Lexicography and degeneracy: Can a pure cutting plane algorithm work?

Arrigo Zanette[1], Matteo Fischetti[1⋆] and Egon Balas[2⋆⋆]

[1] DEI, University of Padova
[2] Carnegie Mellon University, Pittsburgh, PA

Abstract. We discuss an implementation of the lexicographic version of Gomory's fractional cutting plane method for ILP problems and of two heuristics mimicking the latter. In computational testing on a battery of MIPLIB problems we compare the performance of these variants with that of the standard Gomory algorithm, both in the single-cut and in the multi-cut (rounds of cuts) version, and show that they provide a radical improvement over the standard procedure. In particular, we report the exact solution of ILP instances from MIPLIB such as `stein15`, `stein27`, and `bm23`, for which the standard Gomory cutting plane algorithm is not able to close more than a tiny fraction of the integrality gap. We also offer an explanation for this surprising phenomenon.

Keywords: Cutting Plane Methods, Gomory Cuts, Degeneracy in Linear Programming, Lexicographic Dual Simplex, Computational Analysis.

1 Introduction

Modern Branch-and-Cut (B&C) methods for mixed or pure Integer Linear Programs (ILPs) are heavily based on general-purpose cutting planes such as Gomory cuts, that are used to reduce the number of branching nodes needed to reach optimality; see, e.g., Caprara and Fischetti [7] for a survey on B&C methods. On the other hand, pure cutting plane methods based on Gomory cuts alone are typically not used in practice, due to their poor convergence properties.

In a sense, branching can be viewed as just a "symptomatic cure" to the well-known drawbacks of Gomory cuts—saturation, bad numerical behavior, etc. From the cutting plane point of view, however, the cure is even worse than the disease, in that it hides the real source of the problems. In this respect, it is instructive to observe that a main piece of information about the performance of Gomory cuts (namely, that they perform much better if generated in rounds)

⋆ The work of the first two authors was supported by the Future and Emerging Technologies unit of the EC (IST priority), under contract no. FP6-021235-2 (project "ARRIVAL") and by MiUR, Italy (PRIN 2006 project "Models and algorithms for robust network optimization").

⋆⋆ The work of the third author was supported by National Science Foundation grant ♯DMI-0352885 and Office of Naval Research contract ♯N00014-03-1-0133

was discovered only in 1996 (Balas, Ceria, Cornuéjols, and Natraj [3]), i.e., about 40 years after their introduction [12].

The purpose of our project, whose scope extends well beyond the present paper, is to try to come up with a viable pure cutting plane method (i.e., one that is not knocked out by numerical difficulties), even if on most problems it will not be competitive with the branch-and-bound based methods.

As a first step, we chose to test our ideas on Gomory's fractional cuts, for two reasons: they are the simplest to generate, and they have the property that when expressed in the structural variables, all their coefficients are integer (which makes it easier to work with them and to assess how nice or weird they are). In particular, we addressed the following questions:

i) Given an ILP, which is the most effective way to generate Gomory fractional cuts from the optimal LP tableaux so as to push the LP bound as close as possible to the optimal integer value?
ii) What is the role of degeneracy in Gomory's method?
iii) How can we try to counteract the numerical instability associated with the iterated use of Gomory cuts?
iv) Is the classical polyhedral paradigm "the stronger the cut, the better" still applicable in the context of Gomory cuts read from the tableau? The question is not at all naive, as one has to take into account the negative effects that a stronger yet denser (or numerically less accurate) cut has on the subsequent tableaux, and hence on the next cuts.

As we were in the process of testing various ways of keeping the basis determinant and/or condition number within reasonable limits, our youngest coauthor had the idea of implementing the lexicographic dual simplex algorithm used in one of Gomory's two finite convergence proofs [14]. Gomory himself never advocated the practical use of this method; on the contrary, he stressed that its sole purpose was to simplify one of the two proofs, and that in practice other choice criteria in the pivoting sequence were likely to work better. Actually, we have no information on anybody ever having tried extensively this method in practice.

The lexicographic method has two basic ingredients: (a) the starting tableau is not just optimal, i.e., dual feasible, but lexicographically dual-feasible, and the method of reoptimization after adding a cut is the lexicographic dual simplex method; and (b) at least after every k iterations for some fixed k, the row with the first fractional basic variable is chosen as source row for the next cut.

The implementation of this method produced a huge surprise: the lexicographic method produces a dramatic improvement not only in gap closure (see Figure 1), but also in determinant and cut coefficient size.

It is well known that cutting plane methods work better if the cuts are generated in rounds rather than individually (i.e., if cuts from all fractional variables are added before reoptimization, rather than reoptimizing after every cut). Now it seems that if we are generating rounds of cuts rather than individual cuts, the use of the lexicographic rule would make much less sense, in particular because (b) is automatically satisfied—so the lexicographic rule plays a role only

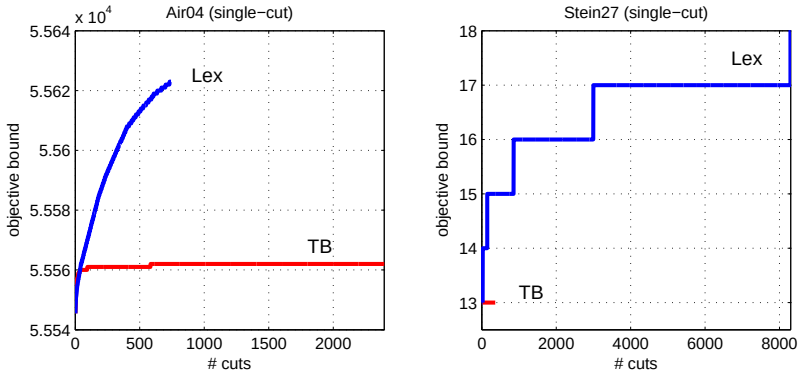

Fig. 1. Comparison between the textbook and lexicographic implementations of single-cut Gomory's algorithm on `air04` and `stein27`.

in shaping the pivoting sequence in the reoptimization process. So we did not expect it to make much of a difference. Here came our second great surprise: as illustrated in Figure 2, even more strikingly than when using single cuts, comparing the standard and lexicographic methods with rounds of cuts shows a huge difference not only in terms of gap closed (which for the lexicographic version is 100% for more than half the instances in our testbed), but also of determinant size and coefficient size (not shown in the figure).

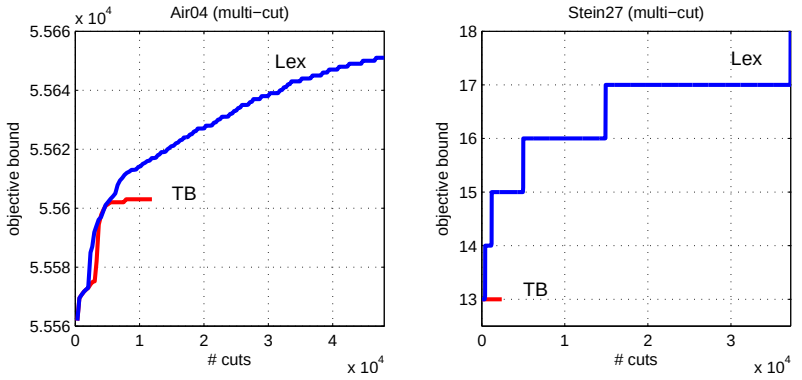

Fig. 2. Comparison between the textbook and lexicographic implementations of multi-cut Gomory's algorithm on `air04` and `stein27`.

In this paper we discuss and evaluate computationally an implementation of the lexicographic version of Gomory's fractional cutting plane method and

of two heuristics mimicking the latter one, and offer an interpretation of the outcome of our experiments.

We also describe a way to round the tableau coefficients when computing GFC coefficients, which turns out to be very effective (together with the lexicographic simplex) in producing numerically stable cuts. The integration of Gomory Mixed-Integer cuts within a lexicographic cutting plane method for pure integer programs is finally addressed.

A preliminary version of the present paper was presented at the 13th International IPCO Conference [19].

2 Gomory cuts

In this paper we focus on pure cutting plane methods applied to solving ILPs of the form:

$$\begin{aligned} \min\ & c^T x \\ & Ax = b \\ & x \geq 0 \text{ integer} \end{aligned}$$

where $A \in \mathbb{Z}^{m \times n}$, $b \in \mathbb{Z}^m$ and $c \in \mathbb{Z}^n$. Let $P := \{x \in \Re^n : Ax = b, x \geq 0\}$ denote the LP relaxation polyhedron, that we assume to be bounded.

The cut generation is of course a crucial step in any cutting plane method, as one is interested in easily-computable yet effective cuts.

In 1958, Gomory [12] (see also [14]) gave a simple and elegant way to generate violated cuts, showing that x^* can always be separated by means of a cut easily derived from a row of the LP-relaxation optimal tableau. The cut derivation is based on a rounding argument: given any equation $\sum_{j=1}^n \gamma_j x_j = \gamma_0$ valid for P, if x is constrained to be nonnegative and integer then $\sum_{j=1}^n \lfloor \gamma_j \rfloor x_j \leq \lfloor \gamma_0 \rfloor$ (as well as $\sum_{j=1}^n \lceil \gamma_j \rceil x_j \geq \lceil \gamma_0 \rceil$) is a valid cut with all-integer coefficients. By subtracting cut $\sum_{j=1}^n \lfloor \gamma_j \rfloor x_j \leq \lfloor \gamma_0 \rfloor$ from $\sum_{j=1}^n \gamma_j x_j = \gamma_0$ one obtains its equivalent "fractional" form $\sum_{j=1}^n \phi(\gamma_j) x_j \geq \phi(\gamma_0)$, where $\phi(z) := z - \lfloor z \rfloor \geq 0$ denotes the fractional part of $z \in \Re$. The cut is typically called *Chvátal-Gomory cut* when written in its all-integer form, and *Gomory Fractional Cut* (GFC) when written in its fractional form; see [9] for a discussion of the relationships between these cuts in a more general context.

According to Gomory's proposal, the cut generating equation is the one associated with a row of the LP optimal tableau whose basic variable is fractional: we will refer to this row as the *cut generating* row, and to the corresponding basic variable as the *cut generating* variable.

GFCs have important theoretical and practical properties. First of all, one can use GFCs read from the LP tableau to derive a finitely-convergent cutting plane method. Secondly, because of the integrality of all the cut coefficients, the associated slack variable can be assumed to be integer, so the addition of GFCs does not introduce continuous variables that could make the rounding argument inapplicable in the next iterations. Moreover, the fact that the cut coefficients

are integer ensures a certain "confidence level" about the numerical accuracy of the generated cuts.

In 1960, Gomory [13] introduced the *Gomory Mixed Integer* (GMI) cuts to deal with the mixed-integer case. In case of pure ILPs, GMI cuts are applicable as well. Actually, GMI cuts turn out to *dominate* GFCs in that each variable x_j receives a coefficient increased by a fractional quantity $\theta_j \in [0, 1)$ with respect to the GFCs (writing the GMI in its $\leq$ form, with the same right-hand-side value as in its GFC counterpart). E.g, a GFC cut of the type $2x_1 - x_2 + 3x_3 \leq 5$ may correspond to the GMI $2.27272727x_1 - x_2 + 3.18181818x_3 \leq 5$. So, from a strictly polyhedral point of view, there is no apparent reason to insist on GFCs when a stronger replacement is readily available at no extra computational effort. However, as shown in the example above, the coefficient integrality of GMI cuts is no longer guaranteed, and the nice numerical properties of GFCs are lost. Even more importantly, as discussed in the sequel, the introduction of "weird fractionalities" in the cut coefficients may have uncontrollable effects on the fractionality of the next LP solution and hence of the associated LP tableau. Finally, GMI cuts introduce continuous slack variables that may receive overweak coefficients in the next iterations, leading to weaker and weaker GMI cuts in the long run. As a result, it is unclear whether GFC or GMI cuts are better suited for a pure cutting plane method for pure integer programs based on tableau cuts.

It is important to stress that the requirement of reading (essentially for free) the cuts directly from the optimal LP tableau makes the Gomory method intrinsically different from a method that works solely with the original polyhedron where the cut separation is decoupled from the LP reoptimization, as in the recent work of Fischetti and Lodi [11] on GFCs or Balas and Saxena [10] on GMI (split) cuts. Actually, only the first round of cuts generated by the Gomory method (those read from the very first optimal tableau) work on the original polyhedron, subsequent rounds are generated from a polyhedron truncated by previously generated cuts.

We face here a very fundamental issue in the design of pure cutting plane methods based of Gomory (mixed-integer or fractional) cuts read from the LP optimal tableau. Since we expect to generate a long sequence of cuts that eventually lead to an optimal integer solution, we have to take into account side effects of the cuts that are typically underestimated when just a few cuts are used (within an enumeration scheme) to improve the LP bound. In particular, one should try to maintain a "clean" optimal tableau so as to favor the generation of "clean" cuts in the next iterations. To this end, it is important to avoid as much as possible generating (and hence cutting) LP optimal vertices with a "weird fractionality"—the main source of numerical inaccuracy. This is because the corresponding optimal LP basis necessarily has a large determinant (needed to describe the fractionality), hence the tableau contains weird entries that lead to weaker and weaker Gomory cuts.

In this respect, dual degeneracy (that is notoriously massive in cutting plane methods) can play an important role and actually can *favor* the practical convergence of the method, provided that it is exploited to choose the *cleanest* LP

solution (and tableau) among the equivalent optimal ones—the sequence of pivots performed by a generic LP solver during the tableau reoptimization is aimed at restoring primal feasibility as quickly as possible, and leads invariably to an uncontrolled growth of the basis determinant, so the method gets out of control after few iterations.

3 Degeneracy and the lexicographic dual simplex method

As already mentioned, massive dual degeneracy occurs almost invariably when solving ILPs by means of cutting plane algorithms. Indeed, cutting planes tend to introduce a huge number of cuts that are almost parallel to the objective function, whose main goal is to prove or to disprove the existence of an integer point with a certain value of the objective function.

In one of his two proofs of convergence, Gomory used the lexicographic dual simplex method to cope with degeneracy. The lexicographic dual simplex method is a generalized version of the simplex algorithm where, instead of considering the minimization of the objective function, viewed without loss of generality as an additional integer variable $x_0 = c^T x$, one is interested in the minimization of the entire solution vector $(x_0, x_1, \ldots, x_n)$, where $(x_0, x_1, \ldots, x_n) <_{LEX} (y_0, y_1, \ldots, y_n)$ means that there exists an index k such that $x_i = y_i$ for all $i = 1, \ldots, k-1$ and $x_k < y_k$. In the lexicographic, as opposed to the usual, dual simplex method the ratio test does not only involve two scalars (reduced cost and pivot candidate) but a column and a scalar. So, its implementation is straightforward, at least in theory. In practice, however, there are a number of major concerns that limit this approach, including the fact that the method rigidly prescribes the pivot choice, thus excluding the possibility of applying much more effective pivot-selection criteria. As a clever approach should not interfere too much with the black-box LP solver used, one could think of using a perturbed linear objective function $x_0 + \epsilon_1 x_1 + \epsilon_2 x_2 \ldots$, where x_0 is the actual objective and $1 \gg \epsilon_1 \gg \epsilon_2 \gg \ldots$. Though this approach is numerically unacceptable, one can mimic it by using the following method which resembles the iterative procedure used in the construction of the so-called Balinsky–Tucker tableau [4], and is akin to the slack fixing used in sequential solution of preemptive linear goal programming (see [2] and [18]).

Starting from the optimal solution $(x_0^\star, x_1^\star, \ldots, x_n^\star)$, we want to find another basic solution for which $x_0 = x_0^\star$ but $x_1 < x_1^\star$ (if any), by exploiting dual degeneracy. So, we fix the variables that are nonbasic (at their bound) and have a nonzero reduced cost. This fixing implies the fixing of the objective function value to $x_0^\star$, but has a major advantage: since we fix only variables at their bounds, the fixed variables will remain out of the basis in all the subsequent steps. Then we reoptimize the LP by using x_1 as the objective function (to be minimized), fix other nonbasic variables, and repeat. The method then keeps optimizing subsequent variables, in lexicographic order, over smaller and smaller dual-degenerate subspaces, until either no degeneracy remains, or all variables are fixed. At this point we can unfix all the fixed variables and restore the origi-

nal objective function, the lex-optimal basis being associated with the non-fixed variables. This approach proved to be quite effective (and stable) in practice.

Lexicographic simplex and Gomory cuts: an entangled pair

The lexicographic simplex method and Gomory cuts form an entangled pair: GFCs and the dual lexicographic simplex are intimately related with each other, in the sense that GFCs are precisely the kind of cuts that allow for a significant lexicographic improvement at each step. It is therefore not surprising that Gomory's (first) proof of convergence strictly relies on the use of the lexicographic dual simplex [14].

In what follows we assume that the tableau rows have been sorted in increasing order of the corresponding basic variables. The lexicographic dual simplex method starts with a lexicographically optimal tableau, which means that all columns are lexicographically positive or lexicographically negative (depending on whether one minimizes or maximizes, and on the sign rule one follows in representing the columns), and preserves this property throughout the pivoting procedure. To fix our ideas, let us opt for minimization and the sign rule that requires all columns to be lexicographically negative, which means that the first entry is the negative of what usually goes under the name of reduced cost. Thus at any stage of the procedure, the first nonzero entry of each column is negative. It is this sign pattern that guarantees a certain property of the sequence of cuts generated under the lexicographic rule, provided that the "right" rounding operation is used in generating the cuts.

As a matter of fact, the Gomory method using the lexicographic simplex can be proved to be convergent only in case the kind of rounding used is consistent with the lexicographic objective. We next discuss very briefly the GFC properties that lead to a convergent method. Let the ith row of the current tableau be

$$x_h + \sum_{j \in J^-} \overline{a}_{ij} x_j + \sum_{j \in J^+} \overline{a}_{ij} x_j = \overline{a}_{i0}$$

where x_h is the basic variable in row i, J^- is the set of indices of nonbasic variables such that $a_{ij} \leq 0$, and J^+ is the set of indices of nonbasic variables such that $\overline{a}_{ij} > 0$. Moreover, let us suppose h is the first index such that $x_h^* (= \overline{a}_{i0})$ is fractional. To simplify notation, assume $h \neq 0$, i.e., the optimal objective value is not fractional.

A key observation is that, due to the lexicographic sign pattern, for each $j \in J^+$ there exist a row $t < i$ with $\overline{a}_{tj} < 0$.

The Gomory rounding procedure can be used to obtain the following GFC, in integer form:

$$x_h + \sum_{j \in J^-} \lceil \overline{a}_{ij} \rceil x_j + \sum_{j \in J^+} \lceil \overline{a}_{ij} \rceil x_j \geq \lceil \overline{a}_{i0} \rceil \tag{1}$$

Note that we round the coefficients of the original row upward. The choice is motivated by the fact that, for a minimization problem, we expect to lexicographically minimize the solution vector of the linear relaxation, hence the cut

is intended to contribute in the opposite direction, namely, to increase lexicographically the solution vector.

Clearly, the round-up operation maintains the nonpositiveness of the coefficients in J^- and the positiveness of those in J^+.

In case no x_j with $j \in J^+$ becomes strictly positive after the lexicographic reoptimization, cut (1) requires

$$x_h \geq \lceil \overline{a}_{i0} \rceil - \sum_{j \in J^-} \lceil \overline{a}_{ij} \rceil x_j \geq \lceil \overline{a}_{i0} \rceil.$$

Otherwise, due to the particular tableau sign pattern, the increase of some x_j with $j \in J^+$ implies the increase of some higher lex-ranked basic variable x_r (the objective function included) by a positive amount. In both cases, a significant lexicographic step is performed: either the cut-generating variable x_h jumps, at least, to its upper integer value $\lceil x_h^* \rceil$, or some higher lex-ranked variable increases by a positive amount. These considerations allow one to conclude that the method converges after a finite number of steps; see [14] for more details.

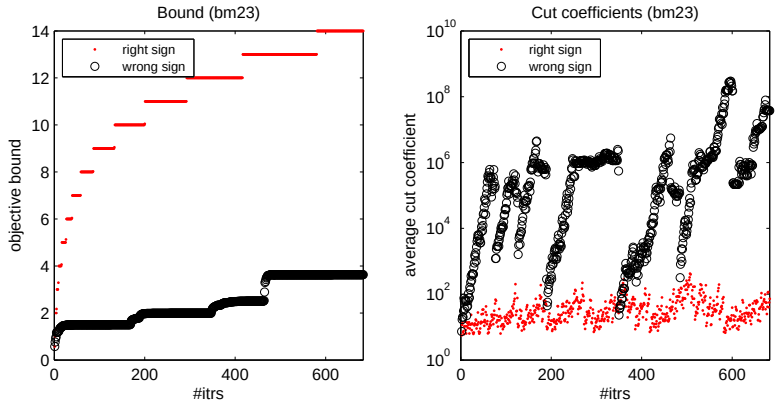

Fig. 3. Impact of rounding direction on GFCs read from the tableau rows.

To show the practical impact of reading the "right" GFC from the tableau rows, in Figure 3 we plot the behavior of two variants of the lexicographic method (in its multi-cut version using rounds of cuts) applied to instance `bm23`: one variant exploits the right GFCs, while the other uses their wrong counterpart. The figure shows a huge difference not only in terms of gap closed, but also of numerical stability (coefficient size).

4 Heuristic variants

While the lexicographic simplex method gives an exact solution to the problem of degeneracy, simple heuristics can be devised that mimic the behavior of lexicographic dual simplex. The scope of these heuristics is to try to highlight the

crucial properties that allow the lexicographic method to produce stable Gomory cuts.

As already mentioned, a lex-optimal solution can in principle be reached by using an appropriate perturbation of the objective function, namely $x_0 + \epsilon_1 x_1 + \ldots + \epsilon_n x_n$ with $1 \gg \epsilon_1 \gg \ldots \gg \epsilon_n$. Although this approach is actually impractical, one can use a 1-level approximation where the perturbation affects a single variable only, say x_h, leading to the new objective function $\min x_0 + \epsilon x_h$. The perturbation term is intended to favor the choice of an equivalent optimal basis closer to the lexicographically optimal one, where the chosen variable x_h is moved towards its lower bound—and hopefully becomes integer.

In our first heuristic, *Heur1*, when the objective function is degenerate we switch our focus to the candidate cut generating variable, i.e., the variable x_i to be perturbed is chosen as the most lex-significant fractional variable. The idea is that each new cut should guarantee a significant lex-decrease in the solution vector by either moving to a new vertex where the cut generating variables becomes integer, or else some other more lex-significant variables becomes fractional and can be cut.

A second perturbation heuristic, *Heur2*, can be designed along the following lines. Consider the addition of a single GFC and the subsequent tableau reoptimization performed by a standard dual simplex method. After the first pivot operation, the slack variable associated with the new cut goes to zero and leaves the basis, and it is unlikely that it will re-enter it in a subsequent step. This however turns out to be undesirable in the long run, since it increases the chances that the GFC generated in the next iterations will involve the slack variables of the previously-generated GFCs, and hence it favors the generation of cuts of higher rank and the propagation of their undesirable characteristics (density, numerical inaccuracy, etc.). By exploiting dual degeneracy, however, one could try to select an equivalent optimal basis that includes the slack variables of the GFCs. This can be achieved by simply giving a small negative cost to the GFC slack variables.

Both the above heuristics involve the use of a small perturbation in the objective function coefficients, that however can produce numerical problems that interfere with our study. So we handled perturbation in a way similar to that used in our implementation of the lexicographic dual simplex, that requires the solution of two LPs—one with the standard objective function, and the second with the second-level objective function and all nonbasic variables having nonzero reduced cost fixed at their bound.

5 Cut validity

Margot [16] addressed the possibility that cutting plane generators produce invalid cuts due to numerical errors, and performed very interesting experiments on this important issue. The outcome of the experiments is that invalid cuts are generated more frequently than expected, so validity should be a major concern in the development of cut generators.

An important step towards a numerical accurate cut generator has been recently taken by Cook et al. [8] for the case of GMI cuts. However, the approach has a potential drawback when used in a pure cutting plane context, as it can produce slightly-modified cuts that do not preserve the nice properties of the original cuts.

When GFCs are considered, validity can be certified by just solving an LP, since these cuts have Chvátal rank 1 with respect to the current formulation. To be more specific, the validity of a GFC of the form $\alpha^T x \leq \alpha_0$ with $(\alpha, \alpha_0) \in \mathbb{Z}^{n+1}$ requires checking whether the maximum value of the LP relaxation with objective function $s := \alpha^T x - \alpha_0$ is strictly less than 1. (GFCs in $\geq$ form can be handled in a similar way.)

In our context, a generic GFC read from the optimal tableau and reexpressed in terms of the structural variables has the form $\sum_{j=1}^{n} \lfloor \overline{a}_{ij} \rfloor x_j \leq \lfloor \overline{a}_{i0} \rfloor$. It is easy to show that the current basis B maximizes the objective function $s := \sum_{j=1}^{n} \lfloor \overline{a}_{ij} \rfloor x_j - \lfloor \overline{a}_{i0} \rfloor$ as well. Indeed, by subtracting from this latter equation the i-th tableau row written as $0 = \sum_{j=1}^{n} \overline{a}_{ij} x_j - \overline{a}_{i0}$ one can project out the basic x variables and obtain the reduced-cost equation

$$s = f_{i0} - \sum_{j=1}^{n} f_{ij} x_j \tag{2}$$

where $f_{ij} \in [0, 1)$ denotes the fractional part of $\overline{a}_{ij}$, and $f_{ij} = 0$ for basic variables x_j. Since $f_{ij} \geq 0$ for all j, the current basic solution x^* is guaranteed to be optimal for the maximization of s, the corresponding optimal value being $s^* = f_{i0} < 1$, which proves the validity of the GFC cut.

In practice, the solution of the maximization problem above will be carried out by a finite-precision solver whose optimality check depends on a certain threshold ϵ used to verify reduced cost signs. From this perspective, it makes sense to assert the validity of a GFC by using the same optimality threshold as in the LP solver, in the sense that cut invalidity would come into play only when the optimality test itself is invalidated by numerical problems. This observation motivates the following definition.

Definition 1. *Given a polyhedron* $P = \{(x, y) \in \mathbb{R}^{n+m}_{+} : \; Ax + y = b\}$ *with* $A \in \mathbb{Z}^{m \times n}$ *and* $b \in \mathbb{Z}^m$*, a basis* B *of* (A, I)*, an integer vector* $(\alpha, \alpha_0) \in \mathbb{Z}^{n+1}$ *and a threshold* $\epsilon > 0$*, we say that cut* $\alpha^T x \leq \alpha_0$ *is a* ϵ-valid (rank 1) cut *if the objective function* $\alpha^T x$ *is* ϵ-maximized *by* B*, according to the classical reduced-cost test* $\alpha^T - \alpha_B^T B^{-1} A \leq \epsilon 1^T$*, and* $\alpha_0 \geq \lfloor \alpha_B^T B^{-1} b + \epsilon \rfloor$.

Accordingly, the ϵ-validity of a GFC cut is still guaranteed if one performs integer roundings by using a small positive threshold ϵ, through operator

$$\text{floor}_\epsilon(\overline{a}_{ij}) = \lfloor \overline{a}_{ij} + \epsilon \rfloor \tag{3}$$

Indeed, with the above redefinition of the $floor$ operator the GFC cut becomes $\sum_j \text{floor}_\epsilon(\overline{a}_{ij}) x_j \leq \text{floor}_\epsilon(\overline{a}_{i0})$ and the reduced costs $f'_{ij} := \overline{a}_{ij} - floor_\epsilon(\overline{a}_{ij}) =$

$f_{ij} - \lfloor f_{ij} + \epsilon \rfloor$ used in (2) to certify GFC validity remain greater or equal to $-\epsilon$, hence within the tolerance that certifies optimality.

In our implementation we use the above $floor_\epsilon$ operator, with ϵ set in a conservative way with respect to the optimality tolerance of the LP solver in use.

6 Computational results

Our set of pure ILP instances mainly comes from MIPLIB 3 and 2003 [5, 1]; see Table 1. It is worth noting that, to our knowledge, even very small instances of these libraries (such as `stein15`, `bm23`, etc.) have never been solved by a pure cutting plane method based on GFC or GMI cuts read from the LP tableau.

Problem	Cons	Vars	LP opt	Opt	% root gap	Source
air04	823	8904	55535.44	56137	1.07	MIPLIB 3.0
air05	426	7195	25877.61	26374	1.88	MIPLIB 3.0
bm23	20	27	20.57	34	39.5	MIPLIB
cap6000	2176	6000	-2451537.33	-2451377	0.01	MIPLIB 3.0
hard_ks100	1	100	-227303.66	-226649	0.29	Single knapsack
hard_ks9	1	9	-20112.98	-19516	3.06	Single knapsack
krob200	200	19900	27347	27768	1.52	2 matching
l152lav	97	1989	4656.36	4722	1.39	MIPLIB
lin318	318	50403	38963.5	39266	0.77	2 matching
lseu	28	89	834.68	1120	25.48	MIPLIB
manna81	6480	3321	-13297	-13164	1.01	MIPLIB 3.0
mitre	2054	9958	114740.52	115155	0.36	MIPLIB 3.0
mzzv11	9499	10240	-22945.24	-21718	5.65	MIPLIB 3.0
mzzv42z	10460	11717	-21623	-20540	5.27	MIPLIB 3.0
p0033	16	33	2520.57	3089	18.4	MIPLIB
p0201	133	201	6875	7615	9.72	MIPLIB 3.0
p0548	176	548	315.29	8691	96.37	MIPLIB 3.0
p2756	755	2756	2688.75	3124	13.93	MIPLIB 3.0
pipex	2	48	773751.06	788263	1.84	MIPLIB
protfold	2112	1835	-41.96	-31	35.35	MIPLIB 3.0
sentoy	30	60	-7839.28	-7772	0.87	MIPLIB
seymour	4944	1372	403.85	423	4.53	MIPLIB 3.0
stein15	35	15	5	9	44.44	MIPLIB
stein27	118	27	13	18	27.78	MIPLIB 3.0
timtab	171	397	28694	764772	96.25	MIPLIB 3.0

Table 1. Our test bed

Possibly after scaling, input data in our testbed is integer. All problems are preprocessed by adding an integer variable x_0 that accounts for the original objective function, from which we can derive valid cuts, as Gomory's proof of convergence prescribes. GFC cuts are derived in integer form. Then slacks are substituted to bring the cut back to the space of original variables. Since all computations deal with integers this avoids round-off errors, and the cuts are ϵ-valid according to Definition 1 above, where $\epsilon = 1e^{-8}$.

We carried out our experiments on a PC Intel Core 2 Q6600, 2.40GHz, with a time limit of 1 hour of CPU time and a memory limit of 2GB for each instance.

Our first set of experiments addressed the *single-cut* version of Gomory's algorithm. Actually, at each iteration we decided to generate *two* GFCs from

the selected cut generating row—one from the tableau row itself, and one from the same row multiplied by -1.

The choice of the cut generation row in case of the lexicographic method is governed by the rule that prescribes the selection of the least-index variable. As to the other methods under comparison, the cut generation row is chosen with a random policy giving a higher probability of selecting the cut-generating variable from those with fractional part closer to 0.5 (alternative rules produced comparable results).

A very important implementation choice concerns the cut purging criterion. The lexicographic algorithm ensures the lexicographic improvement of the solution vector after each reoptimization, thus allowing one to remove cuts as soon as they become slack at the new optimum. As far as other methods are concerned, however, we can safely remove cuts only when the objective function improves. Indeed, if the objective function remains unchanged a removed cut can be generated again in a subsequent iteration, and the entire algorithm can loop—a situation that we actually encountered during our experiments. We therefore decided to remove the slack cuts only when it is mathematically correct, i.e. after a nonzero change in the objective function value, though this policy can lead to an out-of-memory status after a long stalling phase.

Table 2 compares results on the textbook implementation of Gomory's algorithm (TB) and the lexicographic one (Lex). Besides the percentage of closed gap *(ClGap)*, we report 2 tightly correlated parameters to better measure the performance of each method. The first parameter is the cut coefficients size *(Coeff.)*: large coefficients, besides increasing the likelihood of numerical errors, can be a symptom of cut ineffectiveness since they are required to represent very small angles in the space of structural variables. The second parameter is the condition number κ of the optimal basis, which gives a measure of the inaccuracy of the finite-precision representation of a solution x to the linear system $Bx = b$ (the smaller the more accurate the representation). In the table, only the maximum value of the two indicators above during the run is reported. The first column reports one of the following exit-status codes: `(O)` integer optimum, `(T)` time limit, `(M)` out of memory, `(N)` no suitable cuts found as all available cuts where discarded because of their large ($> 10^{10}$) coefficients.

	Textbook						Lex					
Problem		Itrs	Cuts	Time	ClGap	Coeff.	κ		Itrs	Cuts	Time	ClGap	Coeff.	κ
air04	N	267	515	44.95	11.73	8.4e+07	4.6e+12	T	5446	10886	3601.15	**36.83**	1.8e+06	2e+14
air05	N	1237	2391	902.94	2.70	1.5e+08	9.2e+18	T	15859	31712	3601.13	**44.60**	3.2e+05	3.1e+12
bm23	N	313	618	0.23	18.09	1.4e+09	9.2e+18	O	713	1424	0.97	**100.00**	2.4e+02	2.7e+06
cap6000	N	575	1117	55.15	5.36	4.6e+07	2.7e+13	N	12606	25130	2112.15	**26.53**	4.3e+07	8.3e+18
hard_ks100	N	1485	2892	149.03	100.00	2e+08	9.3e+15	O	217	431	0.36	**100.00**	4.5e+05	1.3e+12
hard_ks9	N	164	289	0.18	98.83	2.2e+09	1.2e+13	O	889	1776	0.53	**100.00**	4.8e+04	5.3e+09
krob200	O	41	49	1.40	**100.00**	2.5	1.3e+07	O	1168	2199	256.95	100.00	8.7e+02	4.4e+08
l152lav	N	1099	2160	77.89	37.54	4.2e+08	9.2e+18	O	1122	2219	30.21	**100.00**	1.6e+04	2.1e+09
lin318	O	97	134	8.53	**100.00**	20	5.4e+08	T	2028	3702	3604.78	63.64	2.4e+04	9.7e+10
lseu	N	391	761	0.49	60.75	8.8e+08	9.2e+18	O	15120	30217	33.37	**100.00**	1.9e+04	7.3e+08
manna81	O	271	270	93.41	**100.00**	1	9.2e+18	O	879	878	742.08	100.00	1	9.7e+05
mitre	T	11722	23373	3600.96	**22.76**	1.1e+09	9.2e+18	T	5026	10047	3601.78	14.11	1.3e+06	9.8e+14
mzzv11	T	509	978	3624.01	6.17	6.2e+07	9.2e+18	T	437	870	3604.00	**24.80**	1.2e+02	2.3e+11
mzzv42z	T	1229	2413	3601.25	13.29	1.1e+08	9.2e+18	T	819	1629	3603.23	**43.67**	6.3e+02	5.5e+11
p0033	N	219	434	0.24	10.28	5.1e+08	9.2e+18	O	1496	2957	1.69	**100.00**	1.8e+03	7.6e+06
p0201	N	95	186	0.28	8.11	5.1e+08	2.2e+14	N	188144	373279	2165.83	**85.27**	5.1e+08	1.5e+21
p0548z	T	279514	535298	3601.16	**2.18**	6.5e+08	9.2e+18	T	333934	667812	3601.01	0.03	6.4e+05	1.7e+14
p2756	T	5384	10572	3601.03	0.29	2.1e+09	9.2e+18	T	56113	112222	3601.00	**0.52**	1.2e+04	6.8e+12
pipex	N	3888	7538	1.75	17.01	9.7e+08	2e+14	O	767681	1514866	1402.65	**100.00**	2.7e+06	7.3e+12
protfold	T	112	216	3866.35	5.68	2.5e+08	9.2e+18	T	505	1002	4460.44	**54.38**	1.6e+08	5.6e+14
sentoy	N	21775	36984	9.53	10.82	4.4e+08	9.2e+18	O	5729	11456	14.11	**100.00**	6.5e+04	1.3e+08
seymour	N	210	378	430.49	**21.67**	6.5e+07	9.2e+18	T	779	1538	3605.73	11.23	1.5e+03	2.2e+09
stein15	N	74	123	0.06	25.00	1.9e+09	2.7e+15	O	66	121	0.10	**100.00**	6	2.4e+03
stein27	N	46	87	0.06	0.00	1.1e+09	5.6e+17	O	4283	8254	9.26	**100.00**	76	6.5e+04
timtab1-int	T	99530	172894	3594.30	**22.18**	1.8e+09	9.2e+18	T	785007	1569491	3601.00	4.00	4.9e+08	3.4e+15

Table 2. Comparison between textbook and lexicographic implementation of Gomory's algorithm (single-cut version)

Table 2 shows clearly that in most cases the TB version has huge coefficient sizes and condition numbers, while in Lex all these values remain relatively small along the entire run. While the textbook version stopped in 60% of the cases since it was no longer able to find a cut with coefficients of acceptable size, the lexicographic algorithm encountered this situation in a *single* case. In addition, Lex could solve to proven optimality 12 of the 25 instances of our testbed—some of these instances being notoriously hard for pure cutting plane methods.

For illustration purposes, Figure 4 gives a representation of the trajectory of the LP optimal vertices to be cut (along with a plot of the basis determinant) when the textbook and the lexicographic methods are used for instance `stein15`. In Figures 4(a) and (b), the vertical axis represents the objective function value. As to the XY space, it is a projection of the original 15-dimensional variable space. The projection is obtained by using a standard procedure available e.g. in *MATLAB* (namely, multidimensional scaling [6]) with the aim of preserving the metric of the original 15-dimensional space as much as possible. In particular, the original Euclidean distances tend to be preserved, so points that look close one to each other in the figure are likely to be also close in the original space.

According to Figure 4(a), the textbook method concentrates on cutting points belonging to a small region. This behavior is in a sense a consequence of the efficiency of the underlying LP solver, that has no reason to change the LP solution once it becomes optimal with respect to the original objective function—the standard dual simplex will stop as soon as a feasible point (typically very close to the previous optimal vertex) is reached. As new degenerate vertices are created by the cuts themselves, the textbook method enters a feedback loop that is responsible for the exponential growth of the determinant of the current basis, as reported in Figure 4(d).

On the contrary, as shown in Figure 4(b), the lexicographic method prevents this by always moving the fractional vertex to be cut as far as possible (in the lex-sense) from the previous one. Note that, in principle, this property does not guarantee that there will be no numerical problems, but the method seems to work pretty well in practice.

Finally, Figure 4(c) offers a closer look at the effect of lexicographic reoptimization. Recall that our implementation of the lexicographic dual simplex method involves a sequence of reoptimizations, each of which produces an alternative optimal vertex possibly different from the previous one. As a result, between two consecutive cuts our method internally traces a trajectory of equivalent solutions, hence in the trajectory plotted in Figure 4(b) we can distinguish between two contributions to the movement of x^* after the addition of a new cut: the one due to the black-box optimizer, and the one due to lex-reoptimization. Figure 4(c) concentrates on the slice objective=8 of the Lex trajectory. Each lexicographic optimal vertex used for cut separation is depicted as a filled circle. The immediate next point in the trajectory is the optimal vertex found by the standard black-box dual simplex, whereas the next ones are those contributed by the lexicographic reoptimization. The figure shows that lexicographic reop-

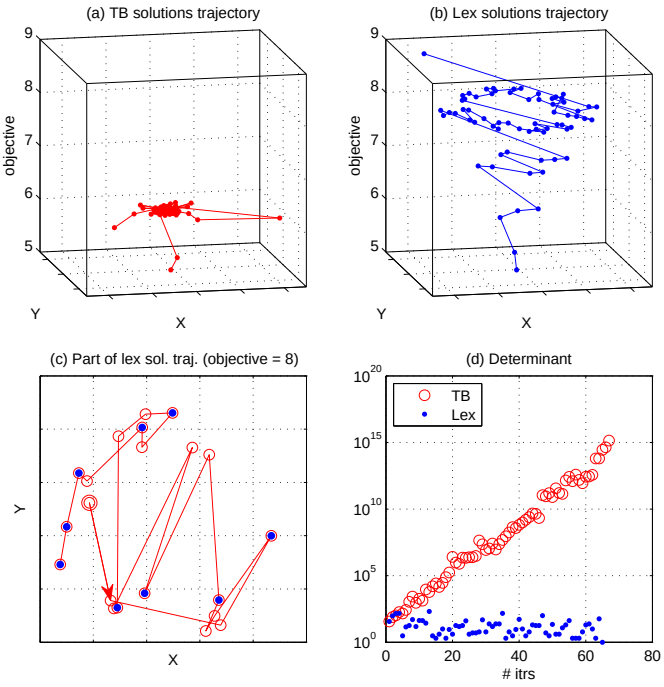

Fig. 4. Problem stein15 (single cut). (a)-(b) Solution trajectories for TB and Lex, resp.; (c) Lower dimensional representation of the Lex solution trajectory; the filled circles are lexicographic optima used for cut separation; their immediate next circles are optima given by the black-box dual-simplex solver, whereas the other points correspond to the equivalent solutions visited during lexicographic reoptimization; the double circle highlights the trajectory starting point. (d) Growth of determinants in TB and Lex (logarithmic scale).

timization has a significant effect in moving the points to be cut, that in some cases are very far from those returned by the black-box dual simplex.

Figure 5 gives an alternative representation of the fractional solutions visited during the optimization of `stein15`. We call it *fractional spectrography*, since it depicts the spectrum of fractionalities along the execution of the algorithm. Fractional spectrography is a pseudo-color plot of the 3-dimensional data set formed by iterations, variables, and variable values. Iterations are represented in the x-axis, variables in the y-axis, while the z-axis, representing variable values, is a grayscale color. Variables are represented in their lexicographic order: variable 0 is the objective function, variable 1 is the lexicographically first variable, x_1, and so on. For a problem with only binary variables, the white color encodes a 1 and the black color a 0. Nonbinary variables as, for example, the objective function, are normalized into the interval $[0, 1]$. All shadings between black and white are fractional values. This coding helps giving a visualization of the degree

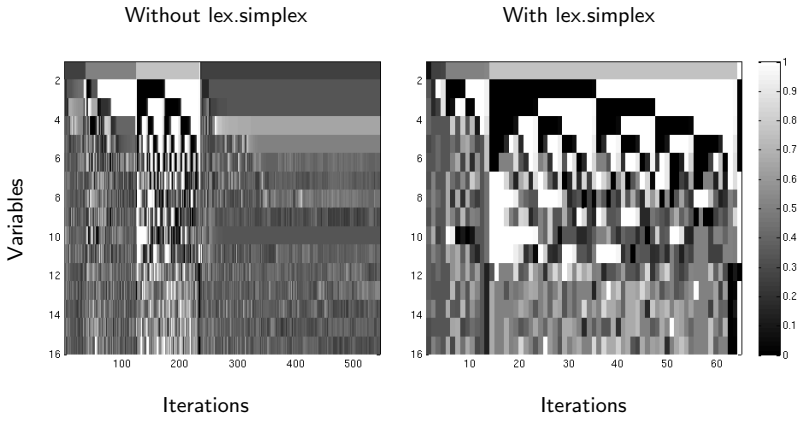

Fig. 5. *Fractional spectrography* of the sequence of solutions provided by the Gomory cutting plane method (one cut at a time).

of fractionality of a solution. In Figure 5, the left chart shows the behavior of the textbook implementation of Gomory cutting planes. It is clear that, from around iteration 250, the method starts visiting fractional solutions that are closer and closer one to each other, until the chart assumes an almost uniform gray shading indicating heavy fractionality persistency. On the contrary, the right-hand-side part of Figure 5 shows the much crisper look of the lexicographic version, where the fractional components tend to flip between integer values and fractionalities are quickly repaired.

To support the interpretation above even further, we performed the experiment of just restarting the LP solver from scratch after having generated the GFCs, so that it is more likely that a "substantially different" optimal solution is found. According to our preliminary tests (not reported here for lack of space), this small change had a significant impact on the performance of the textbook method (though not comparable to that derived from the use of the lexicographic method), showing the importance of breaking the correlation of the optimal LP bases.

A second set of experiments was carried out on the *multi-cut* version of Gomory's algorithm, where cuts are generated in rounds. To be specific, after each LP reoptimization we consider all the tableau rows with fractional basic variable, and generate two GFCs from each row—one from the row itself, and one from the same row multiplied by -1.

The corresponding results are reported in Table 3: the multi-cut version of Lex performed slightly better than in the single-cut mode, as in 13 out of the 26 instances the method reached the optimum. However, the difference between the single- and multi-cut versions is not as striking as one would expect from the experience reported in the literature [3]. This may be due to the fact that, in

the classical non-lexicographic context where cuts are used in conjunction with branching, adding multiple cuts before reoptimizing mitigates the "flattening" tendency of the feasible set in the region where successive individual cuts are applied, leading to a much improved performance. In our pure cutting plane context, instead, the convergence property is theoretically guaranteed by the introduction of a single cut, and the other cuts seem to play a less important role in the long run.

Table 4 reports the results of our two heuristics, *Heur1* and *Heur2*. A comparison with the previous table shows that both heuristics are effective in controlling the coefficient size, determinant, and condition number. The average closed gap is significantly better than in TB, but clearly worse than in Lex.

Figures 6 and 7 give some illustrative plots for instance `sentoy`. The figures clearly show the typical degenerate behavior of TB, with instable phases of rapid growth of determint/coefficients/κ exploring small space regions with shallow cuts. It is worth observing the striking difference in the plots of the *average cut depth*, computed as the geometric distance of the cut from the separated vertex, averaged over all the cuts in a round. Even more interesting, the TB and Lex have a completely different behavior as far as the *optima distance* (computed as the Euclidean distance between two consecutive fractional vertices to be cut) is concerned. As a matter of fact, as already shown by Figure 4, lexicographic reoptimization is quite successful in amplifying the dynamic (and diversity) of the fractional solutions.

	Textbook						Lex					
Problem		Itrs	Cuts	Time	ClGap	Coeff.	κ		Itrs	Cuts	Time	ClGap	Coeff.	κ
air04	N	29	9277	711.79	9.07	8.8e+02	2.8e+09	T	413	133647	3604.53	**24.86**	6.1e+04	9.1e+11
air05	N	43	13342	1061.33	5.32	2.1e+03	2.4e+09	T	776	224190	3600.25	**24.66**	8.9e+03	7.2e+09
bm23	N	107	1348	0.48	18.09	1.6e+09	1.4e+12	O	658	8290	1.11	**100.00**	3.4e+02	2.6e+06
cap6000	N	339	3378	200.75	8.47	3.9e+07	6.9e+15	N	5200	36832	1436.66	**30.27**	3.9e+07	2.7e+18
hard_ks100	N	1752	10689	1037.46	100.00	1.9e+08	4.5e+15	O	105	519	0.29	**100.00**	3.6e+05	1.7e+12
hard_ks9	N	265	1830	0.18	100.00	9.6e+04	1.8e+10	O	139	601	0.11	**100.00**	3e+04	4e+09
krob200	O	101	5029	339.18	100.00	2.4e+02	4.2e+08	O	39	1640	62.86	**100.00**	1.4e+02	1.7e+07
l152lav	N	440	25793	1638.96	32.97	1.1e+03	3.6e+09	O	742	25091	116.45	**100.00**	1.5e+04	1e+09
lin318	O	18	467	22.51	**100.00**	9.3	3.8e+05	O	26	951	106.25	100.00	21	7.9e+06
lseu	N	116	2543	2.78	46.03	6.3e+08	7e+14	O	15662	219216	85.02	**100.00**	1.1e+05	4.7e+10
manna81	O	1	271	4.24	**100.00**	1	1.3e+05	O	10	279	9.68	100.00	1	2.7e+05
mitre	T	154	47919	3798.87	84.80	2.2e+08	1e+18	T	225	71317	3659.66	**87.70**	1.7e+08	1.2e+16
mzzv11	N	20	14379	1940.34	**33.59**	1.2e+04	1.7e+10	T	16	14440	3603.24	31.07	7.2e+05	3.3e+13
mzzv42z	N	20	6887	426.10	**22.62**	1.4e+06	3.7e+12	T	20	15550	3602.99	20.17	3e+07	3.5e+13
p0033	N	368	4511	3.51	95.60	7.8e+08	9.7e+13	O	499	4419	0.94	**100.00**	2.1e+03	1.8e+07
p0201	N	211	5876	23.34	24.19	4.2e+08	1.9e+14	T	176477	4083975	3601.02	**98.24**	3.4e+06	1.5e+15
p0548	N	781	42639	217.06	**52.02**	5.5e+08	2.4e+20	N	1000	51456	46.07	51.78	2.2e+07	1.2e+15
p2756	N	231	9369	274.22	78.63	8.4e+08	1.2e+17	T	22936	326045	3601.11	**79.09**	3.6e+07	1.2e+17
pipex	N	2680	41857	11.67	50.23	6.3e+08	1.9e+13	O	355901	3170373	665.78	**100.00**	4.3e+06	6.2e+13
protfold	T	8	4808	19687.73	8.76	1.3e+08	1.5e+14	T	149	59515	3613.59	**45.26**	15	2.5e+07
sentoy	N	52	707	0.18	3.39	4.7e+08	2.5e+14	O	5348	69208	14.33	**100.00**	9.1e+04	1.4e+10
seymour	N	12	7950	1626.75	16.45	5.1e+07	5.9e+13	T	107	62266	3604.82	**26.89**	3.3e+02	1.2e+08
stein15	N	51	1261	0.37	50.00	1.6e+09	3e+11	O	66	688	0.12	**100.00**	6.3	1.7e+03
stein27	N	40	1821	1.06	0.00	1.2e+09	4.8e+12	O	3132	35859	9.72	**100.00**	77	1.7e+05
timtab1-int	N	148	44479	14.09	28.92	1.4e+08	1.1e+15	N	2982	921030	353.47	**44.20**	1.2e+08	1.1e+16

Table 3. Comparison between textbook and lexicographic implementation of Gomory's algorithm (multi-cut version)

	Heur1						Heur2					
Problem		Itrs	Cuts	Time	ClGap	Coeff.	κ	Itrs	Cuts	Time	ClGap	Coeff.	κ
air04	M	29	9277	711.79	9.07	8.8e+02	2.8e+08	M 26	7671	359.91	**11.90**	2.5e+03	9.1e+10
air05	M	43	13342	1061.33	5.32	2.1e+03	7.5e+09	M 26	7632	399.20	5.32	1.1e+03	3.1e+09
bm23	N	107	1348	0.48	18.09	1.6e+09	1.4e+18	M 660	13828	331.01	**25.54**	2.3e+06	6.9e+14
cap6000	N	339	3378	200.75	8.47	3.9e+07	2.6e+12	M 437	5478	1884.77	**9.10**	1.4e+06	4.5e+14
hard_ks100	N	1752	10689	1037.46	100.00	1.9e+08	1.1e+13	O 3234	6806	432.71	**100.00**	7.1e+03	9.9e+09
hard_ks9	O	265	1830	0.18	**100.00**	9.6e+04	2.3e+05	O 281	2138	0.41	100.00	2.4e+05	5.8e+10
krob200	O	101	5029	339.18	**100.00**	2.4e+02	1.3e+07	O 105	6371	442.80	100.00	2.6e+02	9.6e+07
l152lav	M	440	25793	1638.96	32.97	1.1e+03	1.7e+05	M 200	13912	1118.15	**34.49**	1.2e+04	7.7e+10
lin318	O	18	467	22.51	**100.00**	9.3	5.4e+08	M 29	1559	262.08	99.01	24	1.6e+08
lseu	N	116	2543	2.78	46.03	6.3e+08	9.2e+18	M 520	16394	333.86	**47.78**	2.2e+05	8.3e+12
manna81	O	1	271	4.24	**100.00**	1	9.2e+18	O 1	271	4.39	100.00	1	9.8e+04
mitre	T	154	47919	3798.87	84.80	2.2e+08	9.2e+18	O 153	22616	885.86	**100.00**	5.8e+04	8.1e+13
mzzv11	M	20	14379	1940.34	33.59	1.2e+04	9.2e+18	M 50	21526	2188.58	**40.21**	1.9e+02	1.5e+11
mzzv42z	M	20	6887	426.10	22.62	1.4e+06	9.2e+18	M 70	15777	1593.59	**38.97**	1.3e+03	1.5e+12
p0033	N	368	4511	3.51	95.60	7.8e+08	3.4e+18	O 201	2222	0.41	**100.00**	1.1e+04	6.1e+08
p0201	N	211	5876	23.34	24.19	4.2e+08	9.2e+18	M 670	25540	1462.61	**36.08**	1.1e+06	8.2e+13
p0548	N	781	42639	217.06	**52.02**	5.5e+08	9.2e+18	M 350	22082	1360.48	47.03	3.5e+04	1e+15
p2756	N	231	9369	274.22	**78.63**	8.4e+08	9.2e+18	M 250	15315	1127.85	78.17	5.7e+03	5.7e+12
pipex	N	2680	41857	11.67	**50.23**	6.3e+08	9.2e+18	M 2240	40953	390.15	49.08	1.2e+08	6.7e+14
protfold	N	8	4808	19687.73	8.76	1.3e+08	9.2e+18	T 21	6994	3788.34	8.76	1.6	3.5e+06
sentoy	N	52	707	0.18	3.39	4.7e+08	9.2e+18	M 780	15678	346.80	**19.74**	3.5e+05	9.8e+13
seymour	N	12	7950	1626.75	16.45	5.1e+07	9.2e+18	M 20	9993	2336.60	**21.67**	3.7	2.6e+06
stein15	N	51	1261	0.37	50.00	1.6e+09	1.1e+17	M 500	13772	242.22	50.00	54	6.5e+05
stein27	N	40	1821	1.06	0.00	1.2e+09	9.2e+18	M 280	13604	151.34	0.00	5.8	1.3e+04
timtab1-int	N	148	44479	14.09	28.92	1.4e+08	9.2e+18	M 1320	413451	747.77	**31.95**	1.8e+06	7.3e+14

Table 4. The two heuristics compared (multi-cut version).

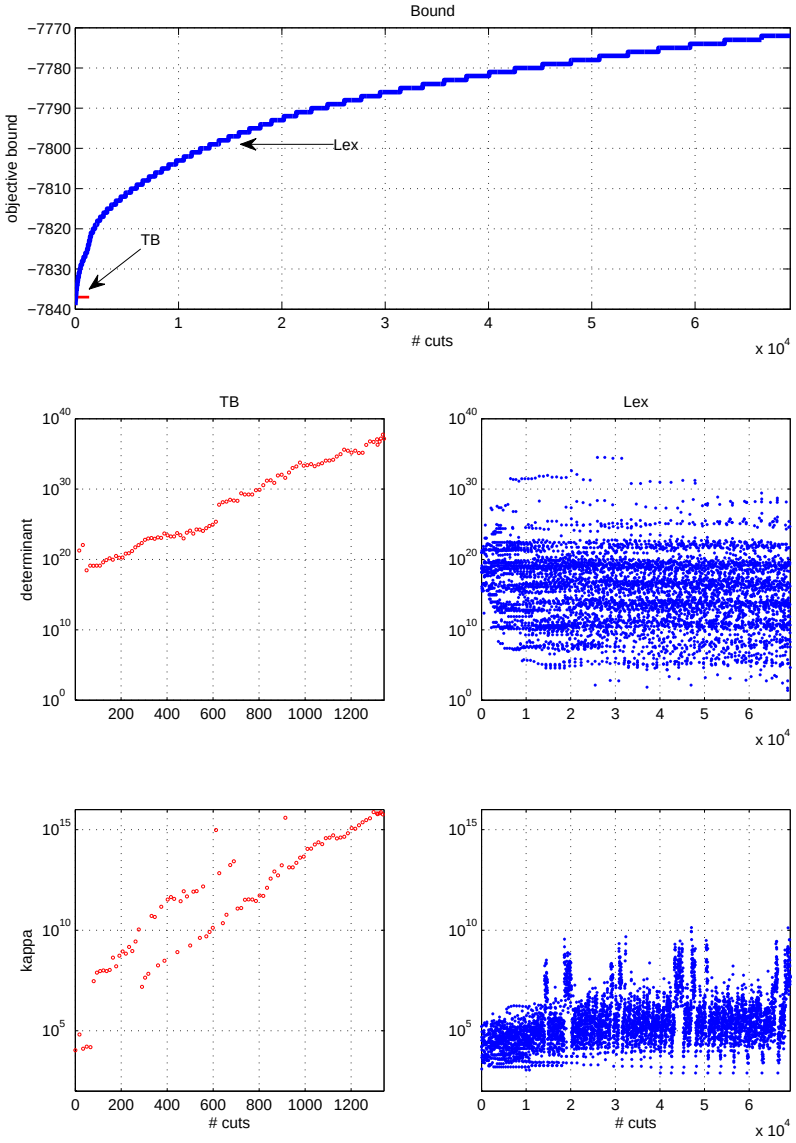

Fig. 6. Comparison between the textbook and lexicographic implementations of the multi-cut algorithm on `sentoy`.

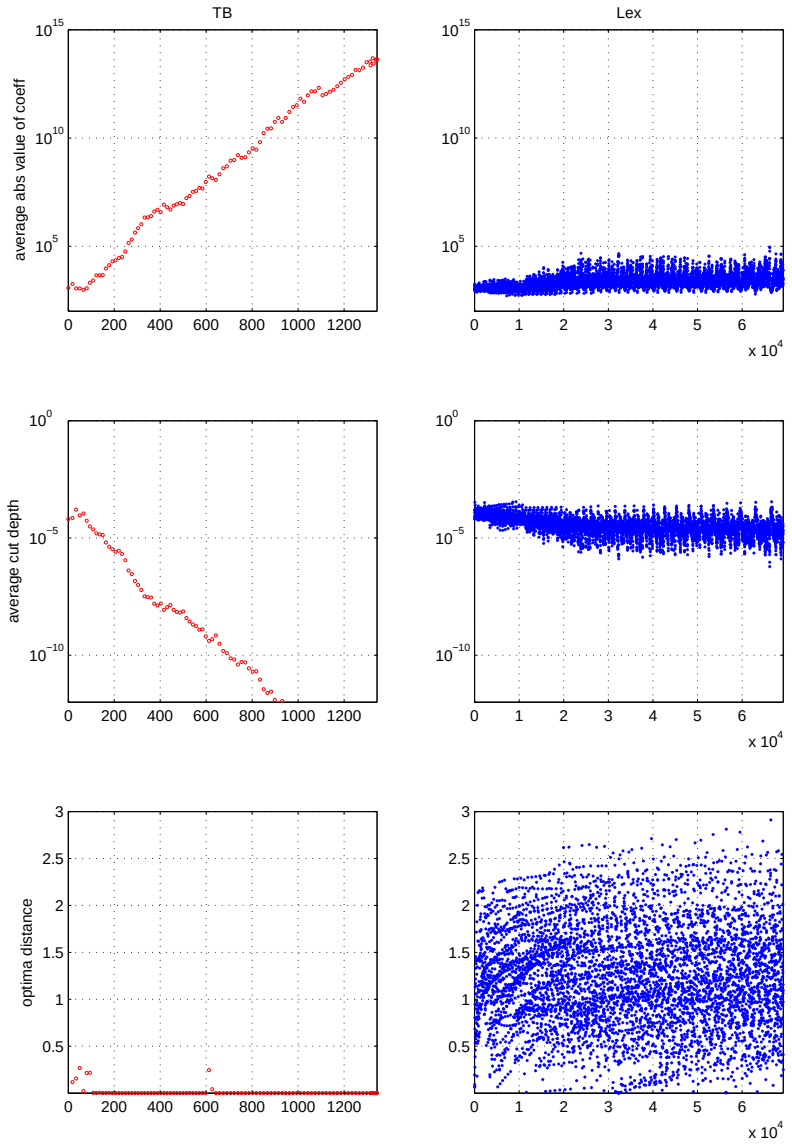

Fig. 7. Comparison between the textbook and lexicographic implementations of the multi-cut algorithm on `sentoy`

7 Approximating GMI cuts for pure integer programs

Using GMI cuts in an all-integer context has the undesired feature of introducing continuous slack variables. This fact can be a problem for two main reasons. First, observations in Section 5 do not hold any more for GMI cuts, since Definition 1 relies on the Chvátal-Gomory rounding argument. Second, fractional values affected by round-off errors are introduced in the all-integer initial formulation, and can significantly contribute to the numerical degradation of the method.

State-of-the-art cut generators use various heuristics to try to ensure the validity of a cut, based on a small weakening of the cut coefficients and right-hand side. However, as already discussed, a general-purpose cut weakening can be a very poor choice in an iterative framework intended to produce long sequences of stable cuts (see Section 5). The challenge here is to weaken GMI cuts in very controlled way, so as to obtain an all-integer counterpart that ensures cut validity (under the assumptions of Section 5) and preserves the strong properties leading to a finitely-convergent cutting plane method. To this end, it is well known [17] that a GFC, in its fractional form, can be obtained by applying the subadditive function $\phi(a_{ij}) = \lceil a_{ij} \rceil - a_{ij} = 1 - f_{ij}$ to the coefficients of row i. Similarly, GMI cuts can be obtained using the subadditive function

$$\phi_{GMI}(a_{ij}) = \phi(a_{ij}) - \delta_j$$

where

$$\delta_j = max \left\{ 0, \ \frac{\phi(a_{ij}) - \phi(a_{i0})}{1 - \phi(a_{i0})} \right\}$$

See Figure 8 for an illustration. So let us rewrite the GMI cut in the following form:

$$\sum_{j \in N \setminus B} (\phi(a_{ij}) - \delta_j) x_j \geq \phi(a_{i0}) \tag{4}$$

Note that this latter cut shares with (1) the sign pattern used (together with the lexicographic property) to prove convergence; see Section 3.

Now, given $\alpha \in \mathbb{Z}_+$ we can approximate (from below) δ_j by replacing it with its discretized counterpart k_j/α, where k_j is the largest positive integer such that $k_j/\alpha \leq \delta_j$. By construction, the resulting d-GMI (d for discretized) cut

$$\sum_{j \in N \setminus B} (\phi(a_{ij}) - \frac{k_j}{\alpha}) x_j \geq \phi(a_{i0}) \tag{5}$$

is a weakening of the GMI cut (hence its validity) that dominates the GFC. In other words, d-GMI cuts play an intermediate role between GMI cuts and GFCs, and they get closer and closer to GMI cuts as parameter α increases. Our procedure might resemble the strengthening procedure of Letchford and Lodi [15]. However the cuts in [15] are substantially different from ours, since d-GMI cuts are a dominated (but hopefully more stable) version of GMIs while there is no strict dominance relationship between GMIs and cuts in [15].

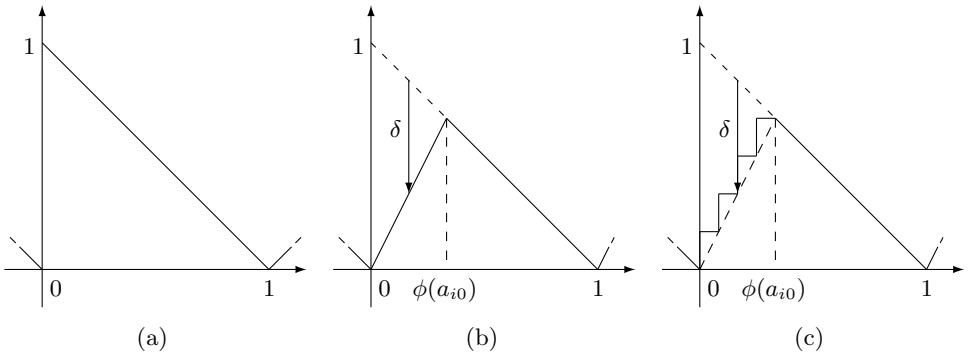

Fig. 8. Different subadditive functions: *(a)* GFC, *(b)* GMI, *(c)* d-GMI

Since α is chosen beforehand, all computations above can be done in integers. In order to derive the cut we can therefore use the numerically more stable rounding function defined in (3). Note that the slack substitution phase needed to add the cut to the previous LP, may involve fractional slacks variables from other cuts. There are two ways of handling this potential source of inaccuracy. The first is to multiply (5) by α so that the cuts added to the problem are all-integer. The second possibility is based on the fact that all slacks are integer multiples of the common fraction $1/\alpha$, so it is easy to compute their rational representation when needed for the slack substitution. In a preliminary computation, the second method exhibited a slightly better numerical stability, so it is the one implemented in our code.

Figure 9 illustrates, for a small problem (`bm23`), the effect of separating d-GMI cuts of different approximation levels within a lexicographic framework, till proved optimality is reached. In the figure on the right-hand side, the x-axis reports approximation error $1/\alpha$ for $\alpha = 1$, 2, 4, 8, and 16, namely 1, 0.5, 0.25, 0.125, and 0.0625 respectively. Tighter approximations were tried but turned out to be numerically unstable, and the run was aborted before convergence. The y-axis reports the number of bits required during the overall process to represent cut coefficients (e.g., to obtain an approximation error of 0.0625, we need approximately 29 bits). It can be seen that, as the approximation gets closer to the real GMI (from left to right in the x-axis), the size of coefficients first grows exponentially, and then tends to saturate along the GMI asymptote.

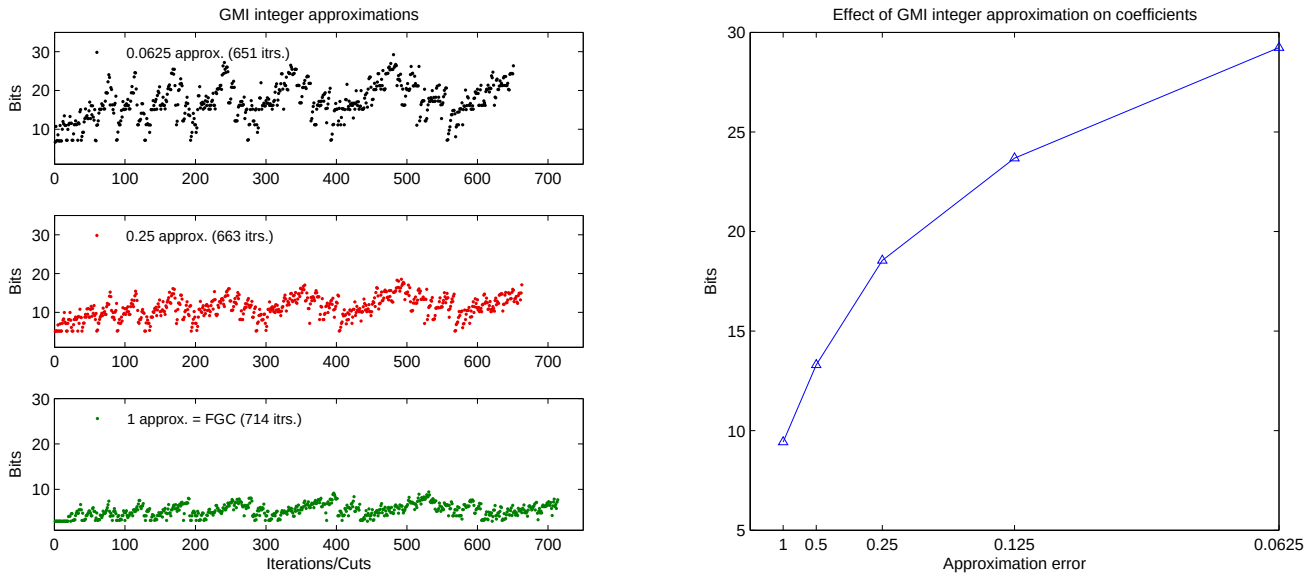

Fig. 9. Gomory's lexicographic method (one cut at a time) using d-GMI cuts on instance `bm23`

Figure 9 (left) illustrates the evolution of the coefficient sizes when moving from GFCs (approximation error 1) to cuts closer and closer to GMI cuts (approximation error 0.25 and 0.0625). On the x-axis we report the iteration (i.e., cut) number, whereas the y-axis gives the maximum number of bits required to represent cut coefficients in each iteration. According to the figure, the three versions with approximation error 1, 0.25 and 0.0625 converge to the optimal solution in 714, 663, and 651 iterations, respectively. This is quite unsatisfactory, as the increased strength of d-GMI cuts reduces the overall number of iterations by, at most, 9%. Moreover, the coefficient fluctuation is highly amplified when the approximation error becomes smaller and smaller. These results were confirmed on other instances, and seem to indicate that the size of coefficients of d-GMI cuts grows exponentially with α, thus making their usage impractical even when cuts are embedded in a lexicographic framework. This also suggests that GMI cuts themselves can be very complex to manage by a pure cutting plane method.

8 Conclusions and future work

Pure cutting plane algorithms have been found not to work in practice because of numerical problems due to the cuts becoming increasingly parallel (a phenomenon accompanied by dual degeneracy), increasing determinant size and condition number, etc. For these reasons, cutting planes are in practice used in cut-and-branch or branch-and-cut mode.

In this paper we have discussed an implementation of the lexicographic version of Gomory's fractional cutting plane method and of two heuristics mimicking the latter one. In computational testing on a battery of MIPLIB problems, we compared the performance of these variants with that of the standard Gomory algorithm, both in the single-cut and in the multi-cut (rounds of cuts) version, and showed that they provide a radical improvement over the standard procedure. In particular, we reported the exact solution of ILP instances from MIPLIB such as `stein15`, `stein27`, and `bm23`, for which the standard Gomory cutting plane algorithm is not able to close more than a tiny fraction of the integrality gap.

We have identified the right choice of direction in rounding the tableau coefficients when generating cuts, which turned out to be very effective (together with the lexicographic simplex) in producing numerically stable cuts.

Future work should address the integration of lexicographic simplex with other kinds of cuts, including GMI cuts. We made a first step in this direction, by introducing a numerically more stable discretized version of GMI cuts. Our preliminary computational results seem however to indicate that these cuts (and hence GMI cuts) are intrinsically more difficult to handle than GFCs, at least within a cutting plane method for pure ILPs.

References

1. T. Achterberg, T. Koch, and A. Martin. MIPLIB 2003. *Operations Research Letters*, 34:361–372, 2006. Problems available at *http://miplib.zib.de*.
2. J. L. Arthur and A. Ravindran. PAGP, a partitioning algorithm for (linear) goal programming problems. *ACM Trans. Math. Softw.*, 6(3):378–386, 1980.
3. E. Balas, S. Ceria, G. Cornuéjols, and N. Natraj. Gomory cuts revisited. *Operations Research Letters*, 19:1–9, 1996.
4. M. L. Balinski and A. W. Tucker. Duality theory of linear programs: A constructive approach with applications. *SIAM Review*, 11(3):347–377, July 1969.
5. R. E. Bixby, S. Ceria, C. M. McZeal, and M. W. P Savelsbergh. An updated mixed integer programming library: MIPLIB 3.0. *Optima*, (58):12–15, June 1998.
6. I. Borg and P.J.F. Groenen. *Modern Multidimensional Scaling: Theory and Applications*. Springer, 2005.
7. A. Caprara and M. Fischetti. Branch-and-cut algorithms. In *Annotated Bibliographies in Combinatorial Optimization*, pages 45–63, 1997.
8. W. Cook, S. Dash, R. Fukasawa, and M. Goycoolea. Numerically safe Gomory Mixed-Integer cuts. *INFORMS Journal on Computing*, 21(4):641–649, 2008.
9. G. Cornuéjols and Y. Li. Elementary closures for integer programs. *Operations Research Letters*, 28:1–8, 2001.
10. Balas E. and A. Saxena. Optimizing over the split closure. *Mathematical Programming A*, 113(2):219–240, 2008.
11. M. Fischetti and Lodi A. Optimizing over the first Chvátal closure. *Mathematical Programming B*, 110(1):3–20, 2007.
12. R. E. Gomory. Outline of an algorithm for integer solutions to linear programs. *Bulletin of the American Society*, 64:275–278, 1958.
13. R. E. Gomory. An algorithm for the mixed integer problem. Technical Report RM-2597, The RAND Cooperation, 1960.
14. R. E. Gomory. An algorithm for integer solutions to linear programs. In R. L. Graves and P. Wolfe, editors, *Recent Advances in Mathematical Programming*, pages 269–302, New York, 1963. McGraw-Hill.
15. A.N. Letchford and A. Lodi. Strengthening Chvátal-Gomory cuts and Gomory Fractional cuts. *Oper. Res. Lett.*, 30(2):74–82, 2002.
16. F. Margot. Testing cut generators for mixed-integer linear programming. *Optima*, 77, 2008.
17. G. Nemhauser and L. Wolsey. *Integer and combinatorial optimization.* Wiley, 1988.
18. M. Tamiz, D. F. Jones, and E. El-Darzi. A review of goal programming and its applications. *Annals of Operations Research*, (1):39–53, 1995.
19. A. Zanette, M. Fischetti, and E. Balas. Can pure cutting plane algorithms work? In *IPCO*, volume 5035 of *Lecture Notes in Computer Science*, pages 416–434. Springer, 2008.