Can pure cutting plane algorithms 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 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 surprizing phenomenon.

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

1 Introduction

Modern branch-and-cut methods for (mixed or pure) Integer Linear Programs 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. 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 troubles. So, a "theoretically convergent" method such as the Gomory one becomes ancillary to enumeration, and no attempt is made to try to push it to its limits. 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 [2]), i.e., about 40 years after their introduction [7].

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 fractional Gomory 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 in the next tableaux, and hence in 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. 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

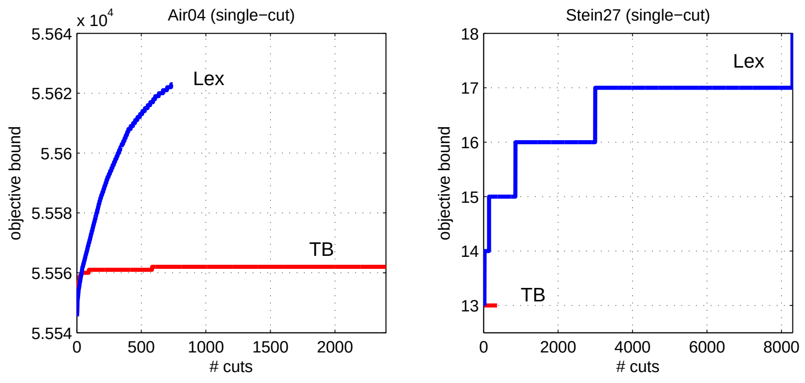

Fig. 1. Comparison between the textbook and lexicographic implementations of single-cut Gomory's algorithm on `air04` and `stein27`.

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 surprize: 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.

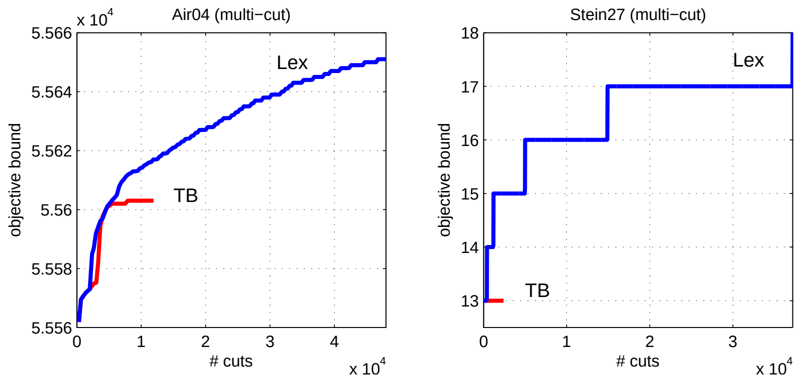

Fig. 2. Comparison between the textbook and lexicographic implementations of multi-cut Gomory's algorithm on `air04` and `stein27`.

In this paper we discuss ad evaluate computationally and 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.

2 Gomory cuts

In this paper we focus on pure cutting plane methods applied to solving Integer Linear Programs (ILPs) of the 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 be bounded.

The structure of a pure cutting plane algorithm for the solution of the ILP problem can be roughly outlined as follows:

1. solve the LP relaxation $\min\{c^T x : x \in P\}$ and denote by x^* an optimal vertex
2. if x^* is integer, then we are done
3. otherwise, search for a *violated cut*, i.e., a hyperplane $\alpha^T x \leq \alpha_0$ separating x^* from the convex hull of integer feasible points, add it to the original formulation, and repeat.

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 [7] (see also [9]) 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 the P, if x is constrained to be nonnegative and integer then $\sum_{j=1}^{n} \lfloor \gamma_j \rfloor x_j \leq \lfloor \gamma_0 \rfloor$ is a valid cut. According to Gomory's proposal, the 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.

The resulting cut, called *Fractional Gomory Cut* (FGC) or Chvátal-Gomory cut, has important theoretical and practical properties. First of all, one can use FGCs 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 FGCs 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. Indeed, once a cut is generated, small fractionalities in the computed coefficients can be removed safely so as to reduce error propagation, whereas FGCs with significant fractionalities are likely to be invalid (because of numerical issues) and hence can be skipped.

In 1960, Gomory [8] 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* FGCs in that each variable x_j receives a coefficient increased by a fractional quantity $\theta_j \in [0, 1)$ with respect to the FGCs (writing the GMI in its $\leq$ form, with the same right-hand-side value as in its FGC counterpart). E.g, a FGC 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 strict polyhedral point of view, there is no apparent reason to insist on FGCs 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, hence the nice numerical properties of FGCs 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. As a result, it is unclear whether FGC or GMI cuts are better suited for a pure cutting plane method based on tableau cuts—a topic that we are going to investigate in the near future.

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 [6] on FGCs or Balas and Saxena [5] 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 (mixed-integer or fractional) Gomory 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 optimized sequence of pivots performed by a generic LP solver during the tableau reoptimization leads invariably to an uncontrolled growth of the basis determinant, and the method gets out of control after few iterations.

3 Degeneracy and the lexicographic dual simplex

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 his proof of convergence, Gomory used the lexicographic dual simplex to cope with degeneracy. The lexicographic dual simplex 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 $(x_0, x_1, \dots, x_n)$, where $(x_0, x_1, \dots, x_n) >_{LEX} (y_0, y_1, \dots, y_n)$ means that there exists an index k such that $x_i = y_i$ for all $i = 1, \dots, k-1$ 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:

1. the ratio test has a worst-case quadratic time complexity in the size of the problem matrix;
2. the ratio test may fail in selecting the right column to preserve lex-optimality, due to round-off errors;
3. the algorithm requires taking control of each single pivot operation, which excludes the possibility of applying much more effective pivot-selection criteria.

The last point is maybe the most important. 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 $x_0 + \epsilon_1 x_1 + \epsilon_2 x_2 \dots$, where x_0 is the actual objective and $1 \gg \epsilon_1 \gg \epsilon_2 \gg \dots$. 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 [3] and is akin to the slack fixing used in sequential solution of preemptive linear goal programming (see [1] and [10]).

Starting from the optimal solution $(x_0^\star, x_1^\star, \dots, x_n^\star)$, 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 original objective function, the lex-optimal basis being associated with the non-fixed variables.

This approach proved to be quite effective (and stable) in practice: even for large problems, where the classical algorithm is painfully slow or even fails, our alternative method requires short computing time to convert the optimal basis into a lexicographically-minimal one. We have to admit however that our current implementation is not perfect, as it requires deciding whether a reduced cost is zero or not: in some (rare) cases, numerical errors lead to a wrong decision that does not yield a lexicographically dual-feasible final tableau. We are confident however that a tighter integration with the underlying LP solver could solve most of the difficulties in our present implementation.

4 Heuristics 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_i, leading to the new objective function $\min x_0 + \epsilon x_i$. 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_i is moved towards its lower bound—and hopefully becomes integer.

In our first heuristic, *Heur1*, when the objective function is degenerate we swap 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 FGC 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 FGC generated in the next iterations will involve the slack variables of the previously-generated FGCs, 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 FGCs. This can be achieved by simply giving a small negative cost to the FGC slack variables.

Both the heuristics above involve the use of a small perturbation in the objective function coefficients, that however can produce numerical troubles 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 Computational results

Our set of pure ILP instances mainly comes from MIPLIB 2003 and MIPLIB 3; 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 FGC or GMI cuts read from the LP tableau.

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

Table 1. Our test bed

Input data is assumed to be 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. Once a FGC is generated, we put it in its all-integer form in the space of the structural variables. In order to control round-off propagation, our FGC separator uses a threshold of 0.1 to test whether a coefficient is integer or not: a coefficient with fractional part smaller than 0.1 is rounded to its nearest integer, whereas cuts with larger fractionalities are viewed as unreliable and hence discarded.

We carried out our experiments in a 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* FGCs 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 3 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 determinant of the optimal basis *(Det.)*. In a sense, the problem being all-integer, the determinant is a measure of the distance from an integer solution: a unit determinant implies an all-integer tableau and solution. Since any coefficient in the tableau can be expressed as a rational number whose denominator is the determinant of the current basis B, the smallest fractional part we could encounter in a tableau is $1/det(B)$—weird tableau entries correspond to large determinants. However, our experiments showed that there are instances with huge determinants (e.g., `mitre`) but numerically quite stable. This is because the size of the determinant is only a weak bound on the degree of fractionality of the solution, as the large denominator can be – and fortunately often is – compensated by a large numerator. A more reliable indicator of numerical precision loss is our third parameter, 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 $Bx = b$.

In the table, only the maximum value of the three indicators above during the run is reported. The first column reports one of the following exit-status codes: `(O)` integer optimum, `(tL)` time limit, `(cL)` limit of 100,000 cuts, `(M)` out of memory, `(E)` numerical errors (either no cuts passed the integrality check, or the problem became infeasible), and `(lE)` if one of the reoptimizations required by the lexicographic method failed for numerical reasons.

A possible failure of the lexicographic method arises when a strict lexicographic improvement is not reached because of numerical issues. In these situations we are no longer protected against the TB drawbacks and we can fail. Precisely, in `sentoy` (single-cut) we failed to improve lexicographically for 27 iterations, in `p0548` for 2597 iterations and in `timtab1-int` for 3 iterations. In multi-cut versions, we failed in `sentoy` for 5 iterations, `p0201` for 57 iterations, in `p0548` for 173 iterations, in `p2756` for 5 iterations, and in `timtab1-int` for 5 iterations.

Table 2 shows clearly that in most cases the TB version has huge coefficient sizes, determinants and condition numbers, while in Lex all these values remain relatively small along the entire run. Moreover, Lex could solve to proven optimality 9 of the 25 instances of our testbed—some of these instances being notoriously hard for pure cutting plane methods.

For illustration purposes, Figure 3 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 3(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 [4]) 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 3(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 incentive in changing 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 3(d).

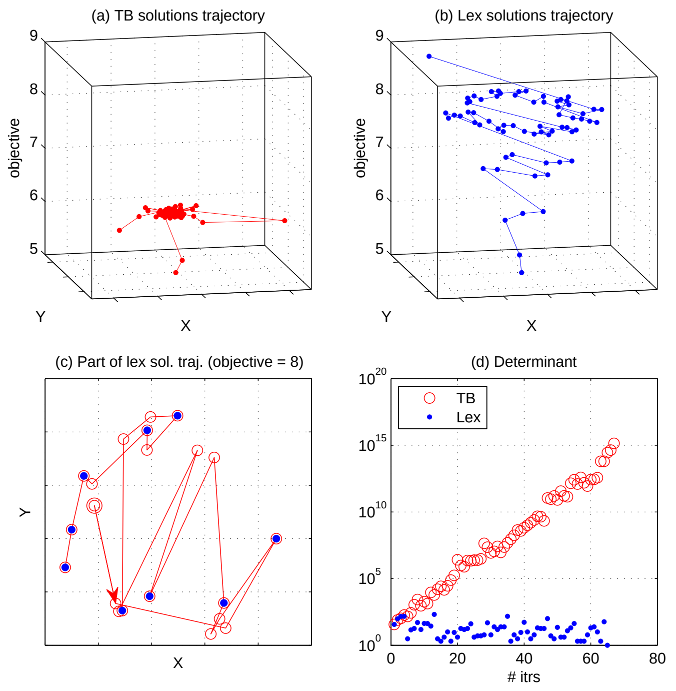

Fig. 3. Problem stein15 (single cut). (a)-(b) Solution trajectories for TB and Lex, resp.; (c) Lower dimensional representation of the the Lex solution trajectory; the filled circles are lexicographic optima used for cut separation; their immediate next circles are optima given by the black-box dual-simplex 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).

On the contrary, as shown in Figures 3(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 be work pretty well in practice.

Finally, Figure 3(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 3(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, an the one due to lex-reoptimization.

Figure 3(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 reoptimization 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.

To support the interpretation above even further, we performed the experiment of just restarting the LP solver from scratch after having generated the FGCs, so that it is more likely that a "substantially different" optimal solution is found. 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.

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.

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 FGCs from each row—one from the row itself, and one from the same row multiplied by -1.

According to Table 3, the multi-cut version of Lex performed even better than in the single-cut mode: in 13 out of the 26 instances the method reached the optimum. Figures 4 and 5 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 3, lexicographic reoptimization is quite successful in amplifying the dynamic (and diversity) of the fractional solutions.

6 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.

Problem	Textbook								Lex							
		Itrs	Cuts	Time	ClGap	Coeff.	Det.	κ		Itrs	Cuts	Time	ClGap	Coeff.	Det.	κ
air04	tL	1197	2394	3602.85	4.42	1.4e+04	1.9e+18	1.8e+13	tL	371	742	3604.92	14.56	1e+04	5.2e+10	5.2e+18
air05	tL	2852	5704	3601.83	4.51	3.8e+05	1e+27	3e+15	tL	1012	2024	3604.92	22.44	4.8e+03	1.9e+10	3.5e+13
bm23	M	5370	5628	2733.43	18.09	1e+15	3.6e+32	2.7e+18	O	713	1426	2.40	100.00	2.4e+02	4.7e+10	1e+10
cap6000	tL	1666	3329	3603.51	8.47	1.5e+10	1.2e+21	9.2e+21	tL	594	1188	3604.17	14.08	2.5e+06	5.1e+09	4.8e+15
hard_ks100	O	3134	6237	1586.82	100.00	8.7e+05	4.1e+08	1.4e+13	O	214	428	1.10	100.00	6.7e+05	6.3e+05	1.5e+14
hard_ks9	O	1058	2107	2.74	100.00	8.3e+14	6.4e+21	1.8e+28	O	889	1778	0.74	100.00	4.8e+04	5.5e+06	1.4e+10
krob200	O	281	532	20.81	100.00	1.2e+02	4.9e+07	3.2e+08	tL	666	1280	3606.81	86.70	2.3e+03	8.6e+07	1e+17
l152lav	tL	3412	6824	3602.10	13.16	6.1e+03	2.7e+09	1.3e+12	O	1152	2278	297.83	100.00	1.7e+04	1.9e+06	5.9e+11
lin318	0	1667	3289	1155.32	100.00	1.6e+03	1.1e+12	5.4e+11	tL	200	346	3618.80	57.69	1.2e+02	1.1e+12	2e+14
lseu	E	4115	8201	671.27	60.75	5.2e+14	7e+31	8.5e+30	O	13541	27068	70.74	100.00	2.4e+04	5.2e+18	2.7e+13
manna81	O	423	423	1338.29	100.00	1	4.9e+173	4.8e+06		81	81	3617.72	9.77	1	1.6e+178	4.8e+06
mitre	tL	6445	12882	3601.11	59.95	1.5e+08	Inf	4.2e+18	tL	77	154	3621.68	2.05	3.2e+03	Inf	1.9e+16
mzzv11	tL	203	406	3618.53	8.98	4.8e+02	6.4e+57	3.4e+12	tL	52	104	3746.52	7.68	61	4.3e+48	8.7e+12
mzzv42z	tL	195	390	3617.33	2.77	8.5e+02	8.3e+53	9.6e+12	tL	25	50	3810.13	5.45	37	7.9e+37	3.7e+14
p0033	E	2919	5824	1054.55	71.15	5.8e+14	1.8e+29	1.1e+31	O	1961	3832	4.22	100.00	2.3e+03	3.5e+17	2.9e+16
p0201	tL	14521	29036	3601.16	13.92	2.6e+11	7.9e+36	7.4e+25	lE	29950	58845	792.35	67.57	2.3e+05	1.2e+22	2.7e+16
p0548z	tL	19279	38446	3601.41	50.11	2.4e+12	Inf	3.4e+28	tL	29199	58216	3603.34	0.03	3.4e+06	2.7e+190	2.4e+16
p2756	tL	5834	11560	3601.75	78.63	5.5e+12	Inf	1e+27	lE	1787	3574	2957.59	0.52	9.1e+02	1.1e+278	5.4e+15
pipex	E	4719	9408	24.84	36.26	7.5e+14	8.1e+26	3.1e+27	cL	50000	99390	146.86	42.43	1.1e+05	3e+10	3.4e+21
protfold	tL	68	136	3811.43	8.76	5.2e+10	Inf	2.1e+17	tL	299	598	3606.09	45.26	30	1.3e+30	2.1e+07
sentoy	E	830	1639	15.28	3.39	4.3e+14	2.1e+37	3e+28	O	5771	11541	47.88	100.00	6.5e+04	4.6e+34	1.5e+16
seymour	tL	124	246	3768.72	6.01	4.7e+08	5e+56	9.1e+20	tL	94	182	3618.15	11.23	15	1.2e+19	1.6e+08
stein15	E	173	313	2.29	12.39	2.1e+16	Inf	2.5e+20	O	65	121	0.18	100.00	17	3.6e+02	1.8e+11
stein27	E	208	365	3.59	0	1.5e+16	Inf	7e+21	O	4298	8301	45.23	100.00	7.2e+02	8.9e+04	1.2e+06
timtab1-int	tL	10784	21501	3603.58	11.94	1.7e+13	Inf	1.2e+27	cL	50000	99887	3124.58	4.00	1.6e+07	6e+250	1.5e+21

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

Problem	Textbook								Lex							
		Itrs	Cuts	Time	ClGap	Coeff.	Det.	κ		Itrs	Cuts	Time	ClGap	Coeff.	Det.	κ
air04	M	35	11912	1209.41	11.23	2.4e+03	6.9e+11	9.3e+09	tL	150	47995	3606.61	**19.21**	7.3e+03	2.2e+09	3.9e+11
air05	M	30	9648	955.09	5.11	7.1e+02	4.2e+11	4.9e+09	tL	261	76263	3598.20	**14.00**	1.5e+03	4.9e+10	9.6e+10
bm23	E	144	2168	9.33	5.28	2.3e+15	5.5e+30	8.7e+24	O	659	8298	1.67	**100.00**	3.7e+02	4.6e+10	3e+14
cap6000	tL	979	11693	3614.59	10.34	1.4e+09	7e+15	4.3e+20	E	966	6769	2110.12	**24.66**	4.5e+06	5.1e+09	1.6e+15
hard_ks100	tL	1950	10770	3603.03	**100.00**	5e+04	2.2e+05	1.5e+12	O	98	439	0.63	**100.00**	5e+05	1.5e+04	2.8e+09
hard_ks9	O	269	1678	0.30	**100.00**	7.8e+04	1.7e+05	6.9e+09	O	139	614	0.12	**100.00**	3e+04	3.6e+04	2.9e+10
krob200	O	44	2017	153.92	**100.00**	1.6e+02	1e+06	9.4e+07	O	17	282	22.18	**100.00**	7.2	1e+06	3.7e+04
l152lav	M	525	31177	2715.78	40.59	5.3e+03	8e+08	1.6e+10	O	681	22364	206.82	**100.00**	1.8e+04	6.7e+05	2.4e+10
lin318	O	15	250	12.82	**100.00**	7	3.4e+07	9.7e+05	O	19	416	124.27	**100.00**	17	3.2e+07	1.8e+08
lseu	E	149	3710	34.35	44.58	1.7e+14	9.8e+33	2e+19	O	1330	18925	6.62	**100.00**	9.6e+03	1.2e+14	2.6e+14
manna81	O	1	270	8.88	**100.00**	1	5.2e+92	3.7e+06	O	2	280	29.58	**100.00**	1	1.7e+97	3.7e+06
mitre	M	94	30016	1262.24	85.52	8.9e+08	Inf	4.2e+20	tL	232	20972	3619.30	**97.83**	6.2e+06	Inf	9.1e+13
mzzv11	M	33	17936	1735.42	38.56	4.9e+02	Inf	2.6e+11	tL	61	25542	3739.41	**40.60**	1.8e+02	2.8e+51	8.5e+11
mzzv42z	M	62	14664	1515.50	33.80	3e+06	Inf	6.5e+14	tL	61	14830	3616.85	**25.50**	8.5e+02	1e+38	8.7e+12
p0033	M	1529	23328	2493.35	81.53	1.8e+14	1.1e+34	2.2e+24	O	87	1085	0.14	**100.00**	4.5e+02	1.6e+13	6e+14
p0201	tL	1332	55408	3602.21	19.32	2e+12	2e+34	2.5e+26	E	27574	629004	1119.71	**74.32**	2.3e+05	3.6e+22	7.4e+11
p0548	M	825	35944	2223.35	**48.98**	1.3e+09	2.4e+137	2e+24	E	461	27067	131.69	47.50	4.8e+06	7.1e+135	5.5e+14
p2756	M	740	19925	2423.48	78.86	4.5e+12	Inf	1.3e+27	E	642	14645	509.72	**79.09**	2.4e+05	Inf	1.1e+13
pipex	M	2929	49391	1921.78	51.25	1.3e+14	3.3e+27	4e+28	O	50000	488285	188.11	**74.18**	2.8e+05	8.5e+11	1e+14
protfold	M	7	5526	1058.84	8.76	4.6e+06	Inf	5.1e+13	tL	159	38714	3607.08	**45.26**	30	1.5e+32	1e+07
sentoy	M	1953	34503	2586.38	18.25	5.1e+14	5.3e+43	7e+30	O	5331	68827	25.85	**100.00**	7.6e+04	3.2e+34	3.9e+14
seymour	tL	18	11748	7517.40	21.67	3.3e+08	1.3e+159	1.4e+16	tL	87	48339	3610.62	**26.89**	78	1e+23	3.6e+07
stein15	E	116	3265	9.50	20.92	2.5e+15	2.4e+26	3.2e+28	O	64	676	0.14	**100.00**	15	2e+02	2.3e+06
stein27	E	57	2399	20.01	0.00	1.7e+15	2.8e+36	8.3e+18	O	3175	37180	19.72	**100.00**	7.5e+02	1.9e+04	1.9e+05
timtab1-int	M	231	73188	585.72	23.59	1.6e+11	Inf	2.1e+21	cL	3165	1000191	841.56	**50.76**	3.5e+06	Inf	1.7e+16

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

	Heur1								Heur2							
Problem		Itrs	Cuts	Time	ClGap	Coeff.	Det.	κ		Itrs	Cuts	Time	ClGap	Coeff.	Det.	κ
air04	tL	1886	3772	3602.45	6.24	7.9e+04	6.5e+27	1.5e+15	tL	1463	2926	3601.57	12.06	1.3e+07	7e+15	5.2e+18
air05	tL	1355	2710	3630.37	4.11	1.1e+10	1.6e+51	4.1e+21	tL	1338	2676	3603.34	4.51	1.1e+05	8.6e+26	3.5e+13
bm23	tL	5760	6111	3602.69	25.54	1.7e+15	2.2e+29	1.7e+20	tL	3938	7876	3602.32	25.54	1.1e+04	2.1e+15	1e+10
cap6000	tL	2024	4048	3601.36	7.85	2.8e+09	6.5e+20	9.6e+20	tL	1379	2757	3605.44	8.47	4.5e+06	3.8e+13	4.8e+15
hard_ks100	O	2212	4403	742.51	100.00	4.9e+11	3.3e+21	1.7e+24	0	627	1251	11.10	100.00	6.4e+05	5.4e+06	1.5e+14
hard_ks9	O	839	1674	0.67	100.00	1.6e+05	1.6e+07	2.5e+11	O	162	322	0.10	100.00	5.4e+05	5.1e+05	1.4e+10
krob200	tL	3009	5978	3602.40	92.40	1.1e+03	1e+16	8.2e+11	M	3051	6066	3544.19	97.15	7.5e+05	1.3e+13	1e+17
l152lav	tL	5910	11813	3601.22	46.68	2.7e+05	4.6e+11	6.4e+13	tL	2071	4140	3604.68	32.97	2e+04	8.2e+16	5.9e+11
lin318	M	1699	3360	2760.11	66.94	5.6e+03	1.2e+14	6.1e+12	M	1002	1966	1203.28	88.43	2.1e+04	2.1e+13	2e+14
lseu	tL	17687	35367	3601.67	69.16	7.3e+11	7.7e+32	1.8e+24	tL	3805	7608	3601.04	47.78	2.5e+05	1.6e+24	2.7e+13
manna81	O	425	425	1329.69	100.00	1	4.9e+173	4.8e+06	O	423	423	1328.05	100.00	1	4.9e+173	4.8e+06
mitre	tL	6003	12003	3601.02	47.89	2.3e+05	Inf	1.6e+14	tL	5973	11934	3601.13	75.15	3e+06	Inf	1.9e+16
mzzv11	tL	69	138	3623.00	4.44	1.7e+02	Inf	5.4e+12	tL	267	532	3612.87	13.45	8.2e+02	2.8e+54	8.7e+12
mzzv42z	tL	125	250	3667.32	5.72	3.2e+02	1.2e+65	2.9e+12	tL	247	494	3601.63	8.96	9.3e+03	1.9e+54	3.7e+14
p0033	O	20667	41287	486.85	100.00	3.3e+08	7.6e+20	1.8e+19	tL	14143	28267	3603.52	94.55	1.3e+08	4e+17	2.9e+16
p0201	tL	10756	21510	3602.02	21.22	3.6e+09	4e+32	8.7e+21	tL	3689	7378	3601.73	17.16	7.6e+06	3.3e+21	2.7e+16
p0548	tL	22096	44063	3601.44	50.61	1.5e+06	4.7e+190	1.4e+18	tL	9523	19007	3603.13	43.19	5.8e+06	1e+190	2.4e+16
p2756	tL	4147	8191	3601.76	78.63	5.6e+10	Inf	1.1e+21	tL	4623	9182	3604.52	77.25	3.8e+04	Inf	5.4e+15
pipex	E	50000	99985	650.65	47.51	1.4e+08	1.8e+19	1.2e+17	tL	26620	53233	3601.96	40.68	1.4e+11	2e+19	3.4e+21
protfold	tL	366	732	3617.67	17.88	4.6	9.6e+37	9.8e+09	tL	435	870	3610.08	8.76	1.5	2.4e+28	2.1e+07
sentoy	E	1184	2357	116.95	7.85	3.9e+14	2.5e+38	1e+29	tL	6951	13900	3602.91	22.71	3.3e+06	8.6e+30	1.5e+16
seymour	tL	201	401	3622.41	6.01	9.1e+02	4.6e+26	1.8e+12	tL	275	548	3615.12	11.23	17	4.9e+17	1.6e+08
stein15	tL	5592	11145	3601.33	75.00	1.2e+03	1.5e+11	1e+09	tL	4148	8293	3602.46	50.00	4.9e+04	1.3e+15	1.8e+11
stein27	tL	3798	7595	3601.57	0.00	1.4e+02	1.1e+11	1.6e+08	tL	3645	7290	3601.92	0.00	6	1.9e+05	1.2e+06
timtab1-int	cL	50000	99997	1676.40	16.02	1.3e+07	Inf	4.3e+18	tL	21474	42946	3601.19	14.50	1.1e+09	Inf	1.5e+21

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

					Heur1								Heur2			
Problem		Itrs	Cuts	Time	ClGap	Coeff.	Det.	κ		Itrs	Cuts	Time	ClGap	Coeff.	Det.	κ
air04	M	30	10250	1287.10	10.40	1.4e+04	2.1e+10	1.8e+12	M	28	9075	511.50	9.73	2.6e+03	2.1e+08	3.9e+11
air05	M	49	15805	1365.48	6.53	3.1e+04	4.1e+24	2.1e+11	M	26	8288	667.99	5.11	1.8e+03	3.5e+10	9.6e+10
bm23	M	1819	8403	3112.35	18.09	5e+14	1.7e+29	6.7e+22	M	631	13800	1933.80	25.54	4e+06	3.5e+23	3e+14
cap6000	tL	484	7002	3605.86	7.23	3.5e+08	4e+16	3.1e+17	M	559	7511	2498.47	9.72	1.5e+08	8e+14	1.6e+15
hard_ks100	tL	150	672	1.24	100.00	1.2e+04	3.3e+04	1.8e+10	O	810	1642	28.45	100.00	3.5e+03	3e+03	2.8e+09
hard_ks9	O	251	1641	0.33	100.00	5.5e+04	2.6e+05	8.3e+10	O	252	2010	1.18	100.00	4.9e+05	2.7e+08	2.9e+10
krob200	O	12	108	3.15	100.00	1.2	1e+06	1.4e+05	O	9	81	2.27	100.00	1.2	1e+06	3.7e+04
l152lav	tL	627	30047	3639.62	46.68	1.4e+09	2.3e+24	5.4e+15	M	106	9889	1034.03	31.44	6.7e+03	1.1e+10	2.4e+10
lin318	O	42	2415	563.91	100.00	1e+02	4.7e+09	5.6e+08	O	35	1503	238.84	100.00	55	7.6e+08	1.8e+08
lseu	M	2604	54498	3388.00	70.21	2.9e+12	2.1e+32	4.8e+19	M	542	16610	1433.12	48.48	8e+06	5.4e+23	2.6e+14
manna81	O	1	270	8.94	100.00	1	5.2e+92	3.7e+06	O	1	270	8.83	100.00	1	5.2e+92	3.7e+06
mitre	tL	148	47954	3651.05	88.90	5.4e+07	Inf	1.1e+17	tL	192	27786	3624.63	91.56	3.4e+04	Inf	9.1e+13
mzzv11	tL	14	10452	5544.73	29.27	8.9e+03	4.9e+73	4.4e+15	M	40	15727	1699.51	34.16	3.5e+02	2.9e+43	8.5e+11
mzzv42z	M	18	6818	2613.82	18.28	2.6e+04	1.2e+52	3.4e+13	M	74	17099	2423.77	28.44	1.5e+03	1.4e+41	8.7e+12
p0033	O	40	460	0.06	100.00	2.5e+02	1.5e+13	4.1e+06	M	2060	37057	3190.24	91.03	3e+07	4.9e+16	6e+14
p0201	tL	2087	92066	3612.42	51.22	1.1e+09	1.8e+26	8.8e+20	M	756	26159	2476.90	37.57	6.9e+04	1.6e+15	7.4e+11
p0548	tL	1401	90042	3601.77	53.75	4.7e+07	2.4e+137	1.9e+16	M	358	24003	1612.11	46.09	2.7e+04	5.1e+133	5.5e+14
p2756	tL	420	17108	3613.50	78.86	8e+12	Inf	3.1e+27	M	283	14402	1852.05	78.40	4.8e+04	Inf	1.1e+13
pipex	E	50000	552448	752.66	55.44	5.3e+09	1e+20	1.5e+20	M	1530	30972	1608.28	48.98	2.2e+07	1.9e+16	1e+14
protfold	tL	19	7886	3775.30	27.01	6.9	6.5e+28	1.2e+09	tL	23	8488	3674.48	8.76	1.4	2.6e+24	1e+07
sentoy	E	118	1712	5.06	4.88	5.1e+14	1.3e+36	1.1e+20	M	1064	20822	2111.59	22.71	7.9e+05	3.2e+29	3.9e+14
seymour	M	19	12084	3081.48	21.67	1.3e+02	1.1e+31	1.6e+11	M	30	16270	3494.90	26.89	8.8	2.9e+17	3.6e+07
stein15	M	985	20707	2133.91	75.00	1.9e+03	1.1e+11	1.5e+09	M	505	13838	1585.78	50.00	44	3.7e+05	2.3e+06
stein27	M	283	13587	937.55	0.00	39	9.8e+06	4.2e+06	M	274	13586	936.43	0.00	4.7	1.8e+05	1.9e+05
timtab1-int	M	383	119945	288.72	37.10	6.2e+06	3.9e+216	3.1e+13	M	688	197581	733.33	43.08	1.4e+07	8.1e+198	1.7e+16

Table 5. The two heuristics compared (multi-cut version).

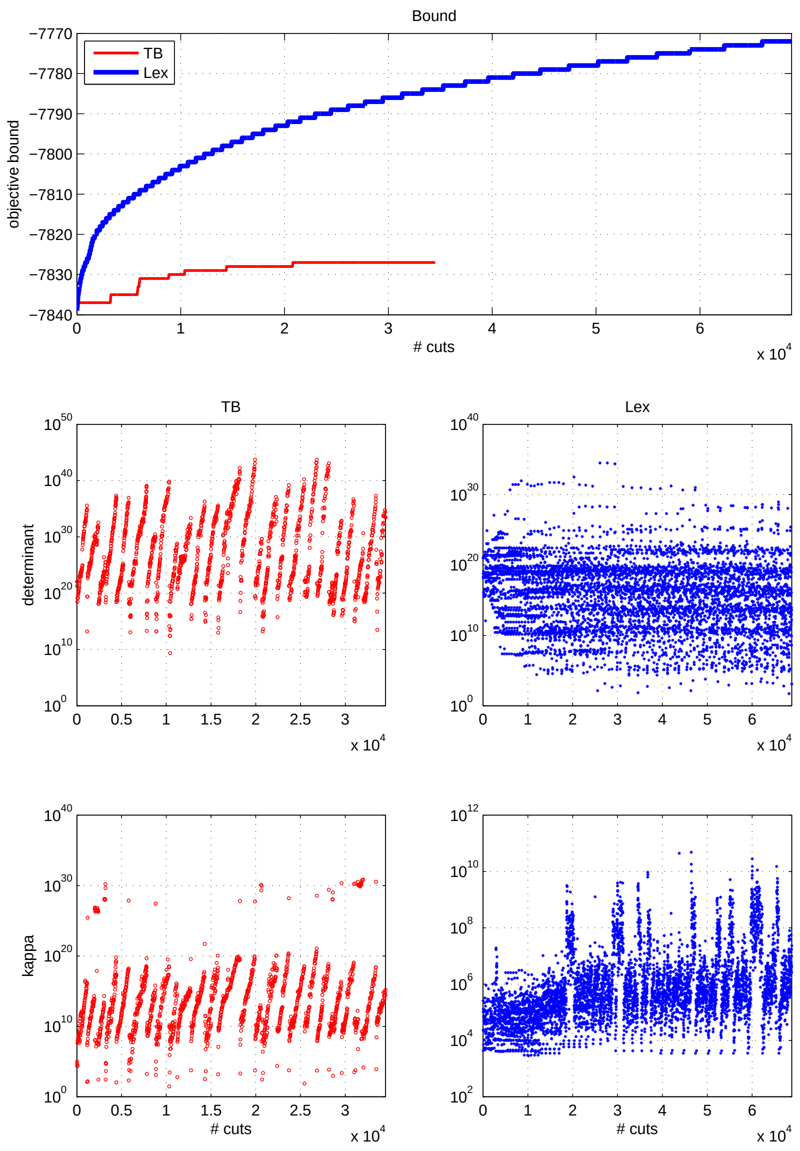

Fig. 4. Comparison between the textbook and lexicographic implementations of multi-cut Gomory's algorithm on `sentoy`.

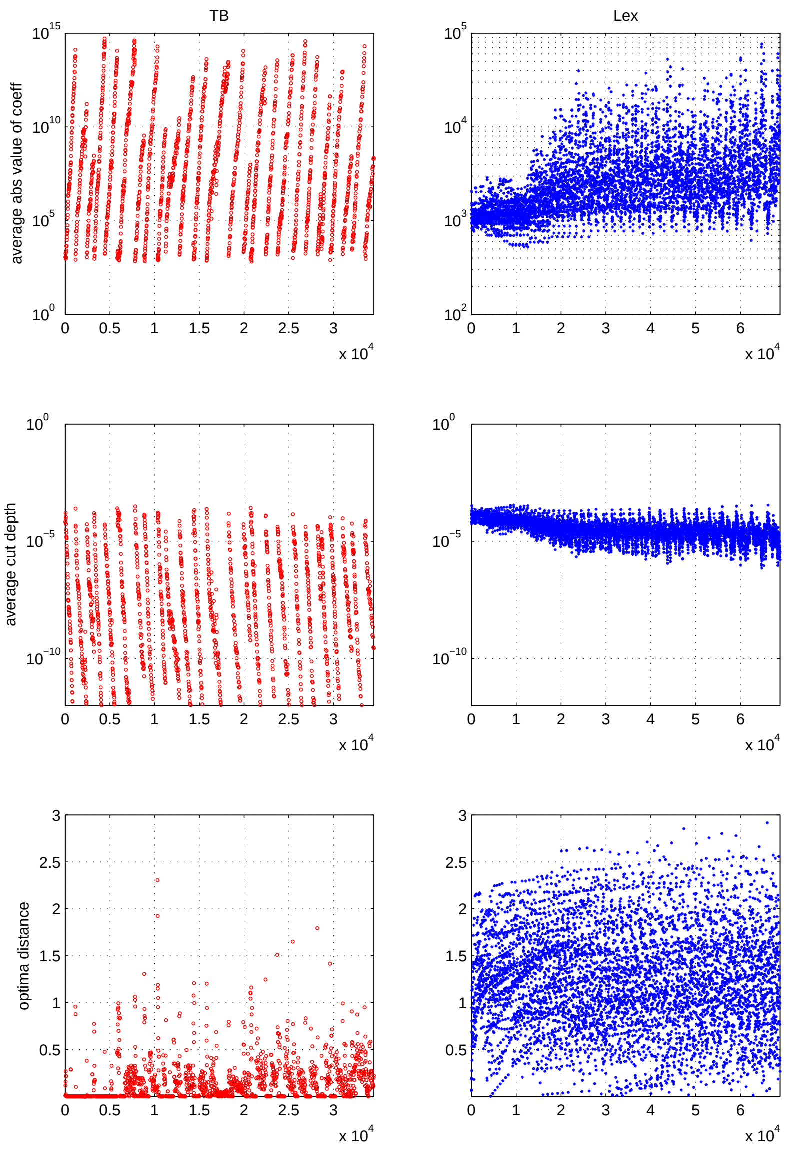

Fig. 5. Comparison between the textbook and lexicographic implementations of multi-cut Gomory's algorithm on `sentoy`

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.

The significance of these result suggests that the lexicographic approach can be applied to any cutting plane algorithm—no matter what kind of cuts you add in the step that generates cuts, you may reoptimize using the lexicographic dual simplex method so as to break dual degeneracy in favor of "cleaner" LP bases associated with better primal vertices to cut. We plan to investigate this topic in the near future.

References

1. J. L. Arthur and A. Ravindran. PAGP, a partitioning algorithm for (linear) goal programming problems. *ACM Trans. Math. Softw.*, 6(3):378–386, 1980.
2. E. Balas, S. Ceria, G. Cornuéjols, and N. Natraj. Gomory cuts revisited. *Operations Research Letters*, 19:1–9, 1996.
3. 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.
4. I. Borg and P.J.F. Groenen. *Modern Multidimensional Scaling: Theory and Applications.* Springer, 2005.
5. Balas E. and A. Saxena. Optimizing over the split closure. *Mathematical Programming*, DOI 10.1007/s10107-006-0049-5, 2006.
6. M. Fischetti and Lodi A. Optimizing over the first Chvátal closure. *Mathematical Programming B*, 110(1):3–20, 2007.
7. R. E. Gomory. Outline of an algorithm for integer solutions to linear programs. *Bulletin of the American Society*, 64:275–278, 1958.
8. R. E. Gomory. An algorithm for the mixed integer problem. Technical Report RM-2597, The RAND Cooperation, 1960.
9. R. E. Gomory. An algorithm for integer solutions to linear programming. In R. L. Graves and P. Wolfe, editors, *Recent Advances in Mathematical Programming*, pages 269–302, New York, 1963. McGraw-Hill.
10. 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.