

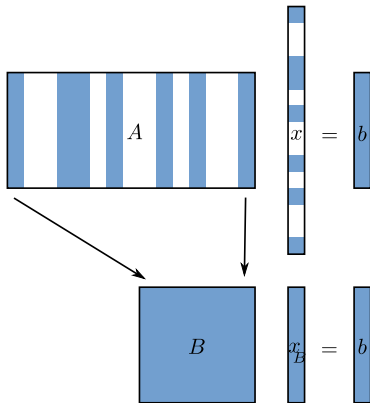
Permutations in the factorization of LP bases

Laurent Poirrier*

Joint work with Ricardo Fukasawa*

* Combinatorics & Optimization, UWaterloo

$$\begin{array}{ll} \min & c^T x \\ \text{s.t.} & Ax = b \\ & x \geq 0 \end{array} \quad (\text{LP})$$



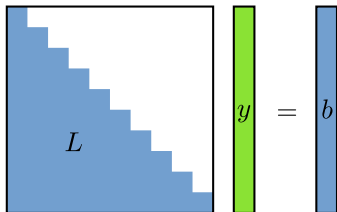
We want x in

$$Bx = b.$$

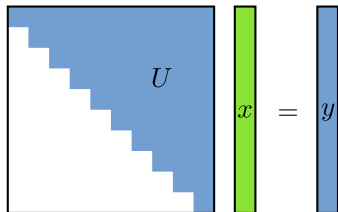
We find L, U triangular s.t. $LU = B$

$$\begin{aligned} LUx &= b \\ \underbrace{L(Ux)}_y &= b \end{aligned}$$

$$Ly = b$$



$$Ux = y$$



Why a custom LU code for LP?

B is **not**:

- ▶ symmetric and positive definite
- ▶ block structured
- ▶ band structured

B is a submatrix of A :

Property P1: “partially” triangular

Property P2: one column changes after each iteration

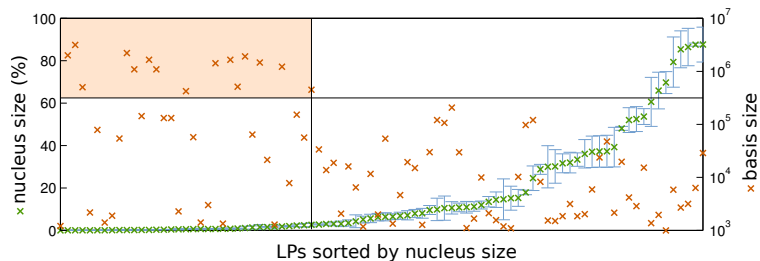
P1. Partially triangular permutation

\exists permutations P, Q s.t. $PBQ =$

$$\begin{pmatrix} \begin{array}{c} \text{blue trapezoid } U^p \\ \text{green staircase } L^p \\ \text{grey rectangle } G \end{array} \end{pmatrix} = \begin{pmatrix} \begin{array}{c} \text{dashed box } I \\ \text{green staircase } L^p \\ \text{dark green staircase } L^q \end{array} \end{pmatrix} \cdot \begin{pmatrix} \begin{array}{c} \text{blue trapezoid } U^p \\ \text{dashed box } I \\ \text{dark blue staircase } U^q \end{array} \end{pmatrix}$$

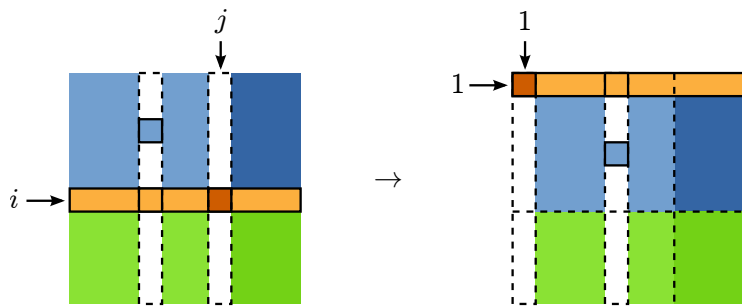
[See e.g. "Computing Sparse LU Factorizations for Large-Scale Linear Programming Bases", Suhl, Suhl, 1990]

P1. Size of the nucleus G



[“On the factorization of simplex basis matrices”, Luce, Duintjer Tebbens, Liesen, Nabben, Grötschel, Koch, Schenk, 2009]

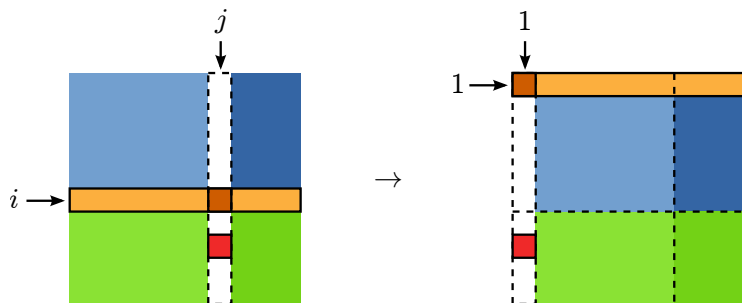
P1. Finding a partially triangular permutation



Proposition:

Permuting singletons to the front yields a minimal size nucleus.

P1. Finding a partially triangular permutation

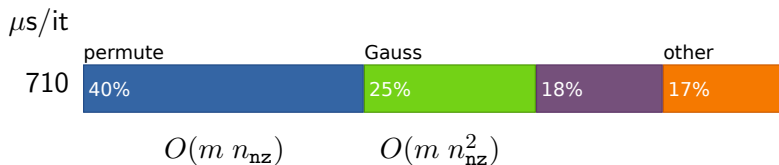


Proposition:

Permuting singletons to the front yields a minimal size nucleus.

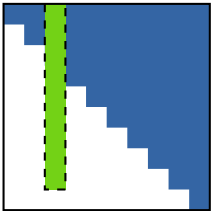
P1. Factorization time

MIPLIB 2010 root nodes:

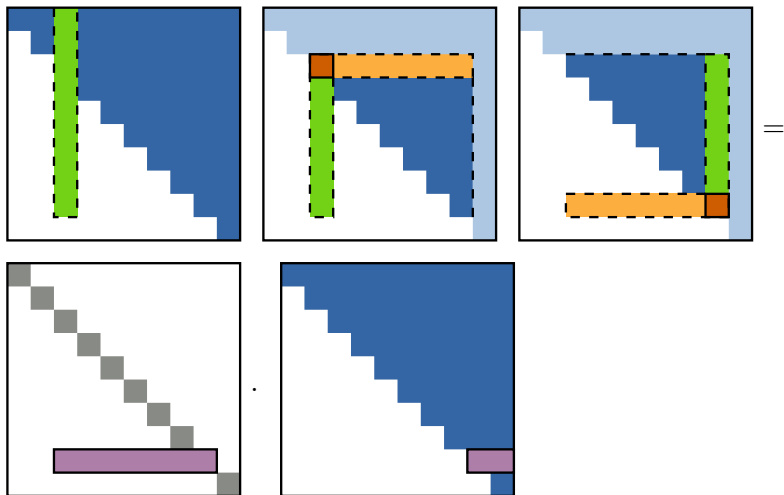


P2. Forrest-Tomlin

$$\begin{aligned} B' &= B - Be_i e_i^T + a_j e_i^T \\ &= LU - LUe_i e_i^T + a_j e_i^T \\ &= L \left(U - Ue_i e_i^T + L^{-1} a_j e_i^T \right) \\ &= L U' \end{aligned}$$

$$U' = \left(\begin{array}{c} \text{[Diagram of matrix } U' \text{]} \end{array} \right)$$
The diagram shows a square matrix with a blue upper triangular region. A vertical green dashed line is drawn through the matrix, representing the modification of the i -th column. The matrix is enclosed in large parentheses.

P2. Forrest-Tomlin



P2. Forrest-Tomlin, after k iterations

$$B = L \underbrace{\eta_1 \cdots \eta_k}_H U$$

$$\begin{aligned} B' &= B - B e_i e_i^T + a_j e_i^T \\ &= LHU - LHU e_i e_i^T + a_j e_i^T \\ &= LH \left(U - U e_i e_i^T + (LH)^{-1} a_j e_i^T \right) \\ &= LH U' \end{aligned}$$

P2. Refactorization time



Further improvement...

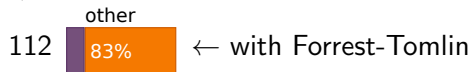
$$U' = \left(\begin{array}{c} \text{[Diagram of a matrix with a green dashed vertical line and a blue shaded region]} \end{array} \right)$$

The diagram shows a square matrix with a white background. A vertical dashed green line is positioned in the second column from the left. To the right of this line, a blue shaded region covers the upper-right portion of the matrix, forming a staircase pattern that descends from the top-left towards the bottom-right. The area below the staircase and to the left of the green line is white.

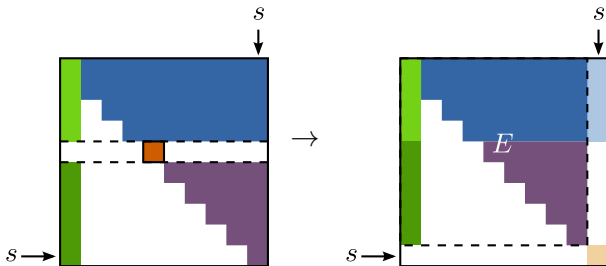
Reid's idea

- ▶ Finds a triangular permutation: 58% of iterations
- ▶ Running time:

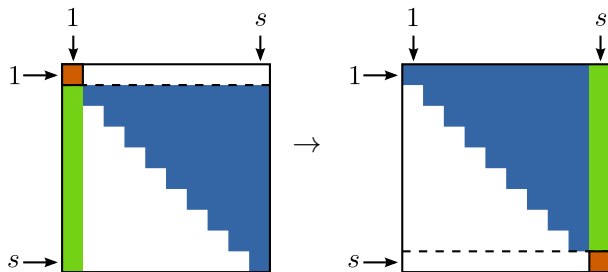
$\mu\text{s}/\text{it}$



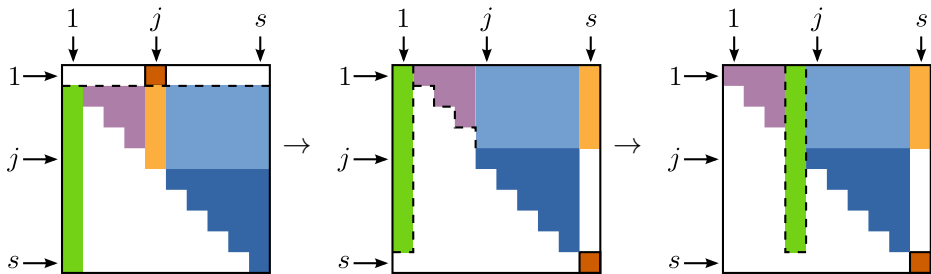
Assumption: no row-singletons in $2, \dots, s$



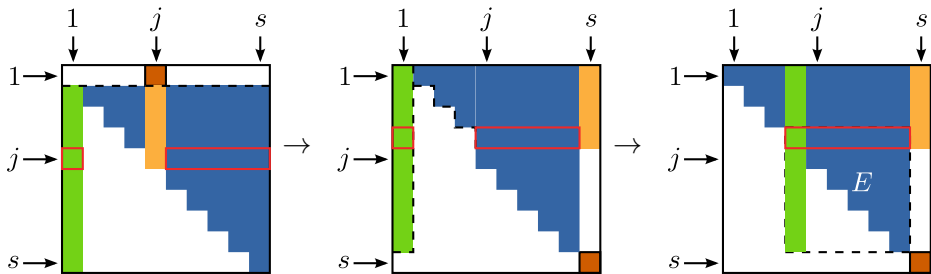
Case 1: E_{11} row-singleton



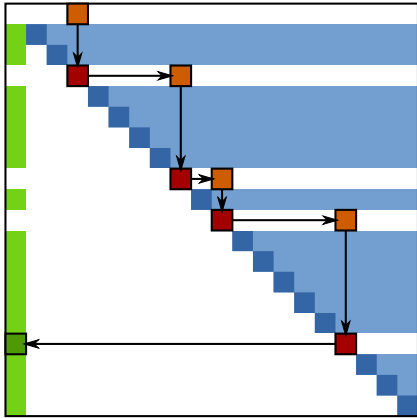
Case 2: E_{1j} row-singleton



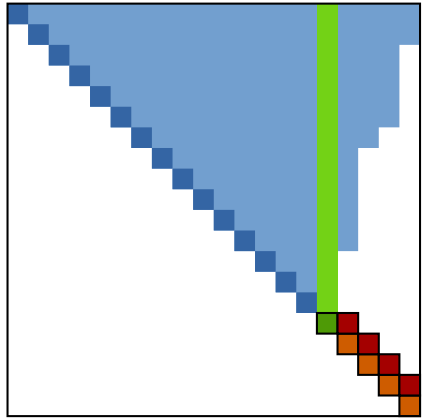
Case 2: E_{1j} row-singleton



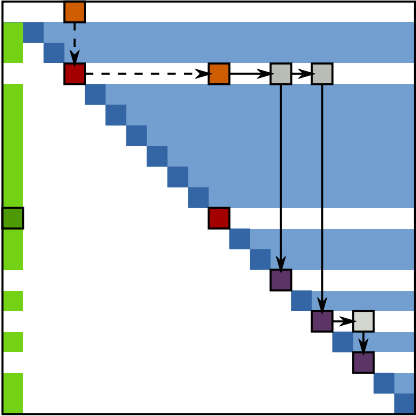
One-path



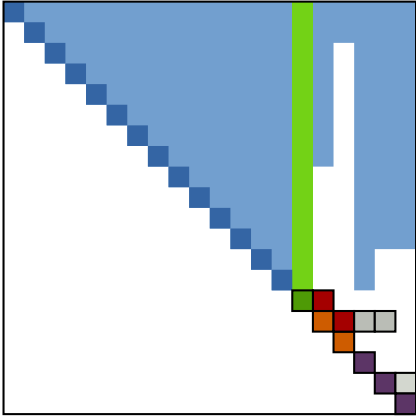
→



Dropping the assumption



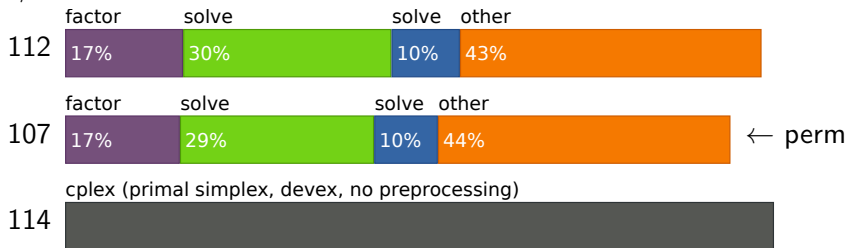
→



Results

- ▶ Finds a triangular permutation: 58% of iterations
- ▶ Running time:

$\mu\text{s}/\text{it}$



Why?

- ▶ Avoids Forrest-Tomlin
- ▶ Fewer η matrices
- ▶ Sparser rhs

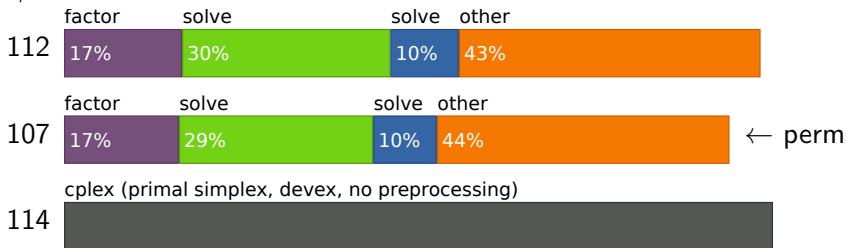
Summary

- ▶ A sparse method for permuting out the spike.
- ▶ Can replace Forrest-Tomlin in 58% of iterations.
- ▶ Decreases density of triangular solve intermediate vectors.

Results

- ▶ Finds a triangular permutation: 58% of iterations
- ▶ Running time:

$\mu\text{s}/\text{it}$



Benchmark framework

Instances:

MIPLIB 2010 benchmark set root nodes, after CPLEX MIP preprocessing.

In CPLEX:

- ▶ disable LP preprocessing
- ▶ force devex pricing
- ▶ force primal simplex

Comparison unfairness

Our code:

- ▶ trained and tested on same instances, same computer
- ▶ finetuned for minimal time per iteration
- ▶ does more iterations (deveX? tradeoff on sparsity?)
- ▶ all-logical starting basis (sparser!)

CPLEX:

- ▶ unusual setting (no LP preprocessing, forced deveX and primal)
- ▶ stricter constraints on robustness (numerical accuracy)

Important implementation caveat

Updating the permutation vectors:

- ▶ asymmetric row/column permutations $\rightarrow \sim 2\times$ more work
- ▶ sliding the mapping needs to be implemented carefully!
the number of pivots is usually small \rightarrow
 - ▶ sort pivots
 - ▶ innermost loop: slide chunks of contiguous indices by a constant number

This is like in F-T, but in F-T there is only one chunk and we slide indices by exactly 1.

