On the Solution of the Linear Systems arising in the Simplex Method with Modern Linear Solvers

Jurjen Duintjer Tebbens

Institute of Computer Science,

Academy of Sciences of the Czech Republic

Workshop on Linear Algebra and Optimization, Birmingham,
September 13, 2007

joint work with

Joerg Liesen, Robert Luce and Reinhard Nabben

(Project B17 of the Matheon Center, Berlin)

1. Linear systems in the Simplex Method

We consider a ,,linear program'' (LP):

- Let $A \in \mathbb{R}^{m \times n}$ with $m < n$ be the constraint matrix
- Let b be the right hand side vector, c be the cost vector and s be the vector of slack variables
- We search for $x \in \mathbb{R}^n$ solving the optimization problem

$$
\begin{array}{lrcl}
\text{max} & & & c^T x \\
\text{s. t.} & (A, I_m)\binom{x}{s} & = & b \\
 & x, s & \geq & 0\,.
\end{array}
$$

The simplex method to solve LP's is considered one of the top ten algorithms of the 20th century [SIAM News - 2000].

In the simplex method an optimal solution is found by traversing a sequence of (neighboring) vertices of the polyhedron defined by the linear constraints of the LP.

For a set of column indices B let $(A, I_m)_{\cdot B}$ be the basis matrix and let N denote the non-basic column indices of A.

A statement of the (Dual) Simplex Algorithm, focusing on the computational steps, is as follows:

A Dual Simplex Algorithm

1. *Pricing*: If the current solution cannot be improved, terminate. Else choose a leaving index $p \in B$ of column to leave the basis matrix.
2. Solve $(A, I_m)^T_{\cdot B} \Delta h = e_p$
3. *Ratio test*: Based on the previous solution, select an entering index $q \in N$ of a column that should enter the basis matrix in order to improve the current solution.
4. Solve $(A, I_m)_{\cdot B} \Delta f = (A, I_m)_{\cdot q}$
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: one with the basis matrix $(A, I_m)_{\cdot B}$ and one with its transpose.

- In every iteration, one column of the basis matrix is replaced with a nonbasic column from A.

Depending on the strategy chosen, the solution of a third or even fourth system per iteration might be necessary.

In one run of the simplex method typically hundreds of linear systems have to be solved.

The solution of these systems accounts for the major slice of computation time: 60-90%.

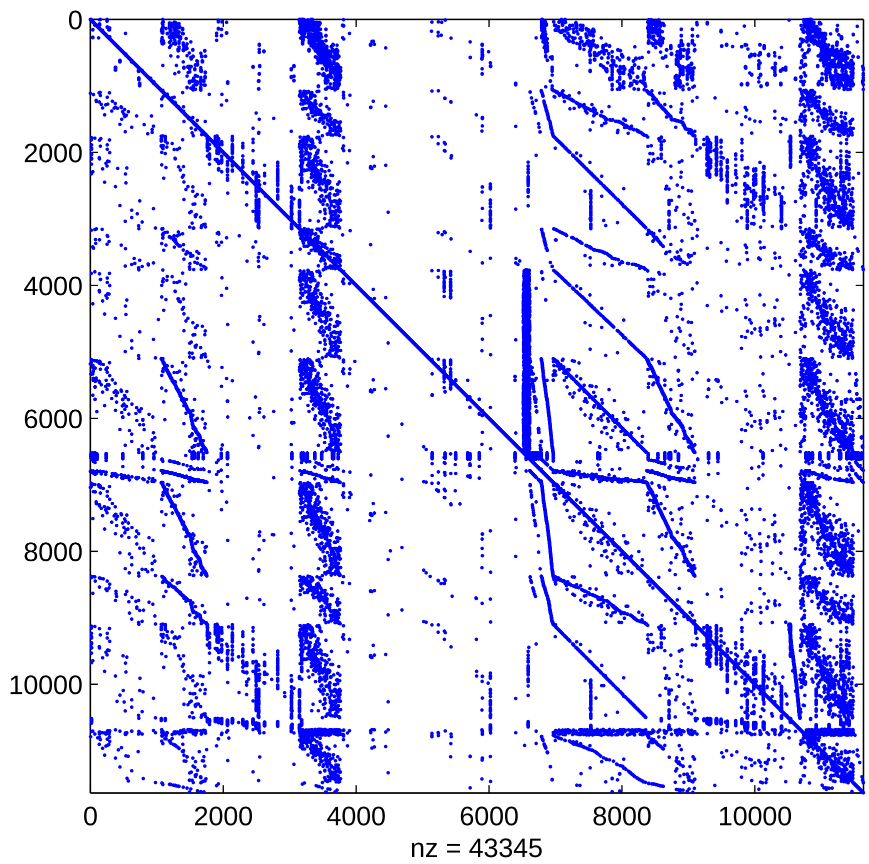
Modern implementations of the Simplex Method all use a direct solution strategy, Markowitz pivoting, dating back from the fifties [Markowitz - 1957] !

Can modern linear algebra provide tools that improve the performance of the linear algebra kernel of LP codes ?

2. Efficient Solution

Properties of basis matrices $(A, I_m)_{.B}$:

- Non-symmetric and indefinite.
- The vast majority is sparse, that is, most constraint matrices 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.
- The 2-norm condition number of an optimal basis matrix may have values of 10^6 to 10^8 .
- The matrices have unpredictable structure; at first sight they seem to lack any structure.



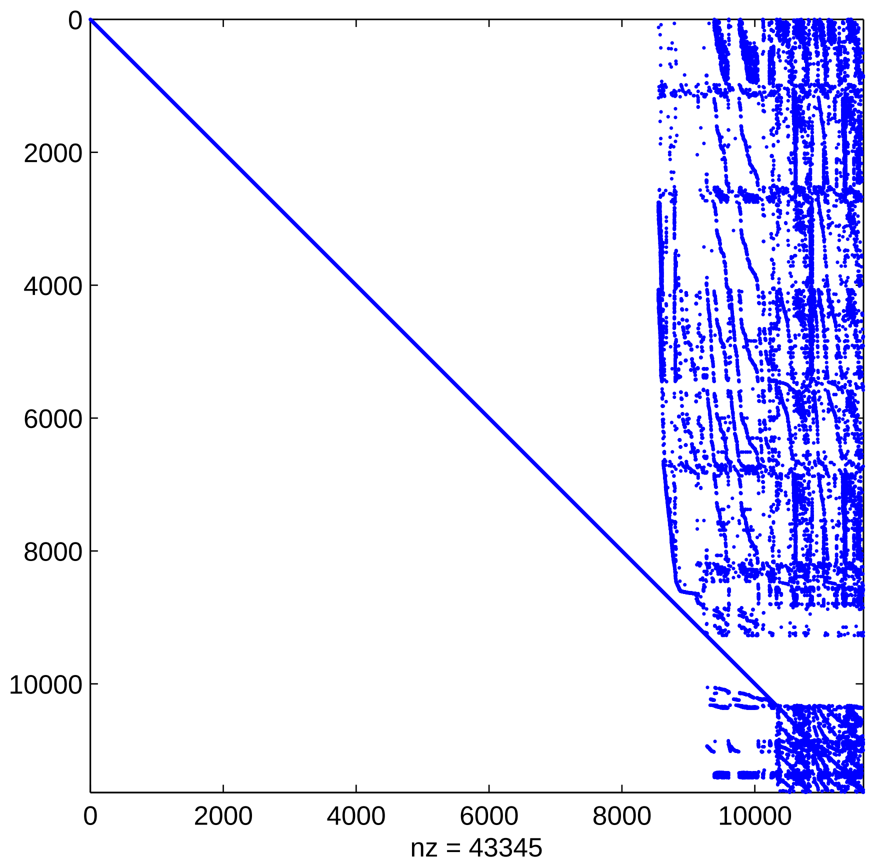
Structure of typical basis matrix (momentum1 from MIPLIB)

However, it is common practice in LP codes to perform a pre-processing step by successively moving column and row singletons to the front (triangulation).

We obtain permutations such that the permuted basis matrix is of the form

$$P A_{\cdot B} Q = \begin{pmatrix} U^0 & * & * \\ 0 & L^0 & 0 \\ 0 & * & N \end{pmatrix},$$

with an upper triangular matrix U^0, a lower triangular matrix L^0 and a matrix N, called the *nucleus* (or *kernel*).



Basis matrix after preprocessing

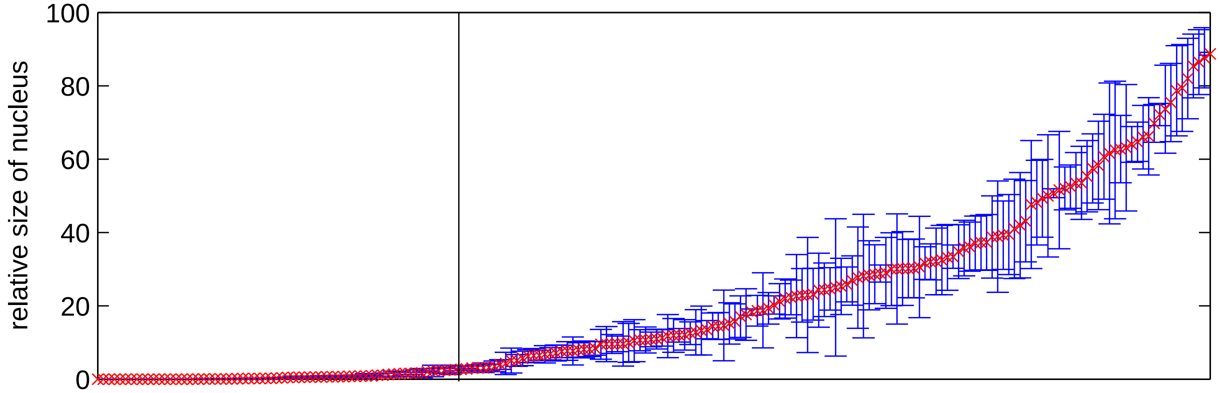
Based on empirical observation, we found that the size of the nucleus is in general *very* small. This effect is the stronger the larger are the considered LP's. Consider for example the following large-scale LP's provided by ZIB:

LP name	no. fact.	no. N	size B	Ø size N	Ø deviation
BER_P*od10	1979	1978	1425456	11519	2298
N_BA2*mann	4913	107	3160202	43	31
aflow_1*50	2658	210	500998	584	436
aflow_2*50	10382	1374	2001998	488	705
scm30*0pre	6016	6013	1220936	31156	6997
ts.lo*0315	913	911	1654588	3684	1779
ts.lo*2029	664	663	1089131	1846	765
ts.lo*2253	739	738	1089128	3186	2037
ts.lo*4012	1368	1366	1654588	12762	10819
ts.lo*4139	1006	1005	2214771	3713	2245

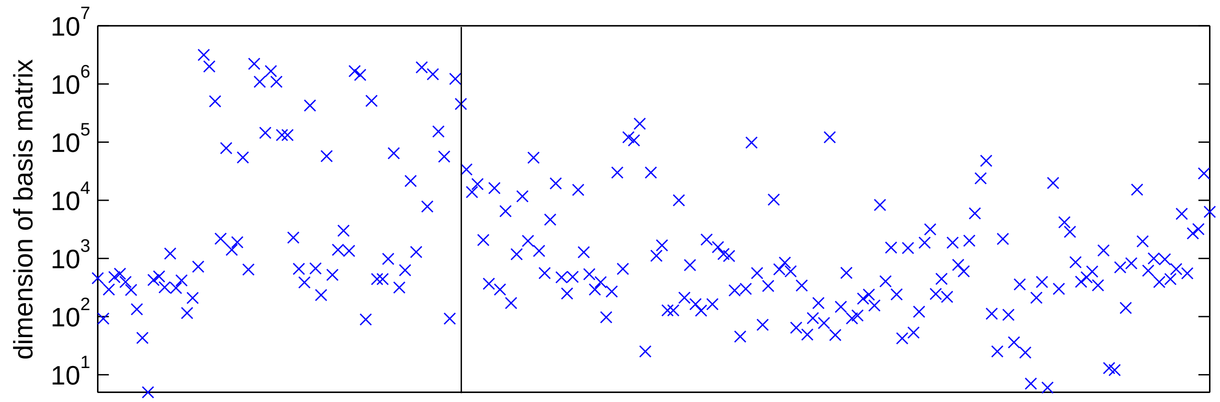
We tested in total 200 LP's (with 392.701 factorizations) from four sources:

1. The NETLIB set of real-world LP's (94 LP's).
2. The MIPLIB 2003 test set of mixed-integer linear programs (60 LP's).
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 $5 * 10^5$, provided to us by ZIB (11 LP's).

They represent a wide range of different application areas, so that the numerical results we present are tightly coupled with the behaviour encountered in practice.



relative size of nucleus
0
20
40
60
80
100



dimension of basis matrix
10^1
10^2
10^3
10^4
10^5
10^6
10^7

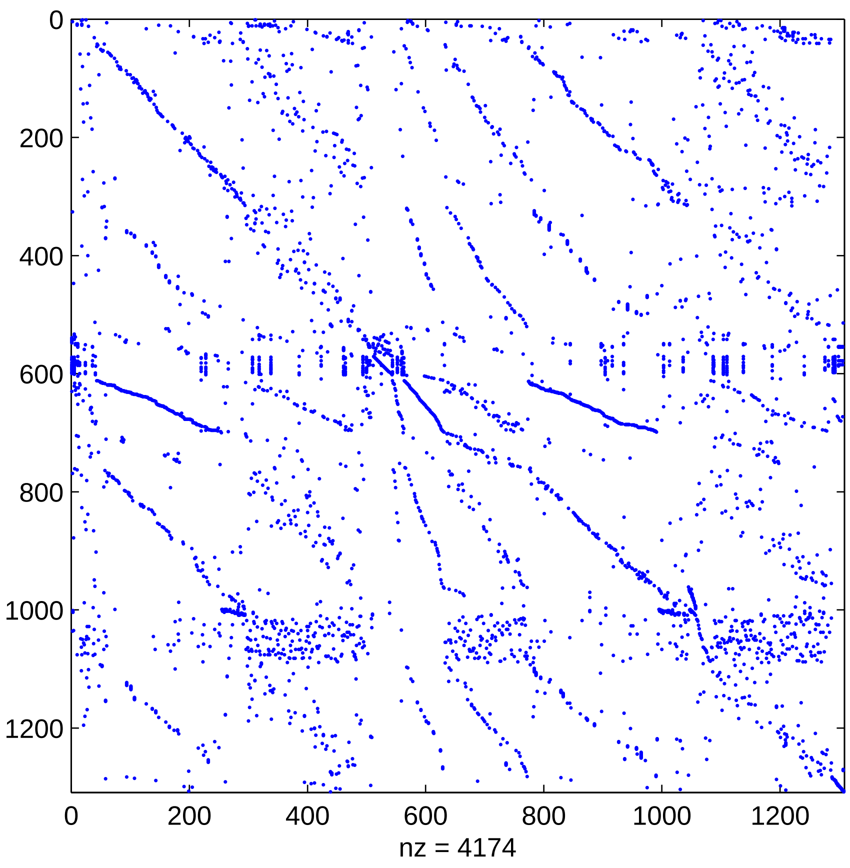
These observations have important consequences:

- Many LP basis matrices have the pronounced, unusual structure:

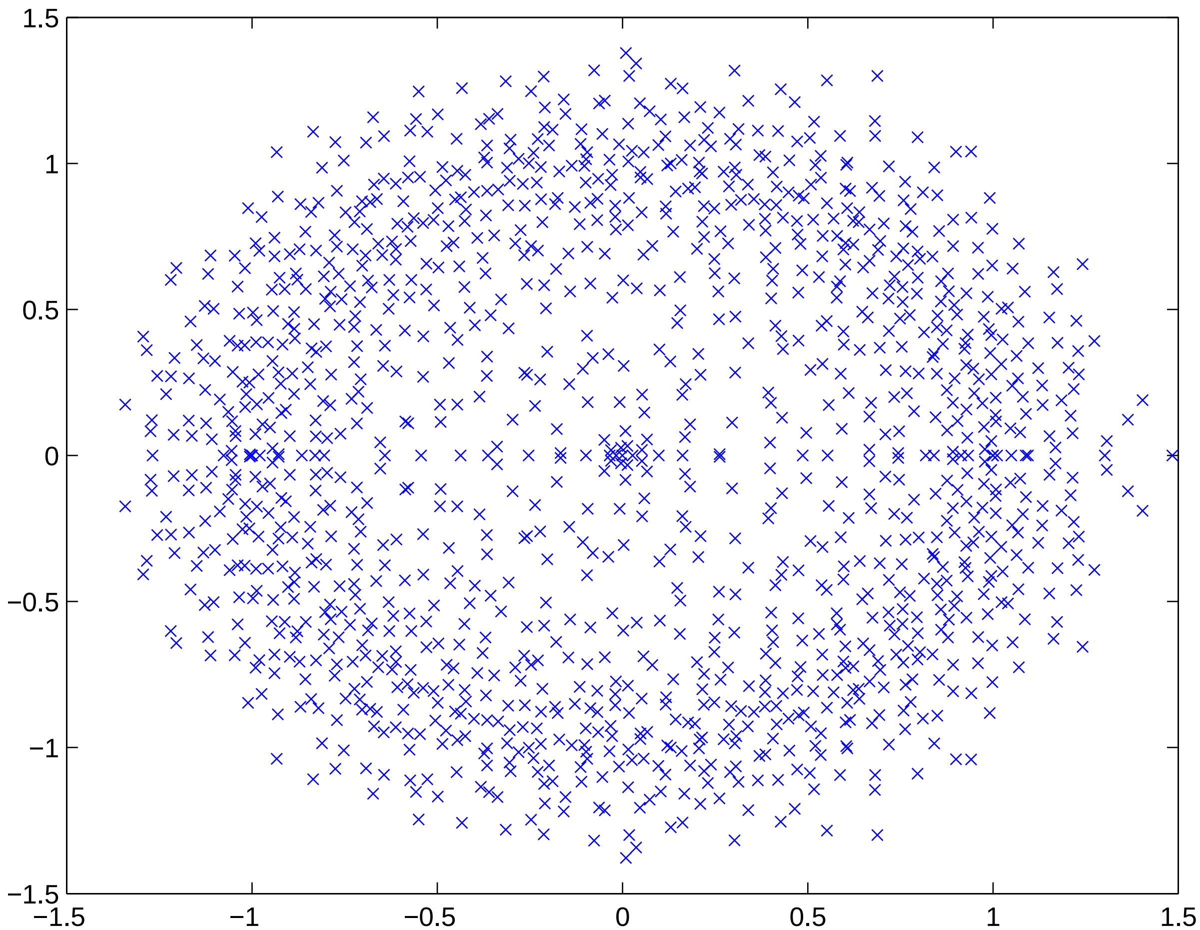
$$\begin{pmatrix} U^0 & * & * \\ 0 & L^0 & 0 \\ 0 & * & N \end{pmatrix}.$$

- The triangulation is crucial for efficient solution; only a system with the nucleus N remains to be solved.
- The sizes of nuclei rarely exceed 10^5; solution of the corresponding linear systems is more or less trivial for modern linear algebra and may be seen as mere ,,cleaning up''.

Still we can ask: What solution method is best for systems with the nucleus?



Structure of nucleus



Spectrum of nucleus

Are iterative methods competitive with direct methods?

- The constraint matrix A is explicitly stored; the advantage of matrix-free implementation does not apply here
- We need ,,exact'' solutions
- The nuclei N are completely unstructured and their spectra indicate rather unfavorable convergence behavior of iterative methods, unless a very good preconditioner is used.
- Even in case such preconditioner is obtained "for free", our results show the fill-in in the LU factorization of N may be very low, which implies that a preconditioned iterative solver would have to compute a good approximate solution within very few iterations.

Not competitive in the setting of Simplex-based LP solvers !

We will next compare different LU-codes by the amount of fill they produce only. A greater number of nonzeros outweighs any performance gain from fast factorization.

Note basis matrices are factorized only periodically and the factorizations are being updated in between. This is straightforward because basis matrices differ by one column only (e.g. Forrest-Tomlin updates). They do create some fill, though.

Fill-in during LU-factorization is restricted to the fill created inside the nucleus. Thus in many cases we have close to optimal fill (only a few percents) for the whole basis matrix.

Markowitz pivoting

During the (sparse) LU decomposition algorithm we have to choose a pivot element from our active submatrix $\hat{A}$.

- Let r_i be the number of nonzeros in row i
- Let c_j be the number of nonzeros in column j
- When $\hat{a}_{ij}$ is pivot, the Markowitz number

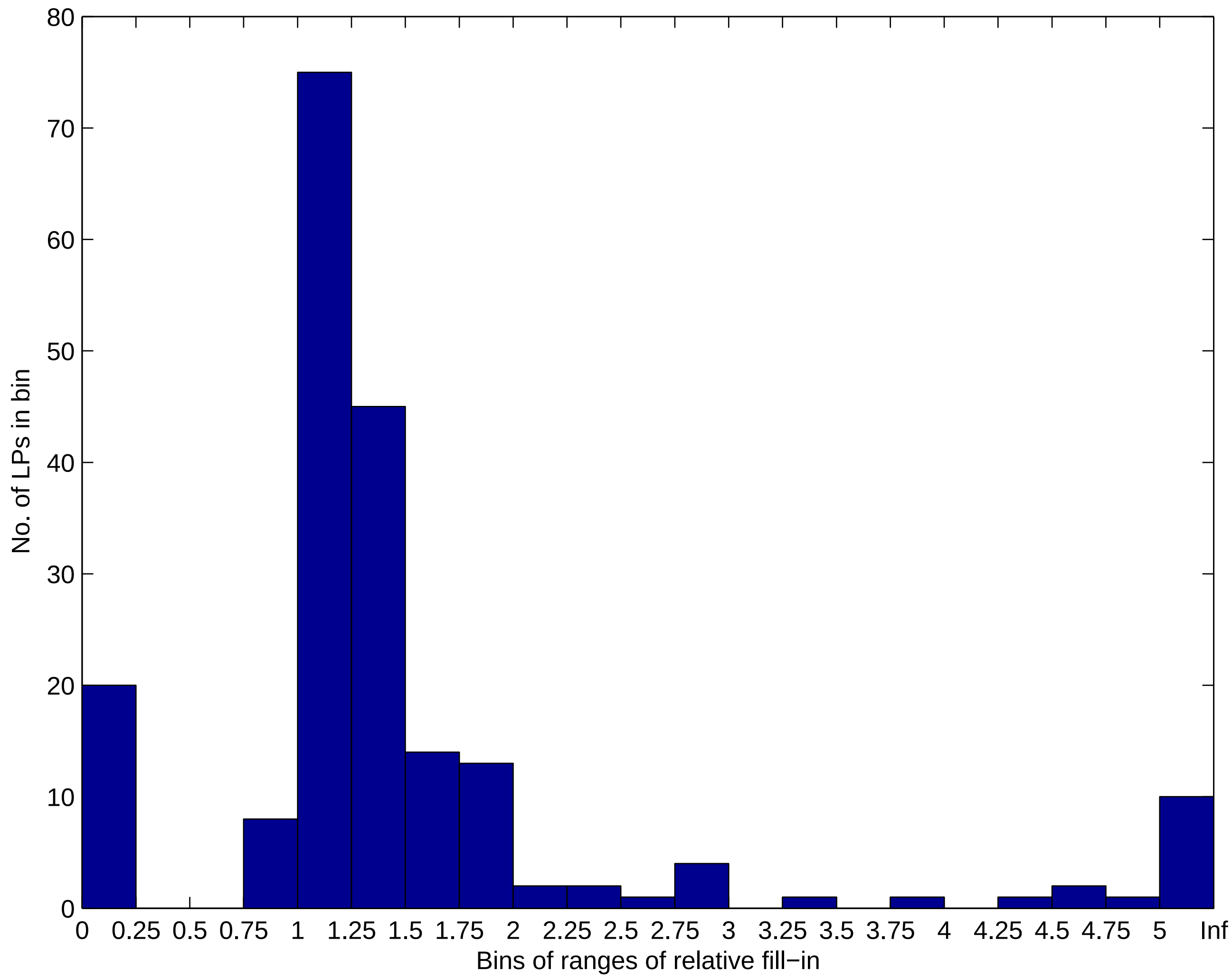
$$m_{ij} = (r_i - 1)(c_j - 1)$$

  is an upper bound for the number of fill-in elements
- Among elements that satisfy, for a threshold ρ,

$$\left|\hat{a}_{ij}\right| \geq \rho \max_{k \in J} |\hat{a}_{ik}|$$

  we choose an element with smallest Markowitz number.

Note Markowitz identifies the nucleus.

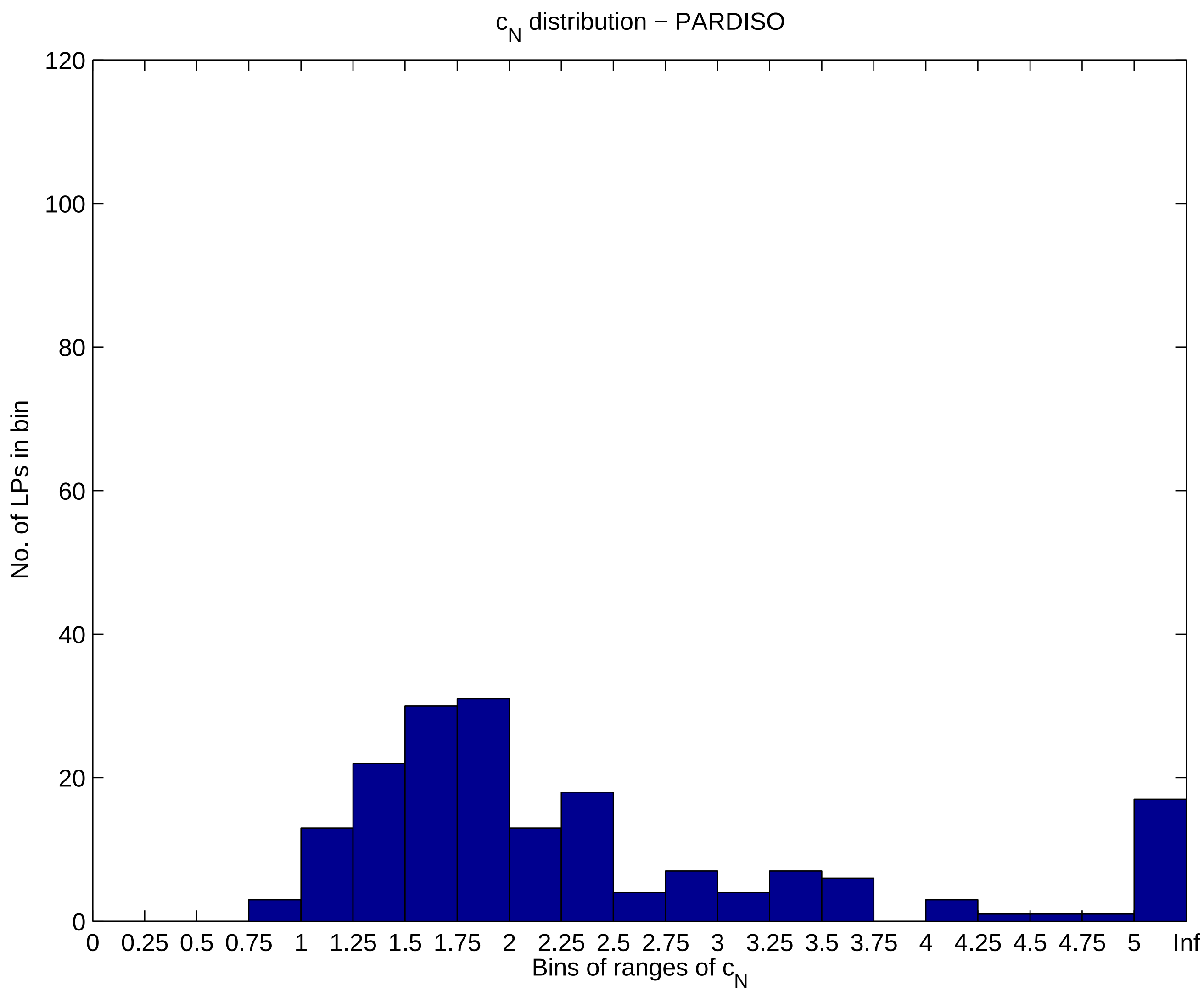

80
70
60
50
40
30
20
10
0
No. of LPs in bin
0 0.25 0.5 0.75 1 1.25 1.5 1.75 2 2.25 2.5 2.75 3 3.25 3.5 3.75 4 4.25 4.5 4.75 5 Inf
Bins of ranges of relative fill−in

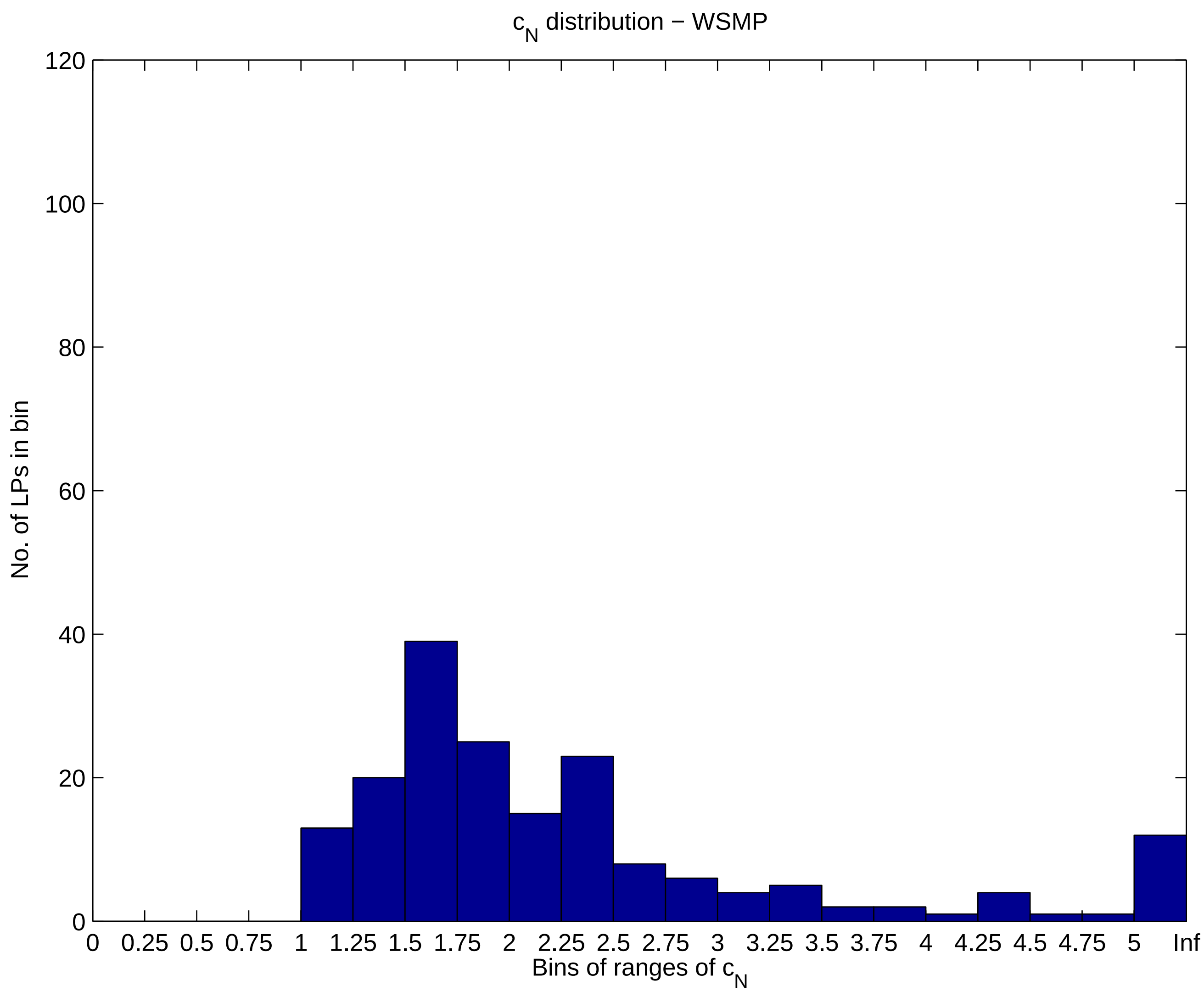
We compared Markowitz pivoting (1957) with more modern LU-factorization techniques (last two decades):

- UMFPACK (right looking multifrontal)
- PARDISO (left/right supernodal with nested dissection)
- WSMP (multifrontal with nested dissection, Block Triangular Form (BTF))

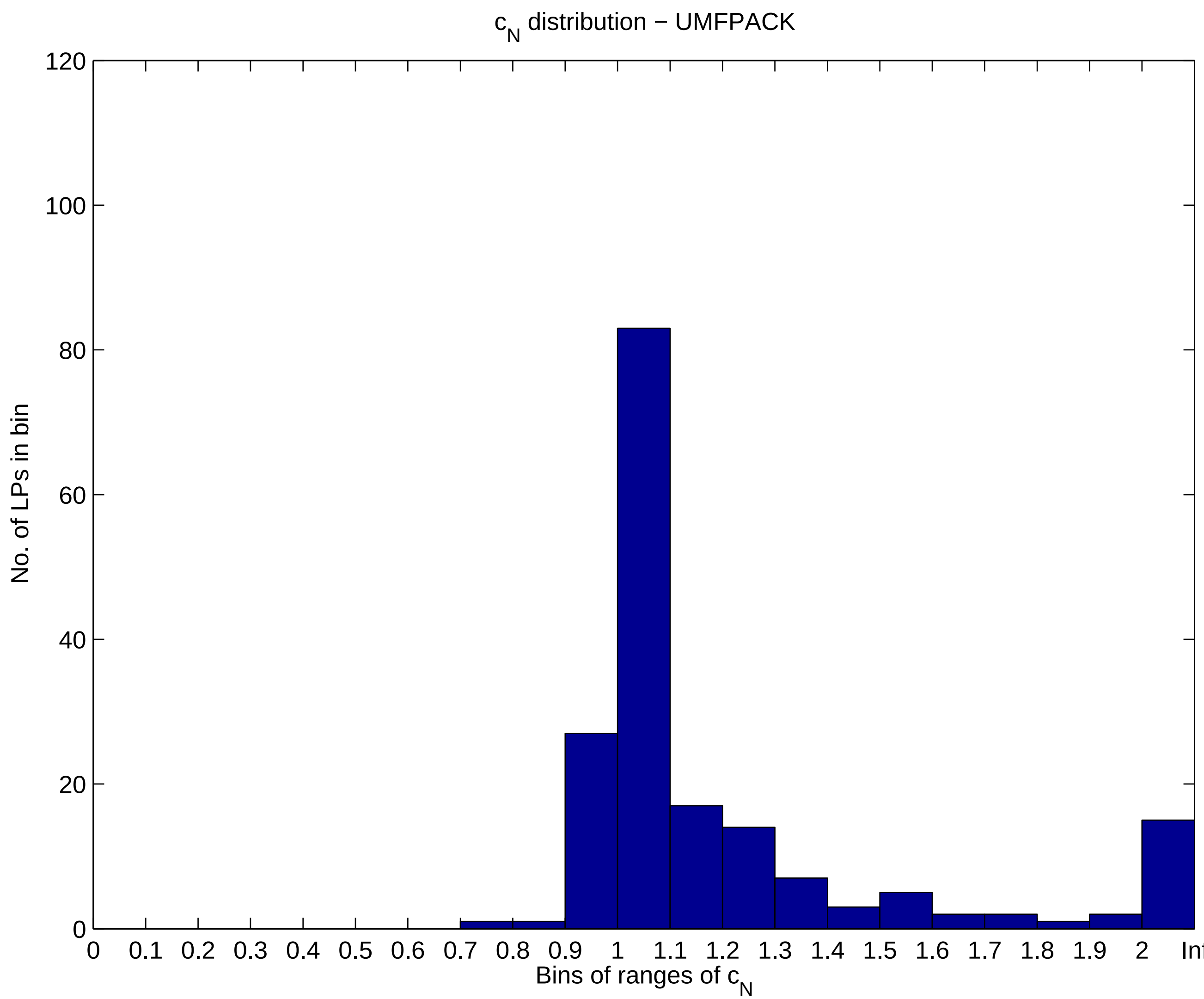
In the following,

$$c_N \equiv \frac{nnz(I - L) + nnz(U)}{nnz(I - L_{Mark}) + nnz(U_{Mark})}\,.$$


c_N distribution − PARDISO
No. of LPs in bin
0
20
40
60
80
100
120
0 0.25 0.5 0.75 1 1.25 1.5 1.75 2 2.25 2.5 2.75 3 3.25 3.5 3.75 4 4.25 4.5 4.75 5 Inf
Bins of ranges of c_N



c_N distribution − WSMP
No. of LPs in bin
0
20
40
60
80
100
120
0 0.25 0.5 0.75 1 1.25 1.5 1.75 2 2.25 2.5 2.75 3 3.25 3.5 3.75 4 4.25 4.5 4.75 5 Inf
Bins of ranges of c_N


c_N distribution − UMFPACK
No. of LPs in bin
0
20
40
60
80
100
120
0 0.1 0.2 0.3 0.4 0.5 0.6 0.7 0.8 0.9 1 1.1 1.2 1.3 1.4 1.5 1.6 1.7 1.8 1.9 2 Inf
Bins of ranges of c_N

Good old Markowitz outperforms all modern solvers (but UMFPACK is close)! This may be due to the

- high sparsity of the nucleus but lack of structure of the nucleus
- fact that Markowitz is a cheap nonsymmetric minimum degree heuristic
- low degree of structural symmetry which may cause bad performance of **colmmd, colamd** etc.
- no exploitation of BTF possible

Conclusion

- Markowitz pivoting is ideal because (1) it detects the nucleus (2) inside the nucleus it yields near minimal fill-in

- Hence LA has not come up with an improvement to solve the Simplex method linear systems for 50 years

- positively said: Already many decades ago, LA provided a near optimal strategy to solve the Simplex method linear systems, namely LU-factorization with Markowitz pivoting. Good news for both communities !

More details can be found in ,,On the linear algebra kernel of Simplex-based LP solvers'', by R. Luce, J. Duintjer Tebbens, J. Liesen and R. Nabben (to be submitted to Mathematical Programming).

Thank you for your attention.