Verified Linear Programming – a Comparison

Christian Keil*

Institute for Reliable Computing, Hamburg University of Technology,
Schwarzenbergstraße 95, 21073 Hamburg, Germany

Key words Linear Programming, Comparison, Rigorous Error Bounds, Lurupa.
2000 Mathematics Subject Classification 65G50, 65K05, 65Y20, 90C05

Linear programming is arguably the most basic form of optimization. Numerous algorithms have been developed and implementations exist in presumably every programming language. The applications range from linear optimization problems to branch-and-bound algorithms for mixed integer problems and global optimization problems.

To eliminate the influence of rounding errors and obtain rigorous results, verification methods have to be introduced. We will analyze different techniques for verified linear programming and take a look at some numerical results.

1 Introduction

Linear Programming is not only the arguably first and most basic kind of optimization students get introduced to. Its relevance today exceeds this by far.

Although some preliminary works exist, for example Fourier [8], de la Vallée Poussin [5], and Kantorovich [13], the success story of linear programming really started in 1947 with the discovery of the Simplex method by Dantzig [3]. Together with the development of the computer, the Simplex method allowed to mechanically solve optimization problems that required more than 100 man-days using hand-operated desk calculators. Its computational power was illustrated by Dantzig [4] using a simple problem of assigning 70 men to 70 jobs. While exploring all possible solutions "would require a solar system full of nano-second electronic computers running from the time of the big bang until the time the universe grows cold", applying standard simplex or interior point software "it takes only a moment to find the optimum solution using a personal computer". In the same paper he stated, "the tremendous power of the simplex method is a constant surprise to me".

The importance of linear programming is exemplified by Lovasz [22]: "If one would take statistics about which mathematical problem is using up most of the computer time in the world, then (not including database handling problems like sorting and searching) the answer would probably be linear programming". That same year Eugene Lawler of Berkeley summarized: "It [linear programming] is used to allocate resources, plan production, schedule workers, plan investment portfolios and formulate marketing (and military) strategies. The versatility and economic impact of linear programming in today's industrial world is truly awesome."

Rounding errors, inevitably introduced by floating point arithmetic, can well deteriorate the results of numerical computations. This is especially true for ill-conditioned problems. The practical relevance for linear programming, is emphasized by Ordóñez and Freund [26]. They analyzed the problems in the Netlib collection [24] of real-life linear programs and observed 71% to be ill-conditioned.

2 Available Algorithms

Several methods are capable of producing rigorous results for linear programs. According to the techniques used, they can be classified into four categories. Algorithms for constraint satisfaction problems,

* e-mail: c.keil@tu-harburg.de

algorithms using rational arithmetic, global optimization algorithms, and algorithms specifically devised for linear programming in the presence of rounding errors and uncertainties in the input data.

Algorithms for constraint satisfaction problems (CSP) offer no means to optimize. Thus rigorous versions can only be used in a limited way for verified linear programming. Assume f^* to be an approximation to the optimal value of a minimization problem. Now build a CSP consisting of the constraints of the linear program plus a constraint on the objective function, $f(x) \leq f^* - \varepsilon$. If a rigorous algorithm verifies this CSP to be infeasible, $f^* - \varepsilon$ is a lower bound on the true optimal value of the linear program. Analogously upper bounds can be derived from feasible CSPs having an additional constraint of the form $f(x) \geq f^* + \varepsilon$. Two implementations of rigorous constraint satisfaction algorithms are ICOS [21] and RealPaver [10]. ICOS is based on constraint programming and interval analysis techniques. RealPaver uses a branch-and-prune algorithm, the pruning step merges constraint satisfaction techniques with the interval Newton method.

The second category of methods is exploiting the fact that the optimal solution and value of a linear program are rational numbers. Thus rigorous results can be obtained using rational arithmetic, but not for interval problems. This has been done for example by Gärtner [9] and Dhiflaoui et al [6]. The only available implementation seems to be perPlex by Koch [19], which only verifies the optimality of an approximate solution. Offering neither means to obtain this approximate solution in the first place, nor to proceed from there if the solution proves suboptimal, it was also excluded from the comparison.

Global optimization algorithms handle problems where the objective and the constraints are defined by smooth algebraic expressions. Hence they can solve linear programs rigorously. Implementations are Numerica [29], GlobSol [14], and COSY [7] (on inquiry it was not possible to obtain a copy of COSY). Numerica combines traditional interval methods (e.g., Hansen–Sengupta's operator) with constraint satisfaction techniques. GlobSol combines an interval branch-and-bound procedure with additional techniques, such as automatic differentiation, constraint propagation, interval Newton methods, and additional, specialized techniques. COSY uses a branch-and-bound scheme featuring a Taylor model arithmetic for the bounding step.

Finally are the algorithms specifically designed for verified linear programming in the presence of rounding errors and uncertainties in the input data. These ideas go back to Krawczyk [20], who computed rigorous enclosures of the optimal solution of a linear program. They were used and refined by Beeck [1] and Rump [28], and extended to degenerate problems by Jansson [11]. Requiring the solution of interval linear systems, all these methods share a computational work being cubic in the number of variables. More recently Jansson [12] and Neumaier and Shcherbina [25] independently devised methods to rigorously bound the optimal value with a quadratic complexity if finite bounds on all variables are known.

Lurupa [16] belongs to this fourth category. Implementing the algorithms developed by Jansson [12], it tries to enclose feasible, near-optimal solutions for the primal and for the dual problem. The iterative algorithm starts by solving the original problem with a standard linear programming solver. Based on the approximate solution, interval arithmetic is now applied to enclose a feasible point. If this succeeds, the rigorous bound on the optimal value follows with the range of the objective function over this enclosure. Otherwise the algorithm starts over, perturbing the linear program to force the approximate solution further into the interior of the feasible region. In the presence of uncertainty in the input data, only approximate solutions to midpoint problems are necessary, and still a standard linear programming solver is sufficient. The algorithm just postprocesses approximate solutions. It can thus be combined with any standard linear programming solver to compute rigorous results.

Lurupa has proven capable of computing rigorous error bounds for the optimal value of linear programming problems with several thousands of variables and constraints (see [17]). Roughly spoken, finite lower and upper bounds could be computed iff the respectively primal and dual problems were not ill-posed. For various problems of the Netlib lp library, lower and upper bounds were obtained, certifying the existence of optimal solutions.

Consisting of modules for the different functional parts, Lurupa is fully implemented in ANSI C++ and easily extendable. The aim is to provide a fast implementation of the algorithms, available as a stand-alone and a library version to be integrated into larger frameworks. For the interval computations it uses PROFIL/BIAS [18]. Solver modules are currently available for lp_solve [2] in different versions.

Additional solver modules have to implement a common interface, their main function is transforming data between the representations used in Lurupa and in the solver.

3 Numerical results

To compare the performance of GlobSol, Numerica, and Lurupa, test problems were generated with the method of Rosen and Suzuki [27]. The objective function and the constraints were chosen to be linear with random, integral coefficients. The optimal solution was chosen random, integral, and nondegenerated between the simple lower and upper bound of -10 and 10 on each variable. Thus the optimal solution as well as its objective value are known exactly. A second test set consists of some small problems from the *Misc* section of Meszaros's collection [23], `kleemin3` – `kleemin8` and `farm`. The dimensions of the `kleemin` problems are indicated by the trailing digit, `kleemin3` has 3 variables and 3 inequalities, `kleemin4` has 4 variables and 4 inequalities and so on. A slightly larger problem is `farm` with 5 inequalities, 2 equations, and 12 variables. As these problems have infinite simple upper bounds, which Numerica and GlobSol do not handle well, an artificial finite upper bound enclosing the whole feasible region was set on the variables.

Most of the settings in GlobSol's configuration file were left at their defaults as suggested by Kearfott [15]. The upper bound on the number of boxes `MAXITR` was increased if GlobSol aborted due to this limit and the maximum cpu time `MAX_CPU_SECONDS` was set to 7200. Peeling was enabled for all bounds. For the problems GlobSol could not solve, the databox was enlarged and peeling disabled. It might be possible to obtain better results by adjusting the stopping tolerances to the specific problem instances but this is not suitable for a benchmarking situation. It also reflects the experience of a user without a deeper insight into the algorithm and its parameters.

Applied to the first test set GlobSol successfully computed enclosures of the optimal solution and value if the number of variables was less than 10. The relative accuracy of the computed bounds was between 10^{-12} and 10^{-1} for the largest problems GlobSol solved. Increasing the number of variables to 10 resulted in GlobSol not terminating within two hours, increasing to 15 resulted in GlobSol running out of memory. For the second test set, GlobSol rigorously solved the `kleemin` problems in under 5 seconds, but did not terminate for `farm`.

For the Rosen and Suzuki test problems, Numerica successfully solved a problem with 5 inequality constraints and 5 variables with a relative accuracy of 10^{-12}. Adding constraints or increasing the number of variables resulted in Numerica not terminating. Looking at the Meszaros problems, Numerica successfully enclosed the optimal values of `kleemin3` to `kleemin6`, but failed due to insufficient memory for the larger ones.

Lurupa enclosed the optimal values and near-optimal feasible points for all problems in less than 10ms using lp_solve version 5.5. The relative accuracy of the computed bounds was at least 10^{-8}. Only for `kleemin8` the accuracy was 10^{-6} due to the quality of the approximate solution.

4 Conclusion

Applying current general optimization software with rigorous result verification to linear programs works up to a dimension of 10 variables. Larger problems require algorithms that exploit the special structure of the problems. Lurupa implements such algorithms and solves medium scale linear programs in a reasonable time (see [17]).

References

[1] H. Beeck. Linear Programming with Inexact Data. Technical Report 7830, Abteilung Mathematik, TU München, 1978.

[2] M. Berkelaar, P. Notebaert, and K. Eikland. lp_solve. World Wide Web. `http://groups.yahoo.com/group/lp_solve`.

[3] G.B. Dantzig. Maximization of a linear function of variables subject to linear inequalities. In T.C. Koopmans, editor, *Activity Analysis of Production and Allocation*, number 13 in Cowles Commission Monographs, pages 339–347, 440 Fourth Ave., New York City, 1951. John Wiley & Sons, Inc.

[4] G.B. Dantzig. *History of Mathematical Programming: A Collection of Personal Reminiscences*, chapter Linear Programming, pages 19–31. Elsevier Science Publishers, 1991.

[5] C. de la Vallée Poussin. Sur la méthode de l'approximation minimum. *Annales de la Société Scientifique*, 35:1–16, 1911.

[6] M. Dhiflaoui, S. Funke, C. Kwappik, K. Mehlhorn, M. Seel, E. Schömer, R. Schulte, and D. Weber. Certifying and repairing solutions to large LPs how good are LP-solvers? In *SODA*, pages 255–256, 2003.

[7] M. Berz et al. COSY Infinity. World Wide Web. `http://www.bt.pa.msu.edu/index_files/cosy.htm`.

[8] J.B.J. Fourier. Solution d'une question particulière du calcul des inégalités. *Nouveau Bulletin des sciences par la Société philomathique de Paris*, pages 99–100, 1826.

[9] B. Gärtner. Exact arithmetic at low cost – a case study in linear programming. *Computational Geometry*, 13(2):121–139, June 1999.

[10] L. Granvilliers and F. Benhamou. Algorithm 852: RealPaver: An Interval Solver using Constraint Satisfaction Techniques. *ACM Transactions on Mathematical Software*, 32(1):138–156, 2006. `http://doi.acm.org/10.1145/1132973.1132980`.

[11] C. Jansson. A Self-Validating Method for Solving Linear Programming Problems with Interval Input Data. *Computing*, Suppl. 6:33–45, 1988.

[12] C. Jansson. Rigorous Lower and Upper Bounds in Linear Programming. *SIAM J. Optim.*, 14(3):914–935, 2004.

[13] L.V. Kantorovich. *Mathematical methods in the organization and planning of production (in Russian).* Publication House of the Leningrad State University, 1939. English Translation: Management Science, Volume 6 (1960), pp. 363–422.

[14] R.B. Kearfott. Globsol. World Wide Web. `http://interval.louisiana.edu`.

[15] R.B. Kearfott. Globsol: History, composition, and advice on use. In C. Bliek, C. Jermann, and A. Neumaier, editors, *Global Optimization and Constraint Satisfaction*, volume 2861 of *Lecture Notes in Computer Science*, pages 17–31. Springer Berlin / Heidelberg, 2003.

[16] C. Keil. Lurupa – Rigorous Error Bounds in Linear Programming. In B. Buchberger, S. Oishi, M. Plum, and S.M. Rump, editors, *Algebraic and Numerical Algorithms and Computer-assisted Proofs*, number 05391 in Dagstuhl Seminar Proceedings. Internationales Begegnungs- und Forschungszentrum fuer Informatik (IBFI), Schloss Dagstuhl, Germany, 2006. `http://drops.dagstuhl.de/opus/volltexte/2006/445`.

[17] C. Keil and C. Jansson. Computational Experience with Rigorous Error Bounds for the Netlib Linear Programming Library. *Reliable Computing*, 12(4):303–321, 2006.

[18] O. Knüppel. PROFIL/BIAS and extensions, Version 2.0. Technical report, Inst. f. Informatik III, Technische Universität Hamburg-Harburg, 1998.

[19] T. Koch. The final netlib-lp results. Technical Report 03-05, Konrad-Zuse-Zentrum für Informationstechnik Berlin, Takustraße 7, D-14195 Berlin-Dahlem, Germany, 2003.

[20] R. Krawczyk. Fehlerabschätzung bei linearer Optimierung. In K. Nickel, editor, *Interval Mathematics*, volume 29 of *Lecture Notes in Computer Science*, pages 215–222. Springer Verlag, Berlin, 1975.

[21] Y. Lebbah. ICOS (Interval COnstraints Solver). World Wide Web. `http://www.essi.fr/~lebbah/icos/index.html`.

[22] L. Lovasz. A New Linear Programming Algorithm – Better or Worse Than the Simplex Method? *The Mathematical Intelligencer*, 2:141–146, 1980.

[23] Csaba Mészáros. Linear programming test problems. `http://www.sztaki.hu/~meszaros/bpmpd`.

[24] Netlib. Netlib linear programming library. `http://www.netlib.org/lp`.

[25] A. Neumaier and O. Shcherbina. Safe bounds in linear and mixed-integer programming. *Mathematical Programming, Ser. A*, 99:283–296, 2004.

[26] F. Ordóñez and R.M. Freund. Computational experience and the explanatory value of condition measures for linear optimization. *SIAM J. Optimization*, 14(2):307–333, 2003.

[27] J.B. Rosen and S. Suzuki. Construction of Nonlinear Programming Test Problems. *Communication of ACM*, 8:113, 1965.

[28] S.M. Rump. Solving Algebraic Problems with High Accuracy. Habilitationsschrift. In U.W. Kulisch and W.L. Miranker, editors, *A New Approach to Scientific Computation*, pages 51–120. Academic Press, New York, 1983.

[29] P. Van Hentenryck, P. Michel, and Y. Deville. *Numerica: A Modelling Language for Global Optimization.* MIT Press Cambridge, 1997.