Noname manuscript No.
(will be inserted by the editor)

A hard integer program made easy by lexicography*

Egon Balas · Matteo Fischetti · Arrigo Zanette

February 16, 2011

Abstract A small but notoriously hard integer program formulated by Donald Knuth fifty years ago is solved by three versions of a lexicographic algorithm using Gomory cuts. The lexicographic cutting plane algorithms are faster than CPLEX on this problem by a factor of at least 10.

Keywords Cutting Plane Methods · Gomory Cuts · Degeneracy in Linear Programming · Lexicographic Dual Simplex · Computational Analysis

1 Introduction

During a recent visit to Carnegie Mellon University, Donald Knuth mentioned in a conversation with one of the authors of this note that early in his career he formulated an integer program that could not be solved with the technology of the time and remained unsolved for another 35 years. Upon being asked for the details of the case, he then kindly provided the information that follows.

As an undergraduate student at Case Institute of Technology, in 1960 Donald Knuth wrote a paper [2] that gave an integer programming formulation to the problem of minimizing drum latency time, i.e. optimizing the allocation of memory positions for the instructions and data of a program stored on a rotating magnetic drum. The formulation used 51 integer variables and 43 inequalities, besides the nonnegativity constraints on each variable. The young Knuth also implemented Gomory's algorithm, or, to put it in his own words, "I used Gomory's original all-integer algorithm, based on

* This note is an Addendum to [5]

E. Balas
Carnegie Mellon University, Pittsburgh, PA
E-mail: eb17@andrew.cmu.edu

M. Fischetti
DEI, University of Padova
E-mail: matteo.fischetti@unipd.it

A. Zanette
DEI, University of Padova
E-mail: zanettea@gmail.com

his preprint dated January 29, 1960; he told me my implementation was probably the first outside of Yorktown Heights" [3]. He then tried to solve the instance on an IBM 650 and of course failed. In spite of its relatively modest size and the obvious historical interest that it presented, the problem remained unsolved for the next 35 years. Finally, in 1995 Dimitris Alevras at ZIB (Berlin) solved it by running it on CPLEX, on a SPARCstation 5. A solution of value of 22,996 was found after generating 8,880 nodes of a branch-and-bound tree, but it took 732,200 nodes of the search tree to prove that solution optimal. Alevras reported that the problem was "amazingly sensitive from a numerical point of view" [3]. Another run, with different CPLEX parameter settings for branching, and using SOS, clique and cover inequalities, needed more than 22,000 nodes just to find a first integer solution.

2 The experiment

Having recently discovered [5, 1] that a lexicographic version of Gomory's cutting plane algorithm for pure integer programs is numerically stable and capable of solving instances of considerable difficulty for other codes, we decided to run it on Dr. Knuth's problem, available for download at `www.dei.unipd.it/~fisch/knuth.lp`. We first ran the problem with IBM ILOG CPLEX 12.2, with default settings. Of course, the CPLEX of today is by far superior to the one of 1995, but it still took more than 50,000 nodes of the branch-and-bound tree (in its default setting) to find a solution of the same value of 22,996 as the one found by Alevras, and to prove its optimality. We then ran the problem with our version of the lexicographic cutting plane algorithm in its simplest variant, which generates a sequence of individual cuts and reestablishes the lexicographic optimality of the tableau after adding each cut. To our pleasant surprise, our algorithm found the same optimal solution of 22,996 after generating 5,618 cuts! The contrast to the over 50,000 search tree nodes that CPLEX needed to accomplish the task is striking. A branch and bound tree with 50,000 nodes implies the solution of as many linear programs. On the other hand, generating 5,618 cuts requires practically no time, the cuts being read off the simplex tableau, and reoptimizing after the addition of each cut requires the solution of one linear program. True, the linear program is solved to lexicographic optimality, which involves considerably more pivots than just finding an optimal solution; but still, the effort is proportional to the number of cuts and simply not comparable to the effort of solving 50,000 linear programs. In particular, the total number of pivots required by our lexicographic method (including those due to lexicographic reoptimization) is surprisingly small: just 31,842 instead of the over 211,000 required by CPLEX. In terms of the time needed by the two codes to solve the problem, the outcome was comparable (9.0 seconds for CPLEX and 6.2 seconds for our code), but the current version of CPLEX is the result of two decades of development by a strong team of programming experts, an "industrial product", whereas our code is merely a research tool, the "handicraft" work of a single student helped by his advisor.

We also solved Knuth's problem by two additional versions of our lexicographic cutting plane algorithm, introduced in [1], and for comparison, with a standard implementation of Gomory's cutting plane algorithm, and with CPLEX 12.2. Table 1 reports the outcome of these runs. The entries of the first column are as follows.

– *TB* stands for the "textbook" implementation of Gomory's 1958 cutting plane method for pure integer programs, in its modern, "multi-cut" variant, which gener-

Method	Time (sec.s)	#LP's	#pivots	#cuts	#nodes	Gap closed
TB (multi-cut)	>8h	2,197	252,225	124,414	0	69%
lex-GFC (single cut)	6.20	5,618	31,842	5,618	0	100%
lex-GFC (multi-cut)	0.30	111	1,941	5,524	0	100%
Bin-GFC	0.15	42	1,032	1,171	0	100%
Cplex 12.2 (default)	9.01	42,607	211,026	102	53,009	100%
Cplex 12.2 (strong-branching)	8.21	91,513	52,710	85	9,585	100%
Cplex 12.2 (aggressive cuts)	4.32	7,298	50,270	148	7,142	100%

Table 1 Comparison of different solution methods; the first 4 are pure cutting plane methods, while the last 3 are branch-and-bound routines of Cplex under 3 different parameter settings.

ates a Gomory cut (or GFC) from every fractional basic variable before reoptimizing. The original, "single-cut" version, which reoptimizes after every cut, breaks down after a few thousand cuts.

- *lex-FGC (single cut)* is our lexigraphic cutting plane algorithm using GFC cuts, with (partial) lexicographic reoptimization after every cut. For details see [1], sections 3 and 7.
- *lex-FGC (multi-cut)* is the same, but with cuts generated from every fractional basic variable before (partial) lexicographic reoptimization
- *Bin-FGC* is a version of the lexicographic algorithm introduced in [1] (section 5), which replaces the objective function with a binary search of its optimal value. Thus lex-GFC (multi-cut) is used repeatedly to explore the existence of a feasible solution within a given cost-interval.
- *CPLEX 12.2* is the integer solver of CPLEX, run under three different parameter settings, namely "default," "strong branching" and "aggressive cuts."

According to Table 1, the best performer in terms of both time and number of linear programs solved is Bin-GFC, which is about 30 times faster than CPLEX under the best parameter settings. The multi-cut version of lex-GFC is about 15 times faster than CPLEX. The growth of the lower bounds given by the LP values with the number of iterations (i.e., number of LP's solved) is shown for the first three methods of Table 1 in Figure 1 (left).

Comparing the data for the best performing version of the lexicographic approach with those for the best CPLEX settings, we find that the former requires 2% of the number of pivots required by the latter. Another highly interesting aspect of our experiments is the fact, illustrated in Figure 1 (right), is that the sequence of determinants of the successive basis matrices after each lexicographic reoptimization, instead of increasing as usual when the cuts produce dual degeneracy and the solutions get closer and closer to each other, in fact keeps decreasing throughout the run. As discussed in [5, 1], the main reason for this behavior is that as a sequence of cuts is added to the simplex tableau, dual degeneracy becomes a typical phenomenon, resulting in multiple optimal solutions to the current LP relaxation. When the standard dual simplex method is used to reoptimize after adding a new cut, the algorithm tends to pick among the many optimal solutions one that is closest to the old solution. This leads to the appearance of very large basis determinants, needed to differentiate between points closer and closer to each other. The lexicographic dual simplex on the other hand does not stop the reoptimization upon finding the first dual-feasible solution, but continues to pivot until it reaches a lexicographically dual-feasible solution, which is usually among those furthest from the previous optimum; see Figure 2 for an illustration of the LP

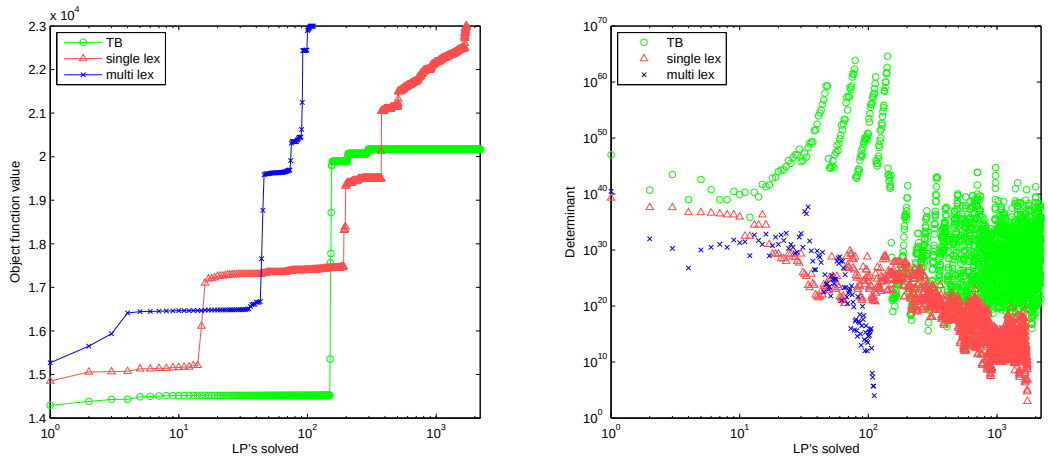

Fig. 1 Lower bound growth for the pure cutting plane methods (left) and comparison of basis determinants (right).

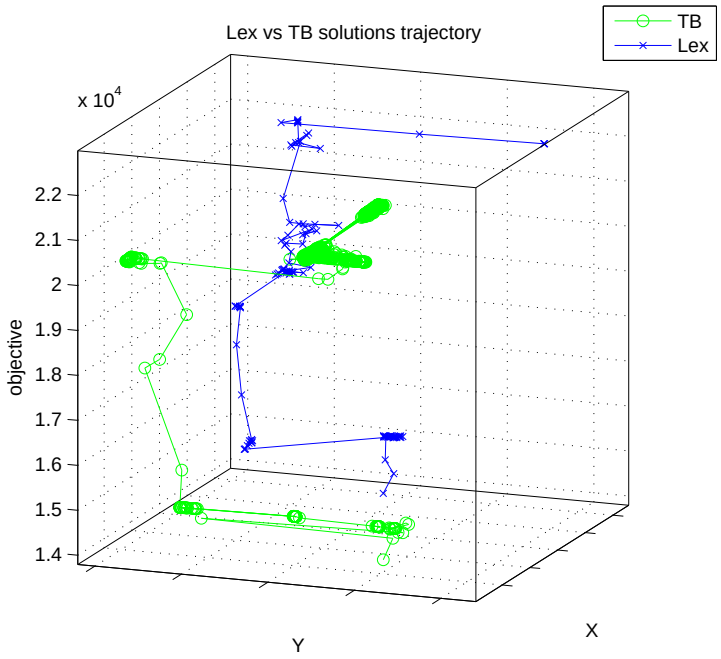

Fig. 2 LP solution trajectories (X and Y represent a projection of the variable's space; see [5] for details)

Method	Time (sec.s)	#LP's	#pivots	#cuts	#nodes	Gap closed
TB (multi-cut)	>8h	6,407	691,746	422,821	0	96%
lex-GFC (single cut)	8.1	6,554	39,528	6,554	0	100%
lex-GFC (multi-cut)	0.9	316	5,015	13,652	0	100%
Bin-GFC	0.6	161	5,027	1,243	0	100%
Cplex 12.2 (default)	5.46	28,352	127,077	76	33,845	100%
Cplex 12.2 (strong-branching)	31.17	348,852	228,993	68	49,061	100%
Cplex 12.2 (aggressive cuts)	21.93	44,704	336,148	215	49,053	100%

Table 2 Comparison of different solution methods for the corrected instance.

optimal solutions. In this way, the determinant of the basis matrix is not forced to increase during the run. Our explanation for the fact that the determinants actually decreased during the run is that whereas the initial matrix included some relatively large numbers (like 50), the cutting planes generated were numerically "clean", i.e. with small coefficients, and as they gradually made the original constraints redundant, they produced basis matrices with smaller numbers and cleaner structure.

In his message to us accompanying the description of the problem, Dr. Knuth states that he has recently discovered an error in the original problem statement, which he hopes will not invalidate the optimal solution found by Alevras: one of the inequalities needs to be replaced with two others, also involving a new variable (this instance is available for download at `www.dei.unipd.it/~fisch/knuth_corrected.lp`; see pages 435–436 of [4]). We also ran the corrected version of the problem, and found that the seemingly minor change makes the problem substantially more difficult: although the optimal solution remains the same as before there is a sharp increase in the computational effort required by either method to solve it. Table 2 reports the results. Although the difference in performance is less sharp than before, CPLEX with the most favorable settings still takes 10 times as long as the lexicographic Bin-GFC.

Acknowledgments

The research of the first author was supported by the National Science Foundation through grant #DMI-0352885 and by the Office of Naval Research through contract N00014-03-1-0133. The research of the second author was supported by the "Progetto di Ateneo" on "Computational Integer Programming" of the University of Padova. We thank Donald Knuth for calling our attention to this remarkable instance and providing all the information available on its past history.

References

1. E. Balas, M. Fischetti, and A. Zanette. On the enumerative nature of Gomory's dual cutting plane method. *Mathematical Programming, Series B*, (125):325–351, 2010.
2. D. Knuth. Minimizing drum latency time. *Journal of the ACM*, 8:119–150, April 1961.
3. D. Knuth. An integer programming problem. Manuscript, July 1993.
4. D. Knuth. *Selected papers on Design of Algorithms*, pages 435–436. CSLI Lecture Notes n. 191, Stanford, California, February 2010.
5. A. Zanette, M. Fischetti, and E. Balas. Lexicography and degeneracy: can a pure cutting plane algorithm work? *Mathematical Programming, Series A*, 2009. DOI: 10.1007/s10107-009-0335-0.