. Scale each cut $\pi x \geq \pi_o$ safely so that $\max\{\|\pi\|_\infty, |\pi_o|\} = 1$.

Step 7. *Add the cuts*. Add the computed cuts to the LP and resolve.

8. *Remove cuts*. Remove from the LP all previously added cuts that have dual variables equal to zero.

This is a straightforward implementation of Gomory cuts, where all the parameters were chosen according to previous experience by the authors in generating Gomory cuts. We do not claim that this is the best choice for implementing Gomory cuts. Instead, the purpose of the paper is to show the advantages of using safe cuts in a reasonable Gomory cut implementation. For a discussion of general cut selection, see Achterberg (2007) and Goycoolea (2006).

An additional implementation detail that one should keep in mind is that the cut violation is checked before scaling, and therefore, it is possible that the added cuts may not be violated enough to change the optimal LP solution returned by the solver. Our code detects such a phenomenon and halts the cut-generation process in case it occurs. In addition, we impose a limit of 128 rounds of cuts. Also, for simplicity, we substitute ranged constraints in the original model by two independent constraints.

Another detail to be mentioned is how to perform the step of computing the tableau row using the multipliers obtained from the row of the basis inverse. As mentioned in §3.3, one cannot safely aggregate equality constraints to get a safe $\mathbb{M}$-representable equality. Thus, before doing the aggregation, we relax each equality constraint to $\geq$ if the associated multiplier of that constraint is positive and to $\leq$ if it is negative, and therefore obtain a safe $\mathbb{M}$-representable inequality that approximates the tableau row.

Finally, recall that the validity of the Gomory cuts depend on the fact that no exceptions occurred, as mentioned in §3.2, so we check if all our cut computations incur an overflow, an underflow, a division by zero, or an invalid operation (such as $0/0$ or $\infty - \infty$). This is easily done by calling the standard C function `fetestexcept` with the argument `FE_OVERFLOW | FE_UNDERFLOW | FE_INVALID | FE_DIVBYZERO`, which returns a bitmap where each bit corresponds to a flag indicating if the corresponding exception occurred. Once any of these exception flags is set, it will remain set until manually cleared. To clear such flags, one can call `feclearexcept` with the same arguments. In all our experiments, these exceptions never occurred.

5.2. TSPLIB Results

Early cutting-plane research on the TSP by Martin (1966), Miliotis (1978), and others adopted general-purpose MIP codes for improving LP relaxations. In later work, these methods were replaced by algorithms for generating TSP-specific cutting planes. With the safe MIR procedures it is possible to combine these ideas, using general cuts to further improve relaxations obtained by problem-specific methods.

We consider MIP relaxations obtained by long runs of the Concorde TSP solver (Applegate et al. 2006). The relaxations were found with `concorde -mC48 -Z3`, the strongest recommended setting of the Concorde separation routines. This setting uses repeated local cuts, up to size 48, as well as the domino-parity inequalities (see Applegate et al. 2006).

In Table 1 we give statistics for the relaxations for all TSPLIB instances having at least 3,000 cities. The only instance left out was fl3795 since, in that case,

Table 1 **MIP Relaxations of the TSP**

Name	Variables	Optimal	LP value	+Cuts	NCuts	Rds.	Gap closed (%)
pcb3038	6,976	137,694	137,684.25	137,684.52	11,010	22	2.51
fnl4461	10,129	182,566	182,558.55	182,559.97	18,000	36	19.08
rl5915	24,939	565,530	565,484.03	565,487.91	28,000	56	8.44
rl5934	25,285	556,045	555,994.47	556,001.02	21,000	42	12.97
pla7397	27,209	23,260,728	23,258,946.65	23,259,208.71	64,556	128	14.71
rl11849	70,259	923,288	923,208.71	923,210.01	38,008	76	1.53
usa13509	129,837	19,982,859	19,981,199.08	19,981,229.43	27,000	54	1.83
brd14051	83,221	469,385	469,353.80	469,353.91	5,500	11	0.35
d15112	110,072	1,573,084	1,572,966.32	1,572,967.08	16,500	33	0.64
d18512	141,025	645,238	645,194.86	645,195.17	20,528	41	0.71
pla33810	67,683 (+)	660,048,945	66,001,233.03	66,001,901.19	64,392	128	1.40
pla85900	167,870 (+)	142,382,641	142,296,659.63	142,299,299.63	64,058	128	3.07

the Concorde LP bound establishes the optimality of the tour. The original number of variables in all these instances would be very large even to load the problem in memory, and thus we run our code on the MIPs obtained after fixing variables to zero by reduced cost. This reduced-cost fixing is based on the Concorde relaxation, and since it is guaranteed not to cut off the optimal solution, any LP bound obtained with general MIP cuts is a valid bound for the original TSP. In the cases of the two largest instances, pla33810 and pla85900, however, a large number of variables still remains after reduced-cost fixing; therefore we use an MIP with only a subset of the remaining variables, and hence the final bounds are not necessarily valid for the original TSP.

The optimal value of the Concorde LP for each instance is reported in the "LP value" column of Table 1. In the "+Cuts" column we report the improved lower bounds obtained by multiple rounds of safe Gomory cuts, after applying the safe-LP bounding technique from Neumaier and Shcherbina (2004). The total number of rounds of Gomory cuts is reported in column "Rds." and the total number of Gomory cuts added is reported in column "NCuts." The improvements range from 0.35% of the Concorde LP optimality gap for brd14051, up to 19.09% of the gap for fnl4461. It is interesting, however, that the very strong LP relaxations obtained by Concorde could be improved with general-purpose MIP cuts, suggesting that this may be a technique worth considering for problem classes that have not received the intense study given to the TSP.

To further illustrate the use of the safe MIR procedure, we constructed a valid MIP relaxation for the 85,900-city TSP instance pla85900, using the full set of variables that were not eliminated by reduced-cost fixing. This much stronger relaxation was obtained from the very long computation described in Applegate et al. (2006). The result after adding Gomory cuts to this instance is reported in Table 2, showing a 0.62% increase in the lower bound. Although this is a modest increase, it is possible that the LP solution is sufficiently altered to permit the further use of Concorde's TSP-specific cutting planes in an iterative fashion. (We have not tested this idea.)

5.3. MIPLIB Results

To test the effectiveness of repeated rounds of Gomory cuts in general, we constructed a test set consisting of the union of the following instances:

- The full set of 65 instances in the MIPLIB 3.0 collection (Bixby et al. 1998).
- The 27 instances in the MIPLIB 2003 collection (Achterberg et al. 2006) that are not included in MIPLIB 3.0.
- The 13 TSPLIB-MIP instances described in §5.2.

We excluded four of these instances because of the presence of variables that may assume negative values (that are not handled in our current implementation), leaving a total of 101 instances. The excluded instances are dsbmip, mzzv11, mzzv42z, and rd-rplusc-21.

To illustrate the impact of multiple rounds of safe cutting planes, we recorded the lower bounds obtained after 1, 2, 4, 8, 16, and 128 rounds of our cutting-plane procedure; if the optimal LP basis does not change after a round of cutting planes, then the particular run is terminated. The lower bounds reported are the ones obtained from the LP solver, which are not guaranteed to be accurate. Ideally, one would use a method like the one of Neumaier and Shcherbina (2004) to compute valid lower bounds, but their method can only be applied to bounded instances. Therefore, we use the LP solver's lower bounds as an estimate of the true bounds.

In these tests, three of the instances were excluded because they each have zero integrality gap (disctom,

Table 2 **Valid MIP Relaxation for pla85900**

Name	Variables	TSP optimal	LP value	+Cuts	Gap closed
pla85900f	300,969	142,382,641	142,381,453.65	142,381,460.98	0.62%

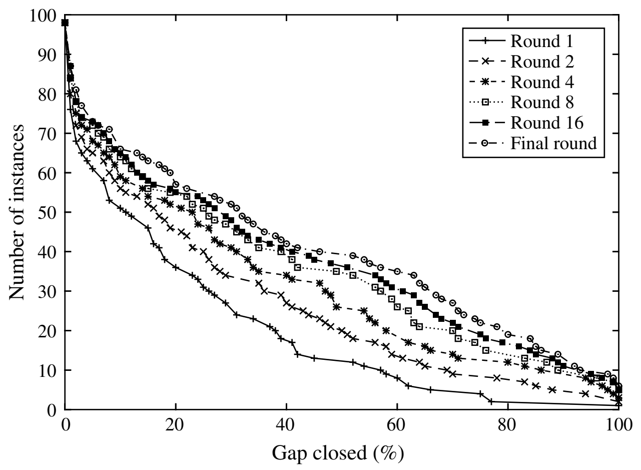

Figure 1 Multiple Rounds of Safe Gomory Cuts

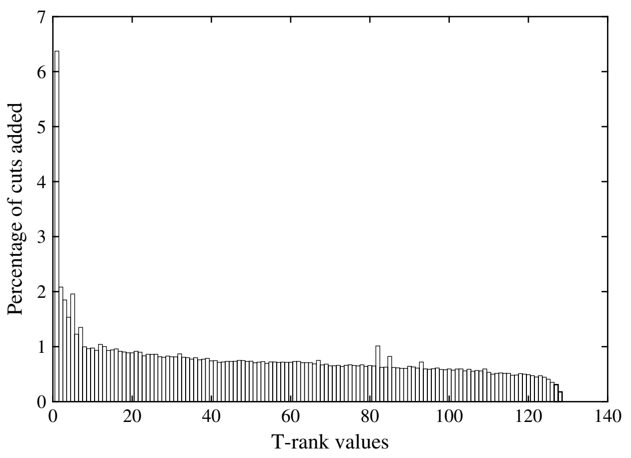

Figure 2 Distribution of T-rank Values of Cuts

enigma, noswot), leaving a total of 98 instances. The results are displayed as curves in Figure 1. The points (x, y) in each curve represent how many instances y closed at least x fraction of the initial integrality gap after adding the indicated number of rounds of cuts. For example, after one round of cuts, 50% of the gap was closed in 12 instances, whereas four rounds of cuts closed 50% of the gap in 26 instances.

Note that even though we are running these experiments doing up to 128 rounds of cuts, this does not mean that we are generating rank 128 cuts. It might well be the case that, since at each round we are only generating at most 500 cuts, we actually only generate rank one cuts throughout the experiment. Because it is hard to determine the exact rank of a cut, we define a computationally tractable variant of the notion of rank, which we call tableau rank (or T-rank for short). Given an MIP formulation, a Gomory cut has T-rank 1 if the associated tableau row from which it is derived is a linear combination of inequalities in the formulation. A cut has T-rank k for some $k \geq 2$ if the associated tableau row is a linear combination of inequalities in the formulation and Gomory cuts with T-rank $k-1$ or less and at least one Gomory cut with T-rank $k-1$. The T-rank of a Gomory cut is an upper bound on its rank. Figure 2 shows that we are indeed generating high T-rank cuts. Each bar in the figure shows what percentage of the total number of cuts generated on all instances had that corresponding T-rank. As expected, low T-rank cuts have a higher percentage, but there is a significant number of cuts being generated with T-rank, all the way up to 128.

We ran some additional tests to determine if the high T-rank cuts are in fact making a difference in the final bound. We ran the code again, doing up to 128 rounds of cuts, but we did not allow the addition of cuts that have T-rank higher than some prespecified parameter (controlled by parameter `-limrank` in the code). Table 3 shows the average gap closed in each of those sets of runs and indicates that indeed the higher T-rank cuts contribute significantly to closing the duality gap.

We also ran experiments to compare the safe version of our Gomory cut implementation with the unsafe one (the unsafe version of Gomory cuts is exactly the same as the safe version, except that all the measures to guarantee safety are turned off). The purpose of this experiment is to test the impact of the safe cut generation procedure on the quality of the final bound obtained.

We noticed that even though the safe and unsafe cuts generated in a particular round can be very similar, the final bounds obtained after several rounds of cuts can vary greatly on individual instances.

Figure 3 compares the final gap closed by adding at most 128 rounds of safe cuts versus at most 128 rounds of the unsafe version of Gomory cuts. The two curves indicate that in spite of individual variations on the final bounds, on average, no significant loss in the bounds is incurred with the use of the safe (but potentially weaker) cutting planes.

We also plot in Figure 4 the average density of the cuts added for each instance. The value in the y axis represents the average density:

$$\text{AvgDens} := \frac{\sum_{c\in\mathscr{C}} NZ(c)}{NCOLS * |\mathscr{C}|},$$

Table 3 Comparison of Lower Bounds Using Different T-rank Safe Gomory Cuts

Max. T-rank	Average gap closed (%)
1	24.33
2	31.07
4	34.79
8	37.24
16	38.86
≤128	39.69

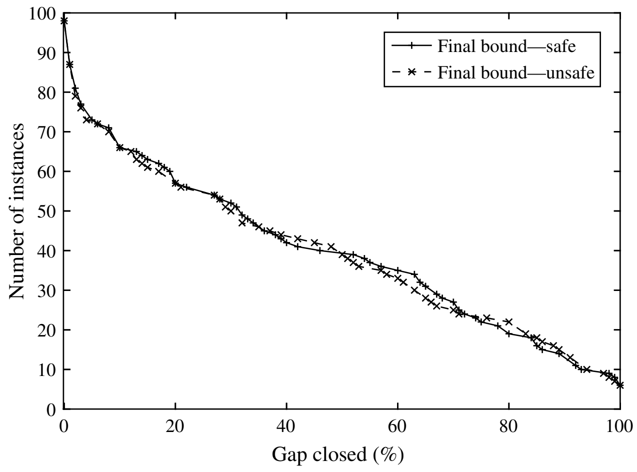

Figure 3 **Safe vs. Unsafe Gomory Cuts**

where $\mathscr{C}$ is the set of all cuts added, $NZ(c)$ is the number of nonzeroes in a cut $c \in \mathscr{C}$, and $NCOLS$ is the number of variables in the given instance. We sorted the instances in increasing order of AvgDens for the safe cuts.

Figure 4 seems to indicate that unsafe cuts are slightly sparser on average than the safe ones, which is somewhat surprising, considering that we are not taking any measures to achieve this goal.

6. Final Remarks

Gomory cuts are widely used in commercial MIP solvers, but because of arithmetic errors, it is common to use ad hoc techniques to reduce the risk of generating invalid cuts, such as relaxing the right-hand side of Gomory cuts and limiting their T-rank. However, these ad hoc techniques do not guarantee validity of the generated cuts.

We have presented a method to generate safe Gomory cuts in floating-point arithmetic using standard C routines. The advantage of using safe Gomory cuts is that one can add multiple rounds of cuts and obtain a significant improvement in the LP bound with the guarantee that all cuts generated are valid. Furthermore, in our experiments, we observed no significant loss in the quality of bounds obtained by using safe Gomory cuts as opposed to their unsafe counterparts computed using standard floating-point arithmetic.

This does not mean, however, that if we add these cuts instead of unsafe Gomory cuts in current floating-point–based MIP solvers that we are guaranteed (or even have a better chance) to obtain the correct optimal solution. For example, a solver could incorrectly assert that a solution to the MIP problem violates a safe Gomory cut if it evaluated the cut in finite precision arithmetic. Moreover, there are other parts of the solver that also need to be safe, such as computation of dual bounds, preprocessing, and verification of LP infeasibility.

Another point that should be emphasized is that by using safe cuts, one cannot use some of the ad hoc measures that are usually applied to unsafe cuts to make them more numerically stable and/or more safe—for example, rounding the coefficients to zero when they are very small. Unsafe cuts that use these safeguards can have advantages over safe cuts, such as being sparser, so these issues have to be weighed more carefully in those cases.

Acknowledgments
The authors thank three anonymous referees for their helpful comments. The first author was supported by National Science Foundation Grant CMMI-0726370 and Office of Naval Research Grant N00014-03-1-0040. The fourth author was supported by FONDECYT Grant 11075028 and Basal project CMM, Universidad de Chile.

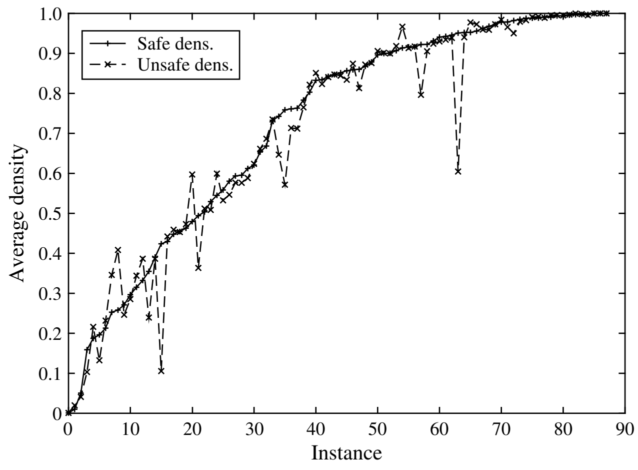

Figure 4 **Comparison of Average Cut Density for Safe vs. Unsafe Gomory Cuts**

References

Achterberg, T. 2007. Constraint integer programming. Ph.D. thesis, Technische Universität Berlin, Berlin.

Achterberg, T., T. Koch, A. Martin. 2006. MIPLIB 2003. *Oper. Res. Lett.* **34**(4) 361–372.

Applegate, D., R. E. Bixby, V. Chvátal, W. Cook. 2006. *The Traveling Salesman Problem: A Computational Study*. Princeton University Press, Princeton, NJ.

Applegate, D., W. Cook, S. Dash, D. G. Espinoza. 2007. Exact solutions to linear programming problems. *Oper. Res. Lett.* **35**(6) 693–699.

Balas, E., A. Saxena. 2008. Optimizing over the split closure. *Math. Programming* **113**(2) 219–240.

Balas, E., S. Ceria, G. Cornuéjols, N. Natraj. 1996. Gomory cuts revisited. *Oper. Res. Lett.* **19**(1) 1–9.

Bertsimas, D., R. Weismantel. 2005. *Optimization Over Integers*. Dynamic Ideas, Belmont, MA.

Bixby, R. E., S. Ceria, C. M. McZeal, M. W. P. Savelsbergh. 1998. An updated mixed integer programming library: MIPLIB 3.0. *Optima* **58** 12–15.

Bixby, R. E., M. Fenelon, Z. Gu, E. Rothberg, R. Wunderling. 2000. MIP: Theory and practice—Closing the gap. M. J. D. Powell, S. Scholtes, eds. *System Modelling and Optimization: Methods, Theory and Applications*. Kluwer Academic Publishers, Dordrecht, The Netherlands, 19–49.

Bixby, R. E., M. Fenelon, Z. Gu, E. Rothberg, R. Wunderling. 2004. Mixed-integer programming: A progress report. M. Grötschel, ed. *The Sharpest Cut: The Impact of Manfred Padberg and His Work*. SIAM, Philadelphia, 309–325.

Cornuéjols, G., Y. Li, D. Vandenbussche. 2003. *K*-Cuts: A variation of Gomory mixed integer cuts from the LP tableau. *INFORMS J. Comput.* **15**(4) 385–396.

Dash, S., M. Goycoolea, O. Günlük. 2009. Two-step MIR inequalities for mixed-integer programs. *INFORMS J. Comput.*, ePub ahead of print August 18, http://joc.journal.informs.org/cgi/content/abstract/ijoc.1090.0337v1.

Dash, S., O. Günlük, A. Lodi. 2008. MIR closures of polyhedral sets. *Math. Programming*, ePub ahead of print May 17.

Espinoza, D. 2006. On linear programming, integer programming and cutting planes. Ph.D. thesis, Georgia Institute of Technology, Atlanta.

Goldberg, D. 1991. What every computer scientist should know about floating-point arithmetic. *ACM Comput. Surveys* **23**(1) 5–48.

Gomory, R. E. 1960. An algorithm for the mixed integer problem. Research Memorandum RM-2597, The Rand Corporation, Santa Monica, CA.

Goycoolea, M. 2006. Cutting planes for large mixed integer programming models. Ph.D. thesis, Georgia Institute of Technology, Atlanta.

IEEE. 1985. Standard for binary floating point arithmetic. ANSI/IEEE Standard 754-1985, Institute of Electrical and Electronics Engineers, Washington, DC.

ISO/IEC. 1999. Programming languages—C. Standard ISO/IEC 9899:1999, International Organization for Standardization, Geneva.

Marchand, M., L. A. Wolsey. 2001. Aggregation and mixed integer rounding to solve MIPs. *Oper. Res.* **49**(3) 363–371.

Margot, F. 2009. Testing cut generators for mixed-integer linear programming. *Math. Programming Comput.*, ePub ahead of print May 29.

Martin, G. T. 1966. Solving the traveling salesman problem by integer linear programming. Technical report, C-E-I-R, New York.

Miliotis, P. 1978. Using cutting planes to solve the symmetric travelling salesman problem. *Math. Programming* **15**(1) 177–188.

Nemhauser, G. L., L. A. Wolsey. 1988. *Integer and Combinatorial Optimization*. John Wiley & Sons, New York.

Neumaier, A., O. Shcherbina. 2004. Safe bounds in linear and mixed-integer linear programming. *Math. Programming* **99**(2) 283–296.

Reinelt, G. 1991. TSPLIB—A traveling salesman problem library. *ORSA J. Comput.* **3**(4) 376–384.

Wolsey, L. A. 1998. *Integer Programming*. John Wiley & Sons, New York.