

Safe bounds in linear and mixed-integer programming

Arnold Neumaier and Oleg Shcherbina

Institut für Mathematik, Universität Wien
Strudlhofgasse 4, A-1090 Wien, Austria
email: Arnold.Neumaier@univie.ac.at, oleg@mat.univie.ac.at
WWW: <http://www.mat.univie.ac.at/~neum/>

June 18, 2002

Abstract. Current mixed-integer linear programming solvers are based on linear programming routines that use floating point arithmetic. Occasionally, this leads to wrong solutions, even for problems where all coefficients and all solution components are small integers. It is shown how, using directed rounding and interval arithmetic, cheap pre- and postprocessing of the linear programs arising in a branch-and-cut framework can guarantee that no solution is lost, at least for mixed-integer programs in which all variables can be bounded rigorously by bounds of reasonable size.

Keywords: Linear programming, mixed-integer programming, rounding errors, directed rounding, interval arithmetic, branch-and-cut, lower bounds, mixed-integer rounding, generalized Gomory cut, safe cuts, safe presolve, certificate of infeasibility.

2000 MSC Classification: primary 90C11, secondary 65G20

1 Introduction

Rounding errors permeate most numerical calculations, and their effect on the final results of an algorithm is a well-studied problem in many branches of numerical analysis. However, very little work seems to have been done about the effect of rounding errors on the quality of solutions in linear and mixed integer programming.

The only study we are aware of, by FOURER & GAY [5], is restricted to the presolve phase of solving a linear program (LP); cf. also BREARLY et al. [1].

It contains the observation that rounding errors in presolve may change the status of an LP from feasible to infeasible, or vice versa. The solution they presented was a straightforward application of directed rounding during presolve. It apparently eliminates these problems because of the simple nature of the presolve operations.

It is well-known that rounding errors affect the solution of linear systems and hence of linear programming solvers. Due to the lack of monotonicity of elimination algorithms, directed rounding is no longer useful.

Usually, backward error analysis is invoked to argue that the computed 'solution' is the exact solution of a problem with slightly perturbed coefficients, so that it should be adequate for almost all problems of interest. However, backward error analysis has no relevance for integer linear programs with integer coefficients, since slightly perturbed coefficients no longer produce problems of the same class.

Moreover, the influence of rounding errors on the solution may be large if the problem has a large condition number. A recent study by ORDÓÑEZ & FREUND [18] revealed that 72% of the (real life!) problems of the NETLIB LP collection [14] are ill-conditioned; after preprocessing (with CPLEX 7.1 presolve), 19% still remained ill-conditioned. Thus ill-condition seems to be fairly frequent in practice.

In particular, ill-condition is likely to arise in LP subproblems created by branch-and-cut procedures at the nodes of the branch tree for integer linear programs. This may result in incorrect decisions, leading to a loss of the solution in branch-and-cut procedures. It is difficult to say how rare or frequent such failures are in practice, since checking rigorously whether a 'solution' obtained for a given MIP is optimal is usually considered too hard to be attempted.

In the following, we show that sometimes, solutions are indeed lost on some problems, when attempting to 'solve' them by commercial state-of-the-art mixed integer programming (MIP) solvers. Already for a simple and innocent-looking low-dimensional problem with only integer variables and small integer coefficients, this happened for six solvers out of seven solvers tried.

We therefore discuss the use of directed rounding and interval arithmetic to guarantee correct decisions in branch-and-cut procedures. (For basics on interval arithmetic, see [13, 15, 16].) It turns out that safety in solving MIPs can be rigorously guaranteed with limited additional computational effort by careful preparation of the LP subproblems and careful postprocessing of the LP solver output. Of particular importance is the fact that the LP solver itself is not required to deliver rigorous results, so that off the shelf software can be used. Since, typically, most of the CPU time is spent in the LP solver, this also implies that the addition of safety should increase the run time by a small amount only.

A recent preprint of JANSSON [9] contains new results on rigorously solving linear programs with uncertain coefficients, using interval arithmetic. For the case of exact coefficients, some of his results overlap with those of Section 3 of the present paper.

In the following, \mathbb{Z} denotes the set of integers, and \mathbb{N} the set of positive integers. x_K denotes the subvector of x indexed by the indices from K . Boxes (interval vectors) are written in bold face.

Acknowledgment

We acknowledge support by the European Community research project COCONUT [4], project reference IST-2000-26063. We also want to thank ILOG for providing us with a CPLEX license while working on this project.

2 Failure of MIP solvers

The commercial MIP solver CPLEX [7] is a professionally maintained, high quality state-of-the-art solver for mixed integer programming problems. While it solved and solves bigger and bigger MIP problems to full user satisfaction,

it is based (as all other state of the art MIP solvers) on floating-point arithmetic, and therefore potentially vulnerable to the errors introduced by the limited precision.

Unfortunately, this occasionally even affects the solution of pure integer problems with integer coefficients where, due to their discrete nature, the user is likely to expect exact results. Here we document a simple example where CPLEX (and many other MIP solvers) failed to find an existing feasible point. The failure is due to rounding errors and not due to the complexity of the search.

We tried to solve the 20 variable integer linear problem

$$\begin{aligned}
 \min \quad & -x_{20} \\
 \text{s.t.} \quad & (s+1)x_1 - x_2 \geq s-1, \\
 & -sx_{i-1} + (s+1)x_i - x_{i+1} \geq (-1)^i(s+1) \text{ for } i = 2 : 19, \\
 & -sx_{18} - (3s-1)x_{19} + 3x_{20} \geq -(5s-7), \\
 & 0 \leq x_i \leq 10 \text{ for } i = 1 : 13, \\
 & 0 \leq x_i \leq 10^6 \text{ for } i = 14 : 20, \\
 & \text{all } x_i \text{ integers,}
 \end{aligned} \tag{1}$$

for the special choice $s = 6$ with CPLEX (Version 7.1.0, March 2001) on a LINUX platform, both with and without presolve. In both cases, CPLEX returned after 16 iterations at the root node the message 'Integer infeasible. Current MIP best bound is infinite.' No further diagnostic information was available.

Surprisingly, upon adding the additional constraint $x_{20} \leq 10$ to this 'infeasible' problem CPLEX produced (after 11 iterations) the solution

$$x = (1, 2, 1, 2, \dots, 1, 2)^T.$$

The same solution was also found with the original constraints, upon changing the objective function to $+x_{20}$. It is easily checked that this is a feasible point (probably the only one). Thus the negative result CPLEX produced on the original problem was erroneous. (The same happened with the more recent version 7.5.0.)

Upon inspection, one finds that the solution is a nondegenerate vertex of the linear programming relaxation (but not the solution of the LP relaxation). The coefficient matrix of the linear constraints active at the solution is nonsingular but extremely ill-conditioned; the numerical rank is 19. It is

likely that CPLEX suffers from the rounding errors introduced through this ill-conditioning.

We also ran the original problem on all MIP solvers (and the only MINLP solver with AMPL input) available (on June 18, 2002) through NEOS [3, 6], namely BONSAIG, FortMP, GLPK, XPRESS, XPRESS-MP/INTEGER, and MINLP. Only FortMP solved the original problem correctly. The other five solvers reported 'no solution found' (BONSAIG), 'global search complete – no integer solution found' (XPRESS) or even, erroneously, 'integer infeasible' (GLPK,XPRESS-MP/INTEGER,MINLP).

Similar difficulties might arise in all mixed-integer problems where the solution of some LP subproblem produces a vertex whose basis is nearly linearly dependent, so that the linear systems solved are ill-conditioned, making the conclusions drawn from the output of the LP solver of dubious value.

Other currently available MIP solvers probably have the same behavior on similar problems, even should they solve the above example correctly. (Note that different MIP solvers may behave differently on this example since rounding errors depend on the algorithm and the arithmetic used.)

On the other hand, the techniques discussed below provide rigorous error control as far as bounding decisions are concerned: Since we have finite (though not very small) bounds for all variables, the results discussed below would provide rigorous bounds and cutting planes, and would thus avoid the error. Lacking access to the source code of CPLEX, we are not able to repeat the tests using the rigorous safeguards derived here.

3 Rigorous bounds on the objective

In mixed integer programming (see, e.g., WOLSEY [21]) linear programs are typically used to find lower bounds on the objective that allow one to decide whether a given node in the branch tree can be fathomed. For reliable results, it is therefore imperative that the computed lower bound is rigorously valid.

However, the output of a linear programming routine is the result of an approximate calculation and hence is itself approximate. Obtaining rigorous error bounds for the solution of linear programming problems is a difficult task (cf. KRAWCZYK [11], JANSSON [8], JANSSON & RUMP [10]), especially in the ill-conditioned case, where the active set might not be identified cor-

rectly.

Fortunately, it is possible to postprocess the approximate result to obtain *rigorous bounds for the objective* with reasonable effort, provided that reasonable bounds on all variables are available. Such bounds are frequently computed anyway in a preprocessing phase using a limited form of constraint propagation, and if the latter is done with sufficient care (using directed rounding to ensure the mathematical validity of each step of the process, as in AMPL [5]), these bounds on the variables are rigorously valid.

We begin by looking at the simpler case where the linear program is given in the standard form

$$\begin{aligned} \min \quad & c^T x \\ \text{s.t.} \quad & Ax = b, \quad x \geq 0. \end{aligned} \tag{2}$$

Its dual is

$$\begin{aligned} \max \quad & b^T y \\ \text{s.t.} \quad & A^T y \leq c. \end{aligned} \tag{3}$$

Now suppose that y is an approximate solution of the dual problem, and

$$r \geq A^T y - c$$

is a rigorous upper bound for the residual $A^T y - c$. Then $c \geq A^T y - r$, hence

$$c^T x \geq (A^T y - r)^T x = y^T Ax - r^T x = y^T b - r^T x.$$

If we assume rigorous upper bounds

$$x \leq \bar{x},$$

and write

$$r_{\pm} = \max(\pm r, 0), \tag{4}$$

we find

$$c^T x \geq y^T b - r_+^T x \geq y^T b - r_+^T \bar{x}.$$

On computers with directed rounding and switches `roundup` and `rounddown` to select the rounding mode, the following piece of code therefore provides a rigorous lower bound μ for $c^T x$:

$$\begin{aligned} & \text{roundup;} \\ & r = A^T y - c; \\ & \delta = \bar{x}^T r_+; \\ & \text{rounddown;} \\ & \mu = y^T b - \delta; \end{aligned} \tag{5}$$

In exact arithmetic, $r \leq 0$, hence $r_+ = 0$, $\delta = 0$, and $\mu = y^T b$ is the textbook lower bound. In floating point arithmetic, the specified rounding mode guarantees rounding in the direction required by the above derivation.

The general case is a little more complicated. We may assume the LP to be given in the form

$$\begin{aligned} \min \quad & c^T x \\ \text{s.t.} \quad & \underline{b} \leq Ax \leq \bar{b}, \end{aligned} \tag{6}$$

where the bounds \underline{b}, \bar{b} , may contain entries $\pm\infty$ coding for missing bounds. (This includes equality constraints, which arise for $\underline{b}_i = \bar{b}_i$, and bounds on the variables, with sparse rows containing a single nonzero.) We introduce the interval vector

$$\mathbf{b} := [\underline{b}, \bar{b}] = \{\tilde{b} \in \mathbb{R}^n \mid \underline{b} \leq \tilde{b} \leq \bar{b}\},$$

and use interval arithmetic to obtain the required lower bound. Now we assume that rigorous two-sided bounds on x are available,

$$x \in \mathbf{x} = [\underline{x}, \bar{x}].$$

The dual of (6) is

$$\begin{aligned} \max \quad & \underline{b}^T y - \bar{b}^T z \\ \text{s.t.} \quad & A^T(y - z) = c, \quad y \geq 0, \quad z \geq 0. \end{aligned} \tag{7}$$

(Infinite bounds get zero multipliers and are not present in an actual implementation.) From an approximate solution of the dual program we calculate an approximate multiplier $\lambda \approx z - y$, and a rigorous interval enclosure for

$$r := A^T \lambda - c \in \mathbf{r} = [\underline{r}, \bar{r}]. \tag{8}$$

Then

$$c^T x = (A^T \lambda - r)^T x = \lambda^T Ax - r^T x \in \lambda^T \mathbf{b} - \mathbf{r}^T \mathbf{x} \tag{9}$$

and

$$\mu := \inf(\lambda^T \mathbf{b} - \mathbf{r}^T \mathbf{x}) \tag{10}$$

is the desired rigorous lower bound for $c^T x$. In exact arithmetic, $r = 0$, and

$$\mu = \inf((y - z)^T \mathbf{b}) = y^T \underline{b} - z^T \bar{b}$$

is again the textbook lower bound. (Note that if $\underline{b} \neq \bar{b}$ then only one of the corresponding bounds can be active, hence one has $y_i = 0$ or $z_i = 0$.)

Finally, we remark that for rigorous *upper* bounds on $c^T x$, one needs to verify the existence of a feasible point close to the computed point. If all coefficients are integral and the solution is integral or rational with nice denominators, this can be done by suitable rounding and verification of the constraints in integer arithmetic. In the remaining cases, this is considerable more expensive to achieve, since one needs to find an interval enclosure for the solution of a linear system for the active part (and then check the validity of the nonactive inequalities). See NEUMAIER [15, 17] for appropriate techniques in the dense case, and RUMP [19] for the sparse case. Very recent work by JANSSON [9] improves on this in certain cases. Fortunately, rigorous upper bounds are *not* required in the logic of branch-and-cut algorithms.

4 Certificates of infeasibility

At some nodes of a branch tree for a mixed-integer program, the generated linear program may be infeasible. In this case, the dual program (7) has no solution but is unbounded or infeasible, and linear programming solvers typically produce (an approximation to) a vector along a ray exposing this. In the case of (7), this is a pair of vectors y, z satisfying

$$y^T \underline{b} - z^T \bar{b} > 0, \quad A^T(y - z) = 0, \quad y \geq 0, \quad z \geq 0. \quad (12)$$

Given an approximate solution of (12), we form again $\lambda \approx z - y$ and a rigorous interval enclosure for

$$r := A^T \lambda \in \mathbf{r} = [\underline{r}, \bar{r}]. \quad (13)$$

Now for any feasible x ,

$$0 = (r - A^T \lambda)^T x = r^T x - \lambda^T A x \in \mathbf{r}^T \mathbf{x} - \lambda^T \mathbf{b}.$$

Therefore, if

$$d := \inf(\mathbf{r}^T \mathbf{x} - \lambda^T \mathbf{b}) > 0, \quad (14)$$

then it is mathematically certain that no feasible point can exist. We say in this case that λ is a **certificate of infeasibility**, since knowing λ allows one to check the infeasibility using (13) and (14), exhibiting the infeasibility of the linear program (6).

In exact arithmetic, $\mathbf{r} = 0$ and $d = \inf(-\lambda^T \mathbf{b}) = y^T \underline{b} - z^T \bar{b} > 0$, so that λ is a certificate of infeasibility. In finite precision calculations, d can be calculated by outward rounded interval arithmetic or (similar as in section

2) bounded below by directed rounding. If the precision of the calculation is high enough, the computed d is still positive, and λ is again a certificate of infeasibility.

In borderline cases, the computed d will not be positive. This reflects an ill-conditioned situation where the precision of the calculation is not high enough to decide whether the dual is indeed unbounded (or infeasible) and (6) is infeasible, or whether the dual maximum is finite but very huge and there is a feasible point with a huge value of the (primal) objective function.

If the coefficients of the linear program (6) are reasonably bounded, this is possible only if some component of x is huge, and therefore does not occur if \mathbf{x} is of reasonable size. However, if it does occur, then the constraint

$$r^T x = \lambda^T A x = \lambda_+^T A x - \lambda_-^T A x \leq \lambda_+^T \bar{b} - \lambda_-^T \underline{b} =: e \quad (15)$$

tends to be very strong. Indeed, if $e < 0$ then (15) forces some component to be huge, and solving the nearly trivial linear program

$$\begin{aligned} \min \quad & c^T x \\ \text{s.t.} \quad & r^T x \leq e, \quad x \in \mathbf{x} \end{aligned}$$

usually produces a huge lower bound for $c^T x$. Note that a rigorous upper bound for e (which suffices for this) can be computed by directed rounding.

If $e \geq 0$, (15) is weak, and no conclusions can be drawn. In the context of a branch-and-cut method, this is usually not a problem since in this case the node cannot be fathomed, and branching on some integer variable almost always remedies the situation sooner or later.

5 Generating safe cuts

The techniques of the preceding sections provide a rigorous lower bound on the objective function of a linear program. However, if the input coefficients are contaminated by rounding errors, these effect the correctness of the lower bound. It is therefore imperative that it is guaranteed that no errors are introduced at the stage where the linear program is formulated.

In mixed-integer programming, the linear program is usually derived from the original linear constraints by (i) fixing certain integer variables, (ii) restricting the range of certain variables, and (iii) adding suitable cutting planes.

Modifications of type (i) or (ii) introduce no error, and several kinds of cutting planes in use (such as clique cuts, cover cuts, GUB cuts) only use integer arithmetic and hence are safe, too (if their derivation is valid). However, if rounding errors in the computation of the coefficients of a linear inequality defining a cutting plane occur, these may invalidate the cutting plane, cutting off a feasible solution by a tiny margin due to roundoff.

In the remainder of the paper, we look at ways to ensure validity of the cut generating mechanism in finite precision arithmetic in a few key transformations: mixed integer rounding, aggregated inequalities, and generalized Gomory cuts. Precisely the same techniques as for aggregation can be applied to achieve rigor in substitution techniques in a presolve routine. (However, note that substitution of equations into equations yields a two-sided inequality with close but nonequal lower and upper bound, as soon as a rounding error is introduced.)

6 Mixed integer rounding

Both MIR cuts and generalized Gomory cuts may be generated by mixed integer rounding (MARCHAND & WOLSEY [12]). Mixed integer rounding (MIR) is a technique for constructing cuts based on the following (or an equivalent) **rounding lemma**. It exploits the knowledge that some or all the variables involved in an inequality are integers.

6.1 Lemma. *Let $s > 0$, $a \in \mathbb{R}^n$, $\alpha \in \mathbb{R}$, and define*

$$q := \frac{\alpha}{s} - \left\lfloor \frac{\alpha}{s} \right\rfloor \in [0, 1[, \quad \beta := (2q - 1)\alpha + 2sq(1 - q),$$

$$\underline{a} := s \left\lfloor \frac{a}{s} \right\rfloor, \quad \bar{a} := s \left\lceil \frac{a}{s} \right\rceil,$$

$$b := |a - \underline{a} - q(\bar{a} - \underline{a})| + \underline{a} - q(\bar{a} + \underline{a}).$$

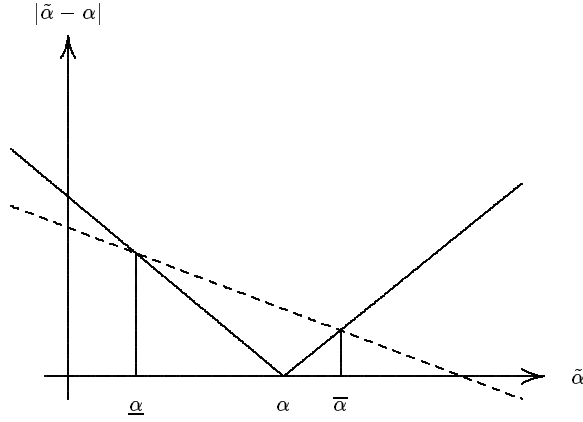
Then

$$0 \leq z \in \mathbb{Z}^n \quad \Rightarrow \quad |a^T z - \alpha| \geq b^T z + \beta. \quad (16)$$

Proof. Denote by z_L the subvector of z indexed by L . For an arbitrary partition of $\{1, \dots, n\}$ into two sets L, U , the number

$$\tilde{\alpha} := \underline{a}_L^T z_L + \bar{a}_U^T z_U$$

Figure 1: Cutting the corner in the graph of $|\tilde{\alpha} - \alpha|$.



is an integral multiple of s . Now the numbers $\underline{\alpha} := s\lfloor\alpha/s\rfloor$ and $\bar{\alpha} := s\lceil\alpha/s\rceil$ are equal or adjacent integral multiples of s , hence $\tilde{\alpha} \notin]\underline{\alpha}, \bar{\alpha}[$. Looking at Figure 1, we see that $|\tilde{\alpha} - \alpha|$ has to be above the dashed line; therefore, $\tilde{\alpha}$ satisfies the inequality

$$|\tilde{\alpha} - \alpha| \geq (1 - 2q)(\tilde{\alpha} - \alpha) + 2sq(1 - q). \quad (17)$$

Using this and the triangle inequality we find

$$\begin{aligned} |a^T z - \alpha| &\geq |\tilde{\alpha} - \alpha| - |(a - \underline{a})_L^T z_L - (\bar{a} - a)_U^T z_U| \\ &\geq (1 - 2q)(\tilde{\alpha} - \alpha) + 2sq(1 - q) - (a - \underline{a})_L^T z_L - (\bar{a} - a)_U^T z_U \\ &= \underline{b}_L z_L + \bar{b}_U z_U + \beta, \end{aligned}$$

where

$$\underline{b} = (2 - 2q)\underline{a} - a, \quad \bar{b} = a - 2q\bar{a}.$$

If we take

$$L = \{k \mid a_k - \underline{a}_k < q(\bar{a}_k - \underline{a}_k)\}, \quad U = \{1 : n\} \setminus L,$$

we find $b_L = \underline{b}_L$ and $b_U = \bar{b}_U$, and the assertion follows. \square

Note that (16) remains valid if the exact value of b and β are replaced by rigorous lower bounds. Calculating these is easy in interval arithmetic; for most of it, directed rounding suffices.

7 Aggregation and MIR cuts

For MIR cuts, an **aggregated constraint** is generated as a linear combination of inequality constraints and then strengthened, utilizing the rounding lemma. If the aggregated constraint is contaminated by roundoff, the MIR cut may be invalid.

To guarantee the validity of the aggregated constraint, we simply shift it by a small amount calculated as follows. We assume that the original constraints have the form

$$\underline{b} \leq Ax \leq \bar{b}, \quad \text{or } Ax \in \mathbf{b} = [\underline{b}, \bar{b}], \quad (18)$$

and that rigorous bounds on the variables,

$$\underline{x} \leq x \leq \bar{x}, \quad \text{or } x \in \mathbf{x} = [\underline{x}, \bar{x}],$$

are available. The aggregated constraint is obtained by multiplying the original constraints by a vector w of weights, where $w_i > 0$ ($w_i < 0$) if the upper (lower) bound of the i th constraint is used. In exact arithmetic, the aggregated constraint takes the form

$$w^T Ax \leq w_+^T \bar{b} - w_-^T \underline{b}.$$

In finite precision arithmetic, both the computed coefficient vector $a \approx A^T w$ and the computed right-hand side may be inaccurate. To account for these inaccuracies we compute an enclosure for the residual

$$r := A^T w - a \in \mathbf{r} = [\underline{r}, \bar{r}] \quad (19)$$

and observe that

$$a^T x = (A^T w - r)^T x = w^T Ax - r^T x \in w^T \mathbf{b} - \mathbf{r}^T \mathbf{x},$$

so that

$$a^T x \leq \gamma := \sup(w^T \mathbf{b} - \mathbf{r}^T \mathbf{x}) \quad (20)$$

is a valid aggregated constraint. γ can be computed by outward rounding interval arithmetic; in exact arithmetic the formula reduces to $\gamma = w_+^T \bar{b} - w_-^T \underline{b}$, as desired. Alternatively, we may sacrifice a little accuracy and proceed as in Section 3 using directed rounding only.

To apply the rounding lemma, we suppose that we have two inequalities

$$e^T x \leq a_1^T z + \varepsilon_1, \quad e^T x \leq a_2^T z + \varepsilon_2, \quad 0 \leq z \in \mathbb{Z}^n. \quad (21)$$

Then

$$\begin{aligned}
e^T x &\leq \min(a_1^T z + \varepsilon_1, a_2^T z + \varepsilon_2) \\
&= \frac{1}{2}((a_1 + a_2)^T z + \varepsilon_1 + \varepsilon_2) - \frac{1}{2}|(a_1 - a_2)^T z + \varepsilon_1 - \varepsilon_2| \\
&\leq d^T z + \delta
\end{aligned} \tag{22}$$

for suitable d and δ , where the final inequality comes from applying the rounding lemma. Since $z \geq 0$, this bound can be made rigorous by using directed (upward) rounding when calculating the coefficients d and δ .

Now suppose that (x^*, z^*) is a fractional point (i.e., one with $z^* \notin \mathbb{Z}^n$). To create an inequality that cuts off z^* , we consider a pair of inequalities (21). In the mixed integer rounding procedure discussed in MARCHAND & WOLSEY [12], (x^*, z^*) is the solution of a previous LP subproblem, and by suitably aggregating active constraints (using some heuristics), the first inequality in (21) is chosen such that equality holds at (x^*, z^*) . Some of the noninteger variables in this inequality may be substituted by the appropriate bounds, or by linear expressions in a single variable only (complementation). To get the second inequality in (21), one puts $a_2 = 0$ and computes ε_2 from available bounds for x . (All this needs to be done with rounding error control, along the same lines as before.) Then the rounding lemma is applied for a suitable $s > 0$, with $a = a_1 - a_2$, $\alpha = \varepsilon_2 - \varepsilon_1$.

To find a suitable value for s we note that, in exact arithmetic, the inequality (22) cuts away (x^*, z^*) iff

$$\Delta(s) := b^T z^* + 2sq(1 - q) - (1 - 2q)\alpha - |a^T z^* - \alpha| \tag{23}$$

is positive (note that b and q depend on s). This suggests that we pick s to make $\Delta(s)$ as large as possible. Natural trial values for s are the discontinuities of the derivatives, which contain the points

$$s \in \left\{ \frac{a_j}{m} \mid z_j^* \notin \mathbb{Z}, m \in \mathbb{N} \right\} \cup \left\{ \frac{\alpha}{2m - 1} \mid m \in \mathbb{N} \right\}.$$

(But the absolute value in b introduces additional discontinuities of the derivatives.)

Usually, a number of trial cuts are explored, and only the successful ones (with a significant value of $\Delta(s)$) are used. Since rounding error control slows down computations a little, it may be profitable to decide upon the cuts to be used using ordinary floating-point arithmetic, and then to repeat the arguments leading to the cuts with rigorous rounding error control. Note that the calculations needed to find a good value of s need not be safeguarded since the bounds are valid for the s actually used, independent of its construction.

8 Generalized Gomory cuts

Instead of using heuristics, one can also proceed systematically as follows, to generate cuts for a MIP with real variables $x \in \mathbb{R}^m$ and nonnegative integer variables $0 \leq z \in \mathbb{Z}^n$. (For variables with upper bounds on integers, one can also use a reduction using complemented variables.)

Let (x^*, z^*) be a solution vertex of the LP relaxation with fractional $z^* \notin \mathbb{Z}^n$. Corresponding to a basis of the LP at the vertex, there is a set of inequalities $Ax + Bz \geq d$ active at the vertex (including equations and active bound constraints other than $z \geq 0$) such that the matrix $(A, B_{:K})$ is square and nonsingular, and $z_k^* = 0$ for all $k \notin K$. Using the factorization available from an active set method for solving the LP, we may solve cheaply the linear system

$$(A, B_{:K})^T y = \begin{pmatrix} 0 \\ a_K \end{pmatrix} \quad (24)$$

for one or more specified integral a_K . By construction,

$$y^T A = 0, \quad \alpha := y^T d = y^T (Ax^* + Bz^*) = y^T Bz^* = a^T z^*,$$

and

$$|y^T Bz - y^T d| = |y^T (Ax + Bz - d)| \leq |y|^T (Ax + Bz - d), \quad (25)$$

and both sides vanish at the vertex. To round (25) into a linear constraint, we choose in the rounding lemma

$$a = B^T y, \quad \alpha = y^T d = a^T z^*, \quad s = 1, \quad (26)$$

to get the linear inequality

$$b^T z + \beta \leq |y|^T (Ax + Bz - d). \quad (27)$$

By construction, a_K is integral, hence $\underline{a}_K = \bar{a}_K = a_K$ and $b_K = (1 - 2q)a_K$. Therefore, (16) reduces for $z = z^*$ to

$$2q(1 - q) = (1 - 2q)\alpha + \beta = (1 - 2q)a_K^T z_K^* + \beta = b^T z^* + \beta \leq 0.$$

Since $q \in [0, 1[$, this is violated except if $q = 0$, i.e., α is integral.

Thus for most integral choices of a , (25) together with the rounding lemma with the choice (26) defines a cut that removes a given nonintegral point z^* from the feasible set. In particular, by choosing for a_K a vector with zeros in all positions except for a single component $a_k = 1$ for some k with nonintegral

z_k^* , we get the traditional Gomory cuts; other choices yield **generalized Gomory cuts**; cf. CERIA et al. [2].

In the presence of rounding errors, Gomory cuts and generalized Gomory cuts are no longer valid, since the arguments in the derivation are based on exact arithmetic. However, by a small amendment of the construction, rigorous cuts approximating the (generalized) Gomory cuts can be constructed as follows.

We choose y as an approximate solution of (24), and use the rounding lemma for $s = 1$, with a and α chosen as approximations to (26). Then the quantities

$$q := A^T y, \quad r := B^T y - a, \quad \delta = \alpha - y^T d$$

are tiny (since they vanish in exact arithmetic), and we have

$$\begin{aligned} b^T z + \beta &\leq |a^T z - \alpha| \\ &= |(y^T A - q^T)x + (y^T B - r^T)z - (y^T d + \delta)| \\ &= |y^T(Ax + Bz - d) - q^T x - r^T z - \delta| \\ &\leq |y|^T(Ax + Bz - d) + \Delta, \end{aligned} \tag{28}$$

provided that Δ is an upper bound for $|q^T x + r^T z + \delta|$. But such a bound can be rigorously obtained using interval arithmetic, if bounds on x and z are given, and the resulting Δ is small if the bounds on x and z are reasonable. Thus (28) gives the rigorous corrected version

$$b^T z + \beta \leq |y|^T(Ax + Bz - d) + \Delta$$

of (27), and this linear inequality can be postprocessed using directed rounding to put it into a standard form.

9 Conclusion

We described an explicit integer linear program of low dimension, where all coefficients and all solution components are small integers, but several state-of-the-art MIP codes claims that no solution exists.

To avoid such errors, it was shown how careful pre- and postprocessing of the input and output of an LP solver can guarantee that no solution is lost in a branch-and-cut framework, at least for mixed-integer programs in which all variables can be bounded rigorously by bounds of reasonable size.

The resulting decisions are rigorously valid independent of the behavior of the LP solver. The safeguards should typically require only a fraction of the work needed for the solution of the linear programs themselves.

References

- [1] A.L. Brearly, G. Mitra and H.P. Williams, An analysis of mathematical programming problems prior to applying the simplex method, *Math. Programming* 8 (1975), 54–83.
- [2] S. Ceria, G. Cornuejols, and M. Dawande, Combining and strengthening Gomory cuts, pp. 438–451 in: *Integer Programming and Combinatorial Optimization* (E. Balas and J. Clausen, eds.), Springer, Berlin 1995.
- [3] J. Czyzyk, M. Mesnier and J. Moré, The NEOS Server, *IEEE J. Comput. Sci. Eng.* 5 (1998), 68–75.
<http://www-neos.mcs.anl.gov/neos/>
- [4] COCONUT, COntinuous COntstraints - Updating the Technology.
<http://www.mat.univie.ac.at/~neum/glopt/coconut/>
- [5] R. Fourer and D.M. Gay, Experience with a primal presolve algorithm, pp. 135–154 in: *Large Scale Optimization: State of the Art*, (W.W. Hager, D.W. Hearn and P.M. Pardalos, eds.), Kluwer, Dordrecht, 1994.
- [6] W. Gropp and J. Moré, Optimization Environments and the NEOS Server, pp. 167–182 in: *Approximation Theory and Optimization*, (M.D. Buhmann and A. Iserles, eds.), Cambridge University Press, Cambridge 1997.
- [7] ILOG CPLEX 7.1 User’s manual, ILOG, France 2001.
- [8] C. Jansson, Zur linearen Optimierung mit unscharfen Daten, Dissertation, Fachbereich Mathematik, Universität Kaiserslautern (1985).
- [9] C. Jansson, Rigorous lower and upper bounds in linear programming, Manuscript, 2002.
- [10] C. Jansson and S.M. Rump, Rigorous solution of linear programming problems with uncertain data, *ZOR – Methods and Models of Operations Research*, 35 (1991), 87–111.

- [11] R. Krawczyk, Fehlerabschätzung bei linearer Optimierung, pp. 215–222 in: Interval Mathematics (K. Nickel, ed.), lecture Notes in Computer Science 29, Springer, Berlin 1975.
- [12] H. Marchand and L.A. Wolsey, Aggregation and mixed integer rounding to solve MIPs, CORE Discussion paper 9839.
<http://www.core.ucl.ac.be/wolsey/mir.ps>
- [13] R.E. Moore, Methods and Applications of Interval Analysis, SIAM, Philadelphia 1981.
- [14] NETLIB Linear Programming Library.
<http://www.netlib.org/lp>
- [15] A. Neumaier, Interval Methods for Systems of Equations, Cambridge Univ. Press, Cambridge 1990.
- [16] A. Neumaier, Introduction to Numerical Analysis, Cambridge Univ. Press, Cambridge 2001
- [17] A. Neumaier, A simple derivation of the Hansen-Blik-Rohn-Ning-Kearfott enclosure for linear interval equations, Reliable Computing 5 (1999), 131–136. Erratum, Reliable Computing 6 (2000), 227.
- [18] F. Ordóñez and R.M. Freund, Computational experience and the explanatory value of condition numbers for linear optimization, MIT Operations Research Center Working paper #OR361-02, submitted to SIAM J. Optimization.
<http://web.mit.edu/rfreund/www/CVfreund.pdf>
- [19] S.M. Rump, Verification methods for dense and sparse systems of equations, pp. 63-136 in: J. Herzberger (ed.), Topics in Validated Computations - Studies in Computational Mathematics, Elsevier, Amsterdam 1994.
- [20] W. Walster, Interval Arithmetic Solves Nonlinear Problems While Providing Guaranteed Results, FORTE TOOLS Feature Stories, WWW-Manuscript, 2001.
<http://www.sun.com/forte/info/features/intervals.html>
- [21] L.A. Wolsey, Integer Programming, Wiley, 1998.