A Comparison of Direct Linear Solvers for the Solution of the Linear Systems Arising in the Simplex Method

Robert Luce†, Jurjen Duintjer Tebbens‡, Jörg Liesen† and Reinhard Nabben†

† Institute of Mathematics, Technical University of Berlin

‡ Institute of Computer Science, Academy of Sciences of the Czech Republic

{luce,liesen,nabben}@math.tu-berlin.de, tebbens@cs.cas.cz

Abstract

THIS poster investigates the appropriateness of several popular modern direct solvers to solve the linear systems arising in the simplex method. It presents and comments the results of numerical experiments with different software packages for a large number of benchmark LP's.

1. Linear Systems in the Simplex Method

LET $A \in \mathbb{R}^{m\times n}$ with $m < n$, $l', u', c \in \mathbb{R}^n$ and $l'', u'' \in \mathbb{R}^m$. The linear programming problem (LP) consists of finding $x \in \mathbb{R}^n$ that solves

$$\begin{aligned} &\min\ c^T x \\ \text{subject to}\quad & l' \leq x \leq u' \\ & l'' \leq Ax \leq u''. \end{aligned}$$

Consider index sets B, N with $B \cup N = \{1, \ldots, m\}$, $B \cap N = \emptyset$, $|B| = m$ (and $|N| = n - m$) and $A_{\cdot\, B}$ nonsingular. Then $A_{\cdot\, B}$ is called basis matrix and an example of a simplex algorithm is

The entering simplex algorithm

1. *Pricing*: If the current solution cannot be improved, terminate. Else: choose an entering index $q \in N$ of a column that should enter the basis in order to improve the current solution. Depending on the pricing strategy, a linear system with $A^T_{\cdot\, B}$ must be solved.
2. Compute the primal search direction Δf from $A_{\cdot\, B}\Delta f = A_{\cdot\, q}$.
3. *Ratio test*: Compute the primal step length and select a leaving index $p \in B$ to leave the basis.
4. Compute the pricing search direction Δg from $A^T_{\cdot\, B}\Delta h = e_p$ and the computation of $\Delta g = A^T \Delta h$.
5. *Update*: $B = B \setminus \{p\} \cup \{q\}$, $N = N \setminus \{q\} \cup \{p\}$

In all implementations of the simplex method each of the individual vertex traversals requires the solution of two linear systems. Depending on the strategy chosen, the solution of a third or even fourth system per iteration might be necessary. The solution of these systems account for the major slice of computation time: 60-90%.

- In every iteration, one column of the basis matrix is replaced with a nonbasic column from A.
- Matrices are non-symmetric, indefinite and seem to lack any structure.
- The vast majority is sparse, that is, most linear programming problems have about 10 to 20 nonzeros per column
- The basis matrix typically contains an important number of unit vectors (25-50%); especially during the first iterations, unit vectors can make a very large part of the basis.

However, the basis matrices do have the following structure: There exist permutation matrices P, Q such, that

$$PA_{\cdot\, B}Q = \begin{pmatrix} U^0 & * & * \\ 0 & L^0 & 0 \\ 0 & * & N \end{pmatrix}, \qquad (1)$$

where L^0 is lower and U^0 is upper triangular. For most problems the dimension of the nucleus N is *much* smaller than the dimension of $A_{\cdot\, B}$. For example, for most very large LP's we consider, the average dimension of the nucleus N is only one or two percent of the dimension of $A_{\cdot\, B}$.
Some modern direct solver packages fail to recognize this structure and produce a fill-in of several times the fill-in needed with the above permutations. In fact, with the above permutations fill-in can arise only in the nucleus. If we measure, for given L and U factors, the fill-in by

$$\frac{nnz(L - I) + nnz(U)}{nnz(A_{\cdot\, B})}, \qquad (2)$$

then for our very large LP's the average fill-in has a value of about $1.01 - 1.02$, that is nearly optimal fill. In the following section we compare, for a representative set of direct solvers, the fill-in produced in the nucleus only (hence in (2), $A_{\cdot\, B}$ is replaced by N.

2. Numerical Experiments

The LP's we used come from four sources:

1. The NETLIB set of real-world LP's (94 LP's). Although this publicly available test set dates back many years and most of the problems are solved within a fraction of a second, we consider them to be interesting for our purpose of comparing fill-in in the factors.
2. The MIPLIB 2003 test set of mixed-integer linear programs (60 LP's). In order to mimic a typical root relaxation for branch-and-bound based MIP algorithms, we solve the resulting LP after applying CPLEX[1] MIP-presolve and relaxing integrality constraints.
3. The LP's from the Mittelmann benchmark of free LP solvers that do not come from source 1 or 2 (35 LP's).
4. Large scale LP's, mostly with the dimension of B exceeding $5 \cdot 10^5$, provided by Thorsten Koch of ZIB[2] (11 LP's).

In total, our test set consists of 200 LP's, representing a wide range of different application areas, so that the numerical results we present are tightly coupled with the behaviour encountered in practice.

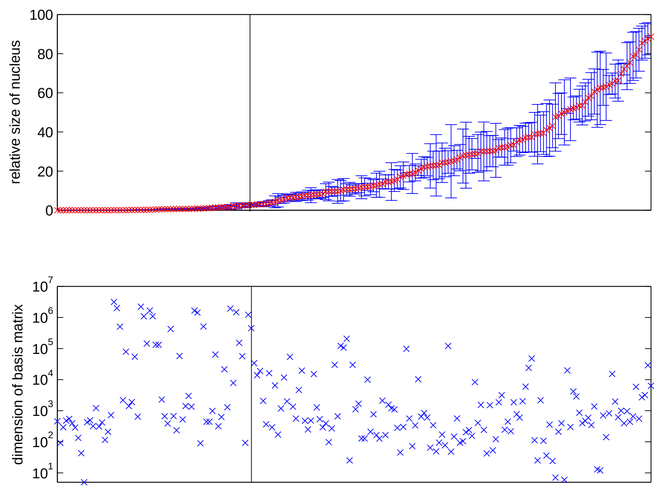


Figure 1: *Ranges of the nuclei sizes of the LP's we tested. The LP's are sorted according to nucleus size. The lower graph indicates the dimension of the corresponding LP.*

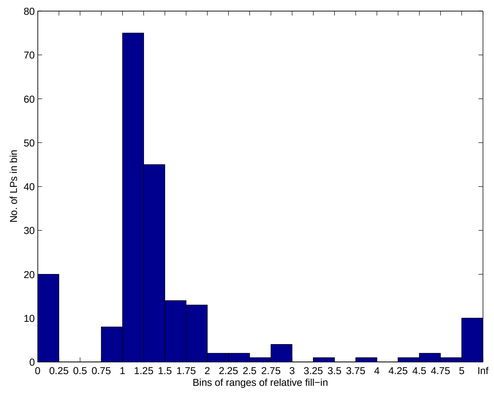
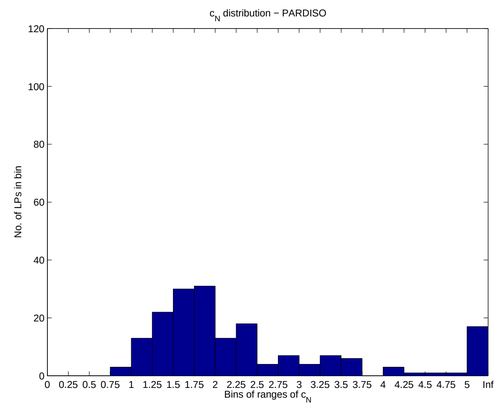
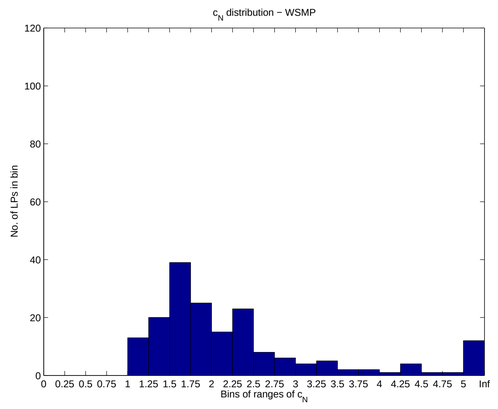
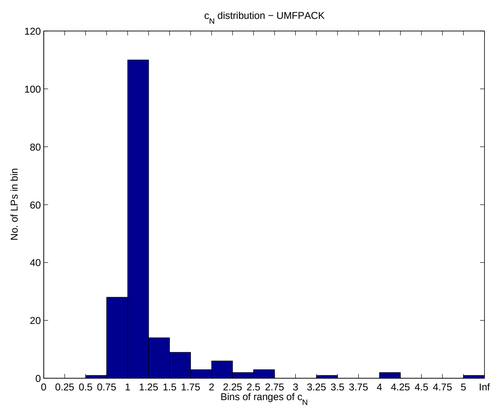


Figure 2: *Distribution of the LP's by their average fill-in (2) with Markowitz pivoting [1] in the nucleus at iterations where the basis matrix was refactorized.*

The distributions of the LP's by their average fill-in (2) in the nucleus with respectively, PARDISO 3.1[3], WSMP 6.9.25[4] and UMFPACK 5.0.1[5], are:







3. Conclusion

AMAZINGLY, the oldest strategy, Markowitz pivoting, outperforms the other solvers. We briefly describe it. In an LU decomposition algorithm we have to choose a pivot element from our active submatrix which we call B. At some iteration, let r_i be the number of nonzeros in row i and c_j the number of nonzeros in column j. Clearly, if b_{ij} was chosen as pivot element, the *Markowitz number*

$$m_{ij} = (r_i - 1)(c_j - 1)$$

is an upper bound for the number of fill-in elements. Among the elements that meet a certain stability threshold, Markowitz pivoting chooses the element with smallest Markowitz number. Markowitz pivoting (1) per definition recognizes the structure (1); (2) is based on a simple heuristic that copes with the unstructured kernel.

Acknowledgements: The work of the first and third author was supported by the Emmy Noether-Programm of the Deutsche Forschungsgemeinschaft. The work of the second author was supported by the MATHEON Deutsche Forschungsgemeinschaft Research Center, Berlin and by the Program Information Society under project 1ET400300415.

References

[1] Markowitz, H.M. *The elimination form of the inverse and its application to linear programming*. Management Sci. 3, pp. 255-269, (1957).

[1] http://www.ilog.com/products/cplex
[2] http://www.zib.de
[3] http://www.computational.unibas.ch/cs/scicomp/software/pardiso/
[4] http://www-users.cs.umn.edu/ agupta/wsmp.html
[5] http://www.cise.ufl.edu/research/sparse/umfpack/