Certifying and Repairing Solutions to Large LPs How Good are LP-solvers? *

Marcel Dhiflaoui Stefan Funke Carsten Kwappik Kurt Mehlhorn
Michael Seel Elmar Schömer Ralph Schulte Dennis Weber
Max-Planck-Institut für Informatik, Stuhlsatzenhausweg 85, 66123 Saarbrücken, Germany

Abstract

State-of-the-art linear programming (LP) solvers give solutions without any warranty. Solutions are not guaranteed to be optimal or even close to optimal. Of course, it is generally believed that the solvers produce optimal or at least close to optimal solutions.

We have implemented a system LPex which allows us to check this belief. More precisely, given an LP and a basis B, it determines whether the basis is primal feasible and/or dual feasible. It can also find the optimum starting from an arbitrary basis (or from scratch). It uses *exact* arithmetic to guarantee correctness of the results. The system is efficient enough to be applied to medium- to large-scale LPs. We present results from the netlib benchmark suite.

1 Introduction

Linear programming

$$\max c^T x \quad \text{subject to} \quad Ax \leq b,\ x \geq 0$$

is one of the most important algorithmic paradigms. Many problems can be formulated as linear programs and efficient solvers are available. Linear programs arising in practice can be quite large with ten-thousands to hundred-thousands of variables and/or constraints. Of course, in these large programs the constraint matrix will be sparse. We use n, m, and nz, to denote the number of variables, constraints, and non-zeroes in the constraint matrix, respectively.

Existing LP-solvers do not claim to solve LPs to optimality. In fact, they come with hardly any guarantee. Feasible problems may be classified as infeasible and vice versa, a solution returned is not guaranteed to be feasible, and the objective value returned comes with no approximation guarantee. Some solvers guarantee that they will give an answer. A notable exception is the work by Gärtner [Gär98]. He developed an exact LP solver (= a solver which always returns the mathematically correct result), which works well for small linear programs with $\min(n, m) < 1000$ as they occur, for example, in computational geometry. His approach does not work for larger LPs arising in typical operations research applications like flight scheduling etc., as it explicitly inverts the basis matrix. It is well known that even if the basis matrix is sparse, its inverse is typically not sparse, hence making the explicit inverse computation hopeless for matrices of size 1000×1000 or more.

2 Why Reliability?

It seems natural to require that widely used software is reliable, i.e., that its answers come with some guarantee. Non-reliable software is hard to use, at least for non-experts. We give just two examples.

Some of us competed in an integer linear programming contest. We used a heuristic to compute feasible solutions and we used linear programming to compute upper bounds. We had found a feasible solution of value 12 and the LP (solved with CPLEX [CPL]) gave us an upper bound of 12.99999999. We erroneously concluded our solution to be optimal. LP-experts told us later that one should never exploit that 12.99999999 is smaller than 13. It would have been safe to exploit that 12.9990 is smaller than 13. How do they know, given that the solvers come with no guarantee?

The output of LP solvers are also used to prove theoretical results in computer science. For example, in [LW99], the authors set up a LP and used CPLEX to calculate the competitive ratios of their online algorithm. What do the computed values mean?

3 Goal, Approach and Results

Our goal is to develop techniques which allow us to solve medium- to large-scale LPs to optimality. Our approach is *verify and repair*. We exploit the fact that LP-solvers

- return a basis (= a combinatorial object) and not just an objective value and a solution vector and that
- hopefully the basis returned is either optimal or at least close to optimal.

We have implemented *LPex*. LPex comprises

- exact methods to determine whether a given basis is primal or dual feasible and
- an exact LP-solver which can either start at a given basis (primal or dual feasible or arbitrary) or can start from scratch.

*{marcel,funke,kwappik,schoemer,mehlhorn,seel,rschulte,dennis}@mpi-sb.mpg.de. This work was partially supported by the IST Programme of the EU under contract number IST-1999-14186 (ALCOM-FT).

	CPLEX solution		Exact Verification		
Problem	RelObjErr	T	V	Res	T
25fv47 (822 × 1571, 11k)	7.27e-15	2.47	0	opt	365.9
80bau3b (2263 × 9799,29k)	1.42e-15	2.97	0	opt	10.68
bandm (306×472,2k)	1.43e-15	0.07	0	opt	17.72
bnl1 (644 × 1175,6k)	8.05e-16	0.92	0	opt	3.57
cycle (1904 × 2857,21k)	0.00e+00	0.46	0	opt	2.62
d6cube (416 × 6184,43k)	9.69e-14	29.57	0	opt	51.56
degen2 (445 × 534,4k)	0.00e+00	0.36	0	opt	0.96
degen3 (1504 × 1818,26k)	6.91e-16	8.08	0	opt	8.79
etamacro (401 × 688,2k)	1.50e-16	0.13	10	feas	1.11
fffff800 (525 × 854,6k)	0.00e+00	0.09	0	opt	4.41
fit2d (26 × 10500,138k)	4.25e-16	6.03	0	opt	64.02
fit2p (3001 × 13525,60k)	8.50e-16	38.56	0	opt	16.03
greenbea (2393 × 5405,31k)	2.05e-16	4.52	0	opt	257.23
greenbeb (2393 × 5405,31k)	0.00e+00	4.35	0	opt	750.55
perold (626 × 1376,6k)	5.49e-14	2.35	0	opt	1959.45
pilot.ja (941 × 1988,14k)	5.92e-12	8.82	0	opt	2697.46
pilot.we (723 × 2789,9k)	2.93e-11	3.8	0	opt	1654.64
scsd6 (148 × 1350,5k)	0.00e+00	0.1	13	feas	0.52
scsd8 (398 × 2750,11k)	7.54e-16	0.48	0	opt	1.52

Table 1: Verification of CPLEX solutions for problems in the Netlib library. The first column gives the name of the instance, and the number of constraints, variables,and non-zeroes. The columns labeled T show the time (in seconds) for solving the LP with CPLEX and the time for checking the basis return with LPex. In all but two cases the basis returned was optimal. In the remaining two cases the basis was at least feasible. Column V indicates the number of violated dual constraints (= number of variables with negative reduced cost). RelObjErr is the relative error in the objective value at the basis returned by CPLEX, e.g., for problem pilot.we CPLEX found the optimal basis and returned an objective value with relative error 2.93e-11.

CPLEX Solution			Exact Verif.	Exact Correction	
Problem	RelObjErr	T[sec]	dCon	RelObjValCor	T [sec]
etamacro	1.50e-16	0.18	10	5.90e-09	2.43
scsd6	2.81e-16	0.14	13	2.38e-11	5.93

Table 2: LPex's repair of "optimal" CPLEX bases. The column RelObjValCor shows the relative difference in objective value between the optimal solution and the solution returned by CPLEX.

We use the first kind of methods to *verify* solutions returned by existing LP-solvers, see Table 1, and we use the exact LP-solver to search for optimal solutions starting from the basis returned by existing solvers, see Table 2. One can also use LPex to solve linear programs from scratch, see Figure 3.

4 Methods

LPex comprises the following modules. All modules are based on exact arithmetic (integer or rational or modular).

A module to discover block structure in matrices [RT78]. It first computes a permutation of the columns such that all diagonal elements become non-zero (a perfect matching) and then computes the strongly connected components of the directed graph defined by the matrix. The components form the blocks and the matrix becomes block-triagonal.

A module to compute LU-factorizations of sparse matrices with integer and rational entries. The module implements Bareis's method and uses exact integer and rational arithmetic.

Problem	LP Solver	Objective Function Value	Time [sec]
fit1d[0]	LPex	-9.1463780924209277e+03	325.77
(1.026 × 24,	CPLEX	-9.1463780924209241e+03	0.28
13.404 nz)	SoPlex	-9.1463780924209132e+03	2.36
	PCx	-9.14637**6**59e+03	1.19

Table 3: Timings and solution accuracy for different LP solvers. The first incorrect figure in the objective value is shown in bold.

A module to solve sparse linear systems over the rationals using Wiedemann's method [Wie86] over finite fields and Chinese remaindering. Wiedemann's method parallelizes with almost linear speed-up. We have achieved a speed-up of almost 40 on a PC-cluster with 40 processors.

A module to determine whether a basis of a linear program is primal (dual) feasible. This module uses the preceding modules as work horses.

Three modules for solving linear programs starting from a given basis. The modules implement the primal simplex, the dual simplex, and the criss-cross method, respectively.

A module for solving linear programs from scratch.

A server module that allows one to verify LP-solutions over the web. (`http://www.mpi-sb.mpg.de/~funke`).

5 Conclusions

We have shown that the verify-and-repair strategy is able to solve medium to large LPs to optimality. The approach exploits the fact that existing inexact LP-solvers usually give a solution which is close to optimal. Our current approach fails, if this is not the case. We expect that our LPex-server will allow us to collect instances which cause difficulties with existing inexact solvers and/or with exact verification.

The source code of the LP solver will be available soon on `http://www.mpi-sb.mpg.de/~funke`.

References

[CPL] CPLEX. `www.cplex.com`.

[Gär98] Bernd Gärtner. Exact arithmetic at low cost—a case study in linear programming. In *Proceedings of the Ninth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 157–166, San Francisco, California, 25–27 January 1998.

[LW99] Leah Epstein, John Noga, Steve Seiden, Jiří Sgall and Gerhard Woeginger. Randomized online scheduling on two uniform machines. In *Proceedings of the Tenth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 317–326, N.Y., January 17–19 1999. ACM-SIAM.

[RT78] Rose, D. J. and Tarjan, R. E. Algorithm aspects of vertex elimination on directed graphs. *SIAM J. Appl. Math.*, 34(1):176–197, 1978.

[Wie86] Wiedemann, D.H. Solving sparse linear equations over finite fields. *IEEE Transactions on Information Theory*, IT-32(1):54–62, 1986.