Parallel Sparse PLUQ Factorization modulo p

Charles Bouillaguet [1] Claire Delaplace [12] Marie-Emilie Voge [1]

[1] CFHP, CRIStAL, Université de Lille

[2] EMSEC, IRISA, Université de Rennes 1

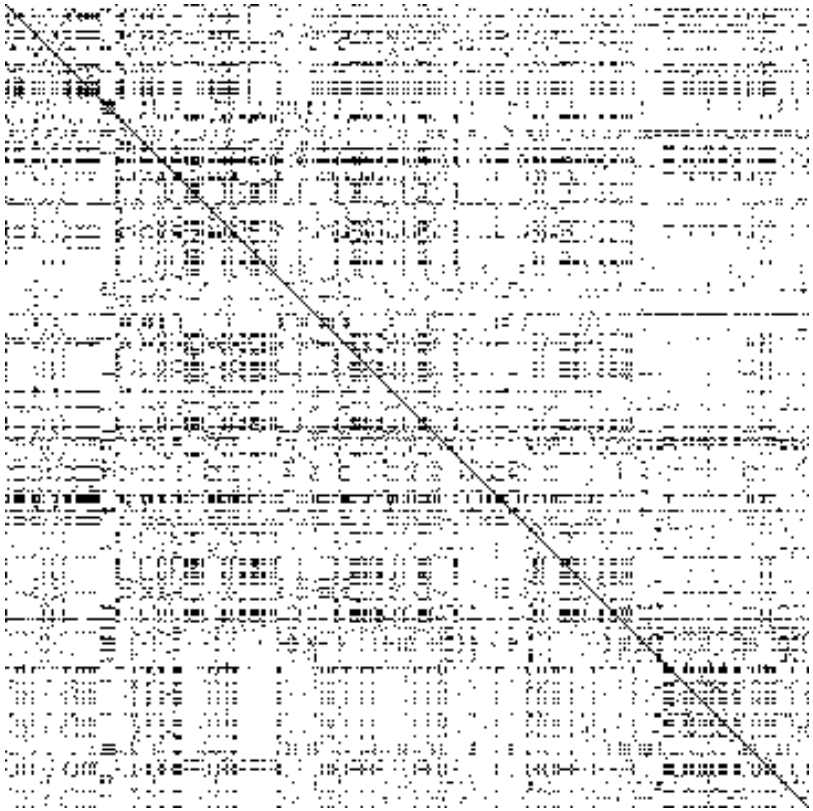
PASCO, July 2017

Background

Sparse Linear Algebra
Modulo p (coefficients : int)

Operations

- Rank
- Linear systems
- Kernel
- etc...

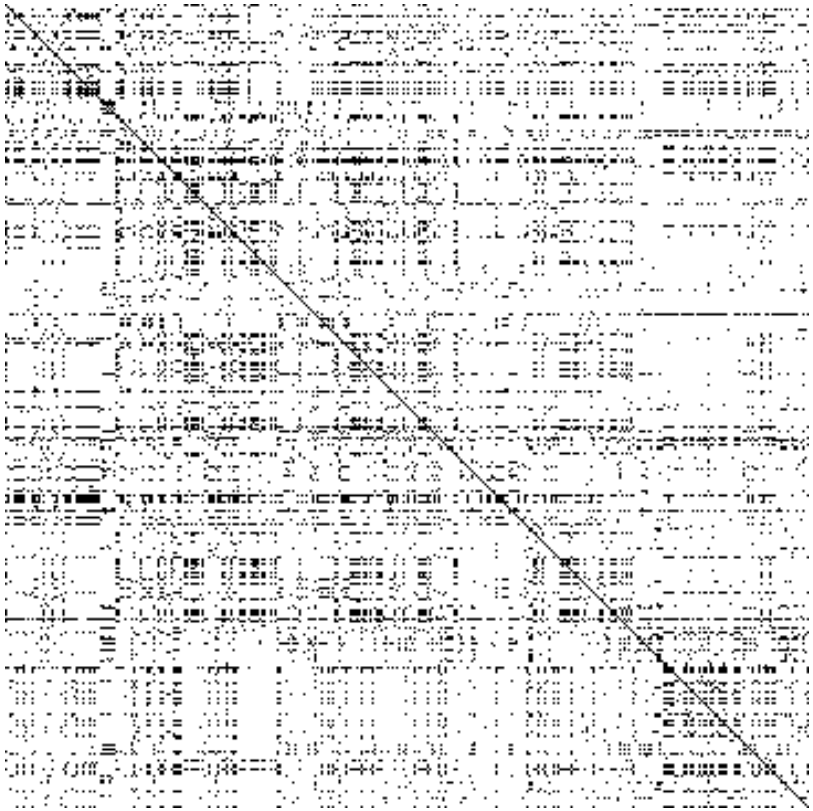


Background

Sparse Linear Algebra
Modulo p (coefficients : int)

Operations

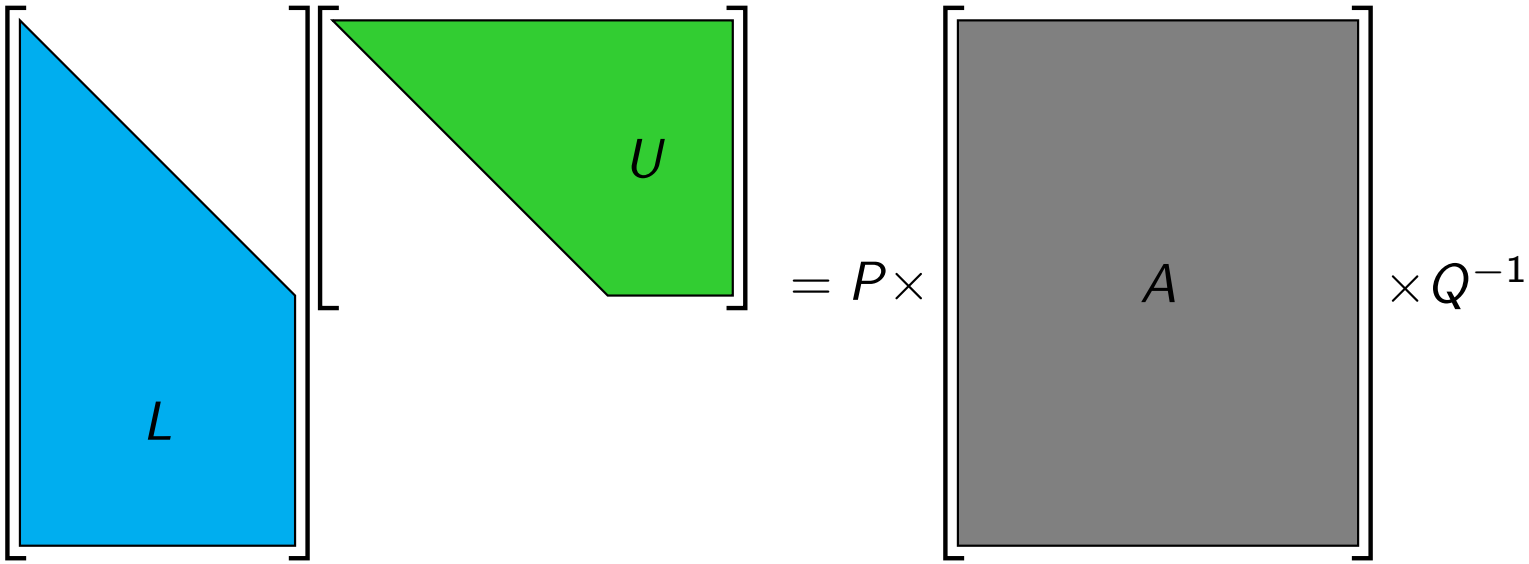
- Rank
- Linear systems
- Kernel
- etc...



Two families of Algorithms

- **Direct methods** (Gaussian Elimination, LU, ...)
- **Iterative methods** (Wiedemann, Lanczos...)

PLUQ Factorization



- L has non zero diagonal
- U has unit diagonal
- A can be rectangular
- A can be rank deficient

The Sparse PLUQ Algorithm from [BD16]

Description

- Find many pivots without performing any arithmetical operations
- Compute the Schur complement S
- Compute the rank of S

The Sparse PLUQ Algorithm from [BD16]

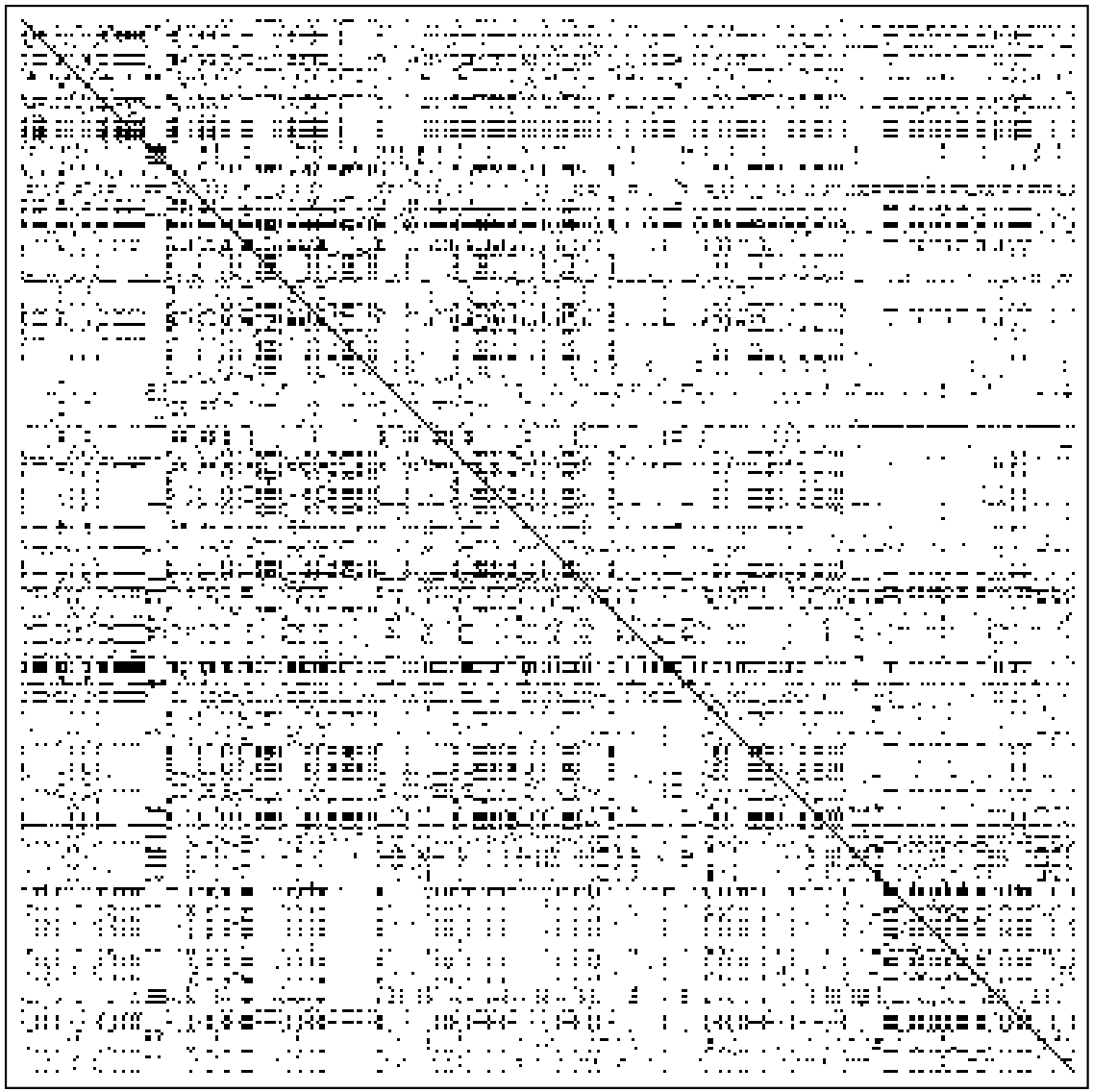
Description

- Find many pivots without performing any arithmetical operations
- Compute the Schur complement S
- Compute the rank of S

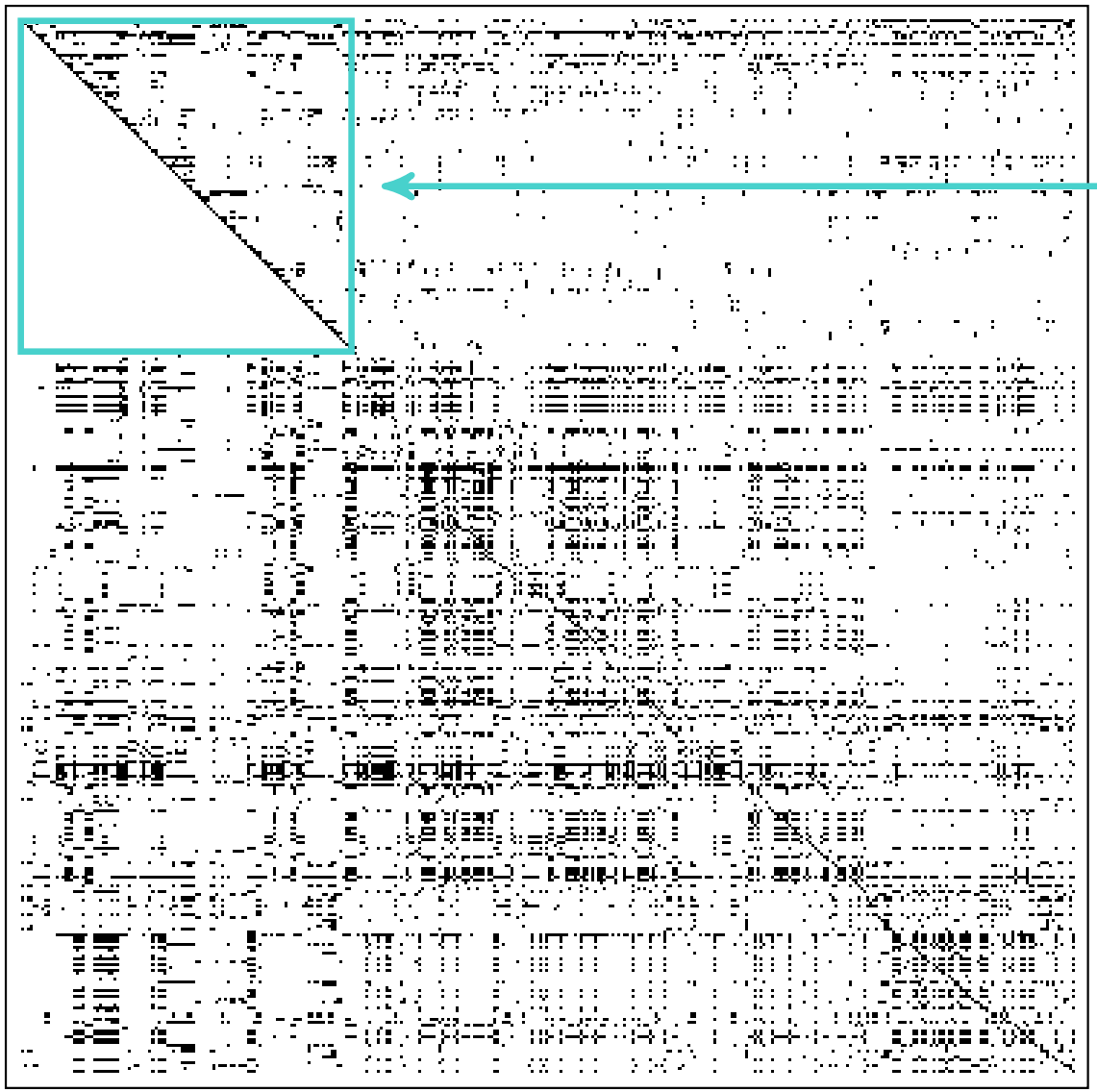
Rank of S

- Recurse
- Dense rank computation
- Wiedemann Algorithm
- ...

Example:

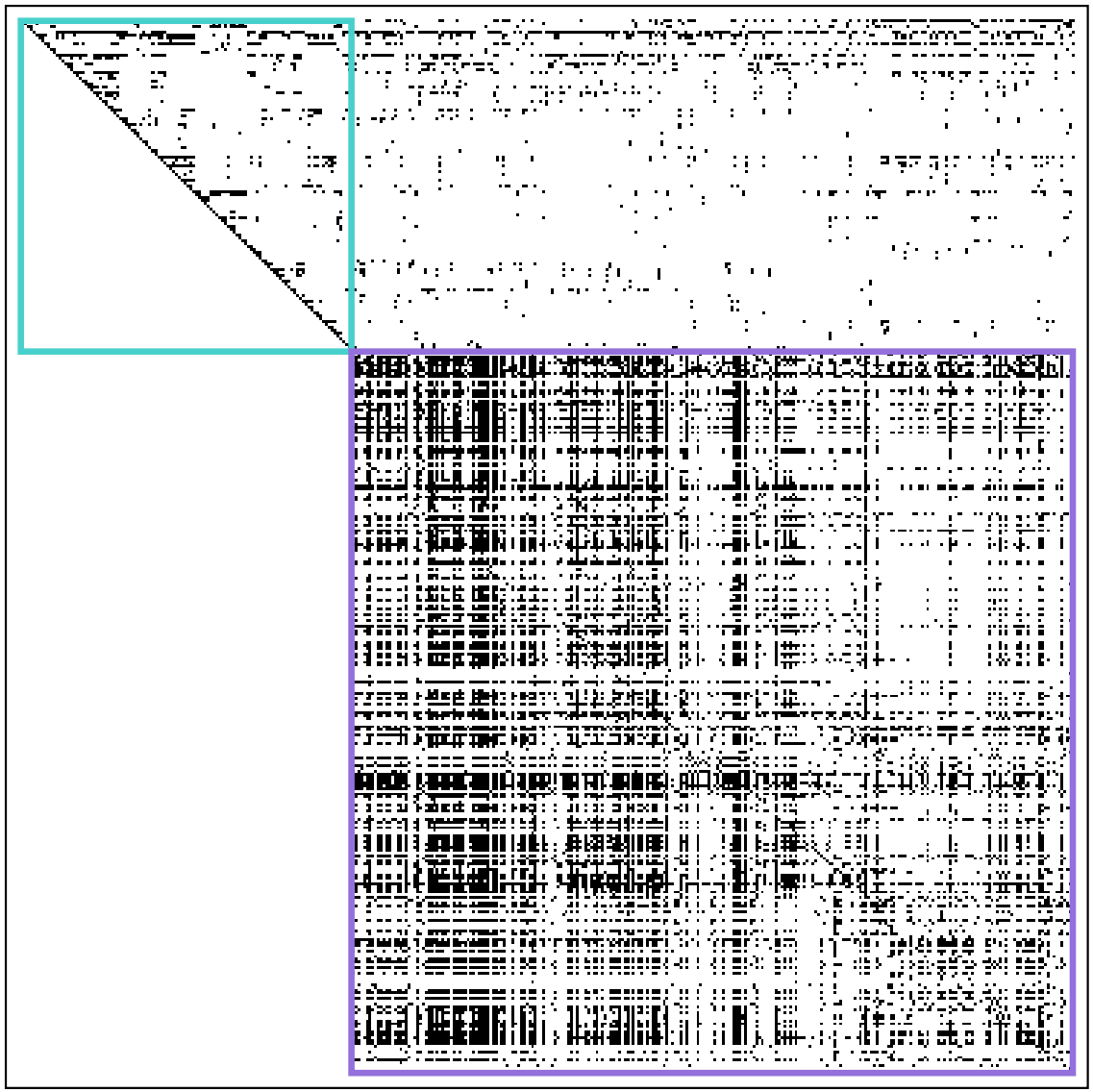


Example:



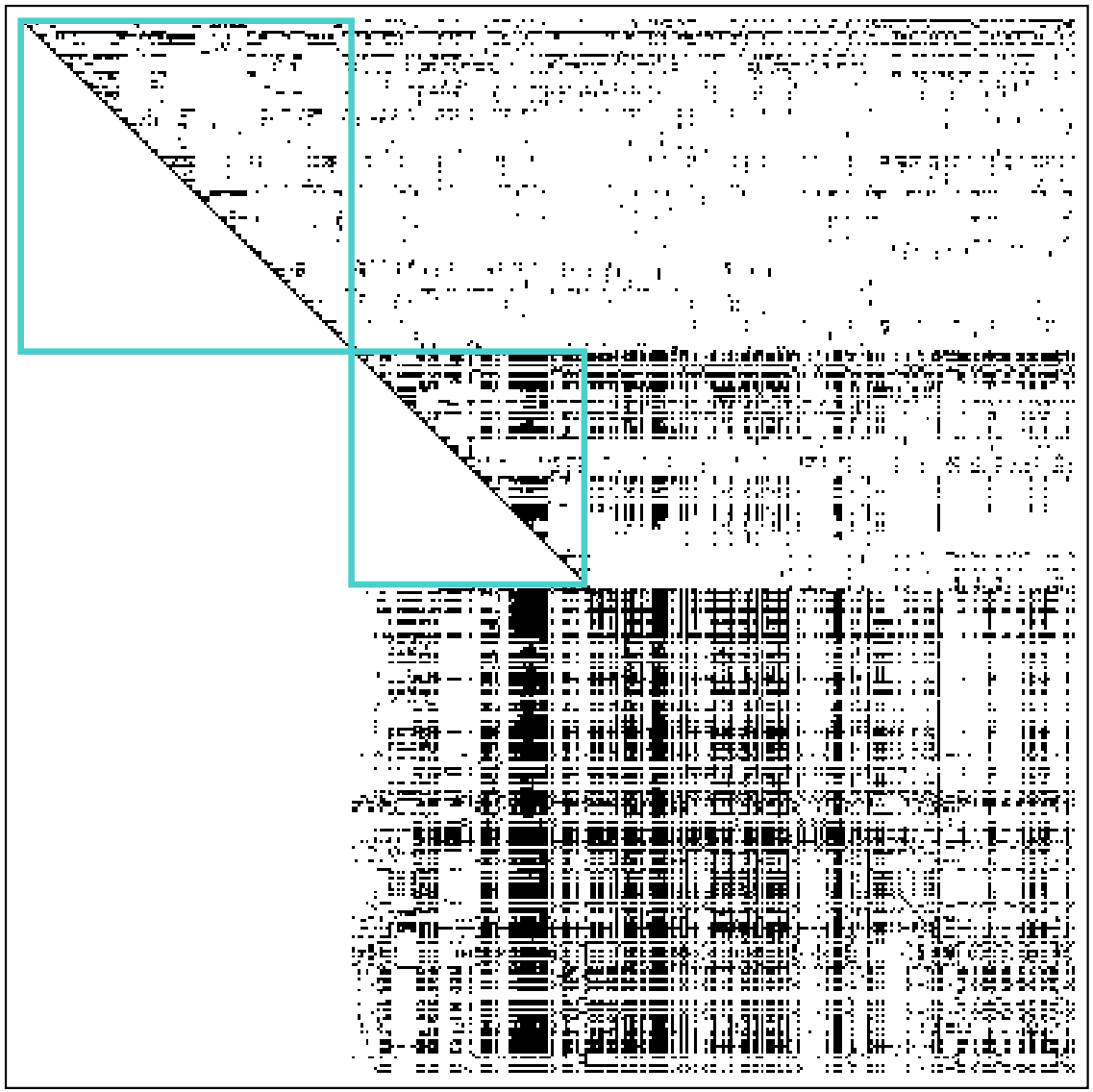
Set of structural pivots

Example:

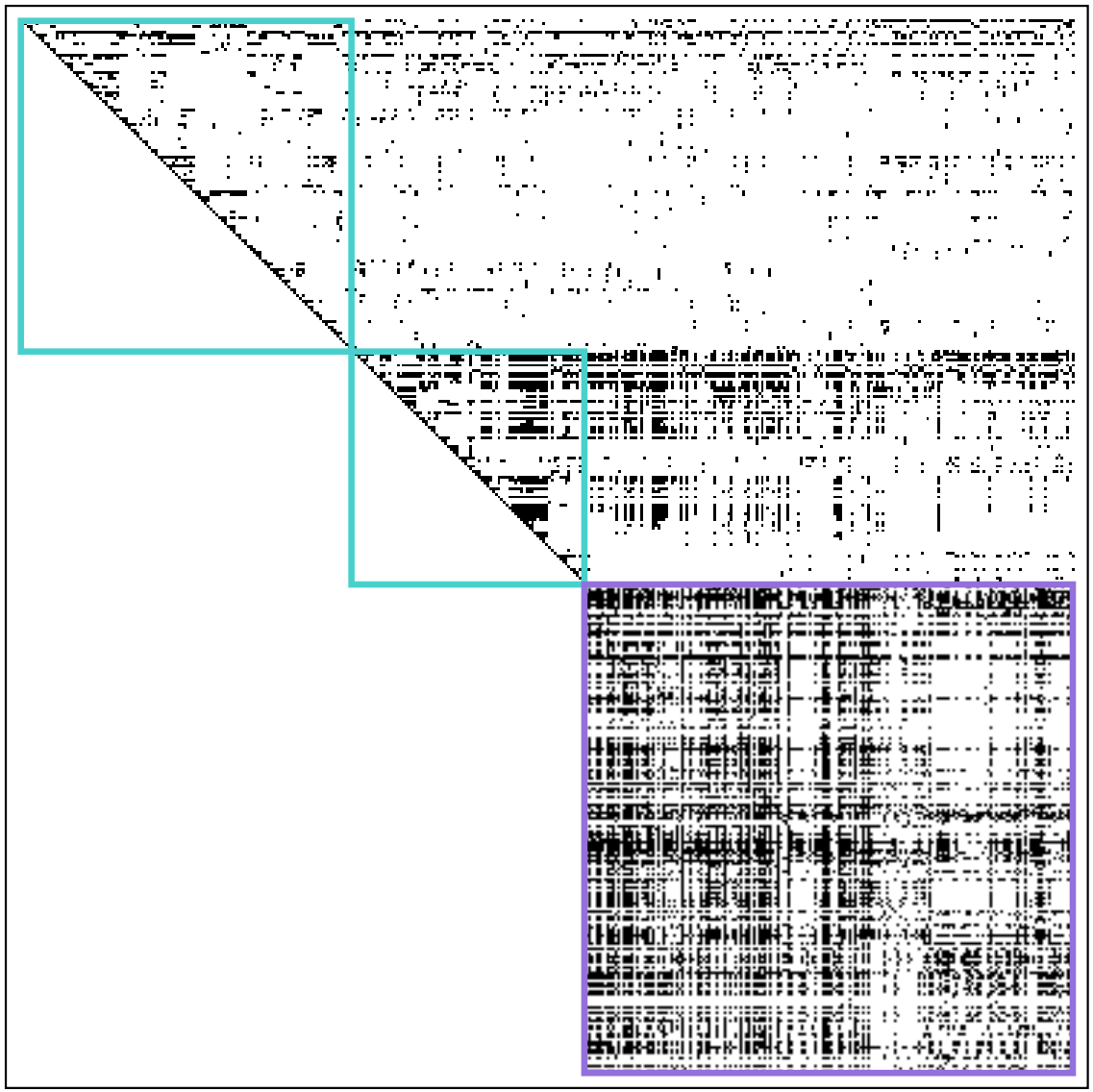


← Schur Complement

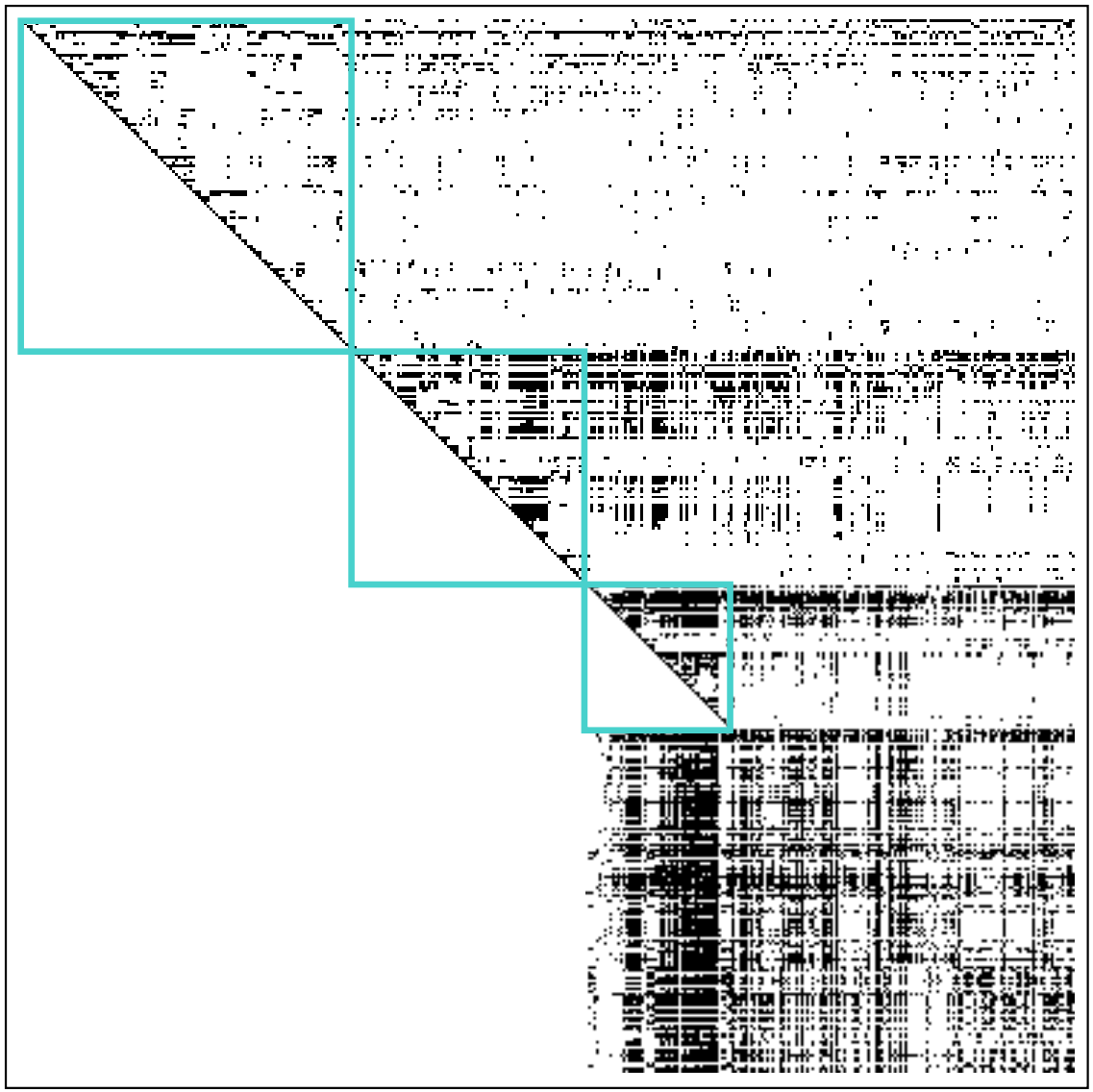
Example:



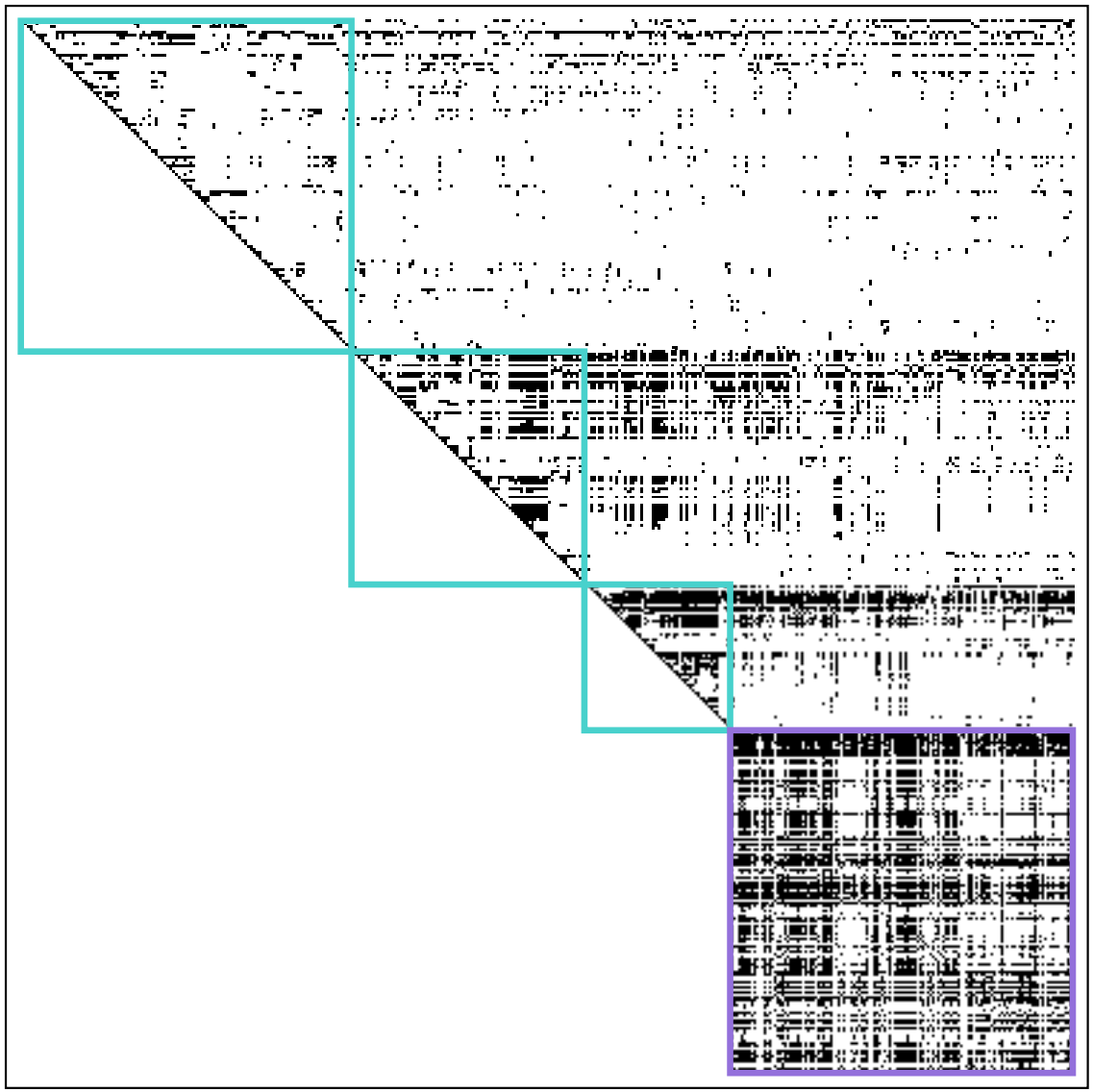
Example:



Example:



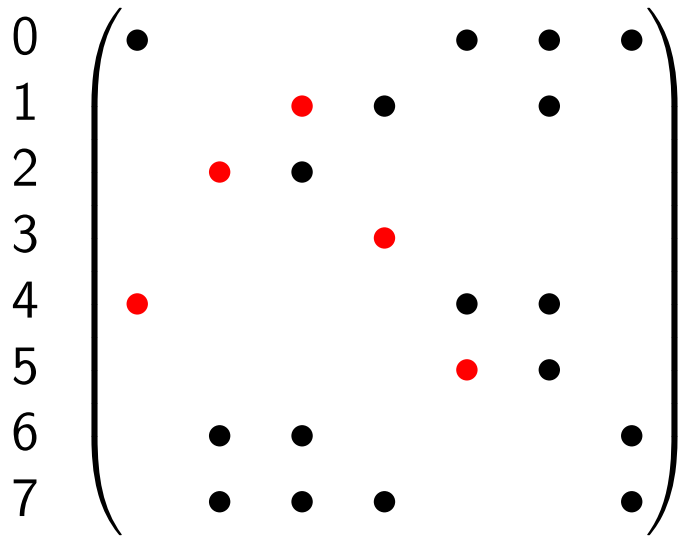
Example:

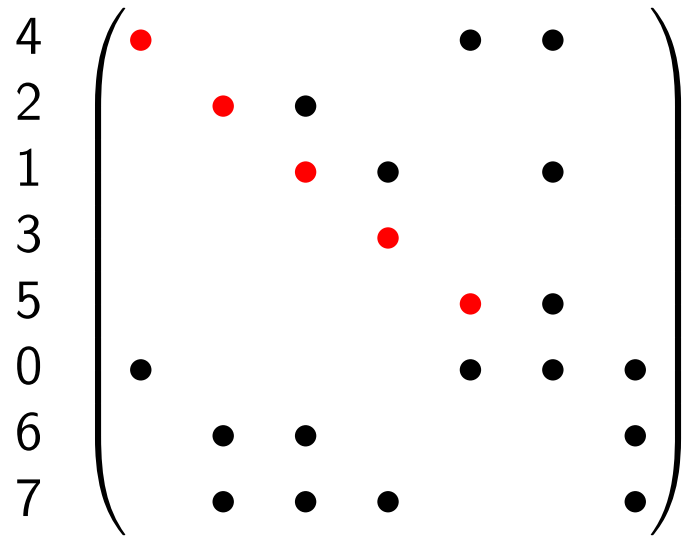


← Dense rank computation

Structural Pivot: Initial Idea

Faugère-Lachartre Heuristic [FL10]





Description

- Each row is mapped to the column of its leftmost coefficient.
- When several rows have the same leftmost coefficient, select the sparsest.
- Move the selected rows before the others and sort them by increasing position of the leftmost coefficient.

Find a Large Set of Structural Pivots

Why

- Enable us to reduce the number of the elimination steps
- Keep U sparse $\Rightarrow$ fast elimination steps

Find a Large Set of Structural Pivots

Why

- Enable us to reduce the number of the elimination steps
- Keep U sparse $\Rightarrow$ fast elimination steps

How

- Find the largest set: NP-Hard
- Find a large set using heuristics

Find a Large Set of Structural Pivots

Why

- Enable us to reduce the number of the elimination steps
- Keep U sparse $\Rightarrow$ fast elimination steps

How

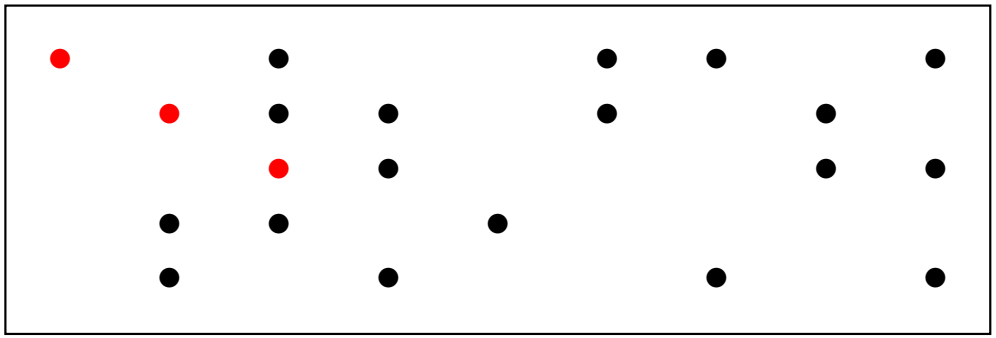
- Find the largest set: NP-Hard
- Find a large set using heuristics

Our Work

We designed two algorithms:

- A fast algorithm (*disappointing in most cases*)
- A parallelizable greedy algorithm

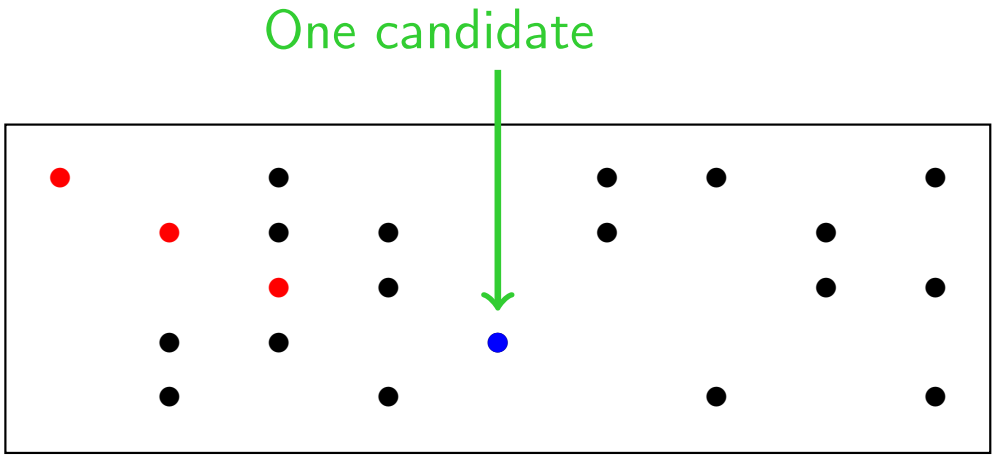
Greedy Algorithm



Choice of new pivots:

The pivotal submatrix must form a DAG $\Rightarrow$ If there is a cycle, the entry can't be selected.

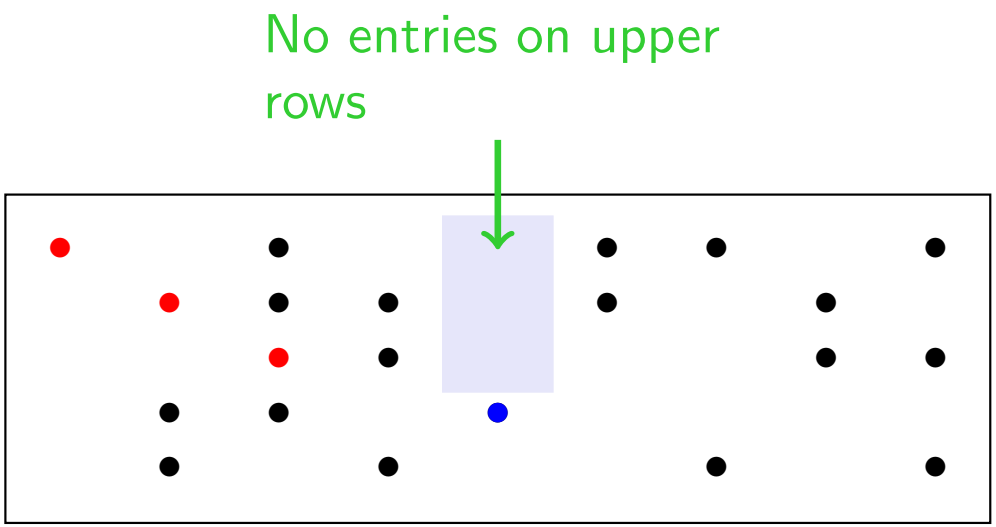
Greedy Algorithm



Choice of new pivots:

The pivotal submatrix must form a DAG $\Rightarrow$ If there is a cycle, the entry can't be selected.

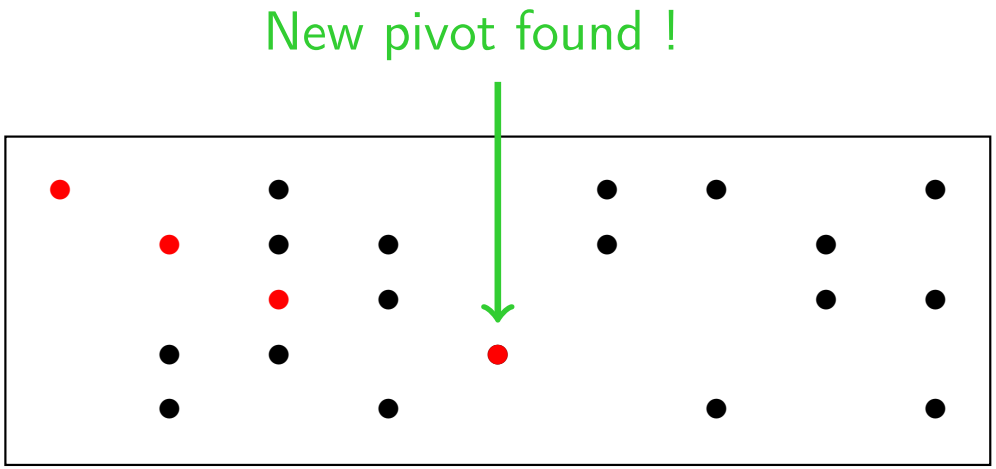
Greedy Algorithm



Choice of new pivots:

The pivotal submatrix must form a DAG $\Rightarrow$ If there is a cycle, the entry can't be selected.

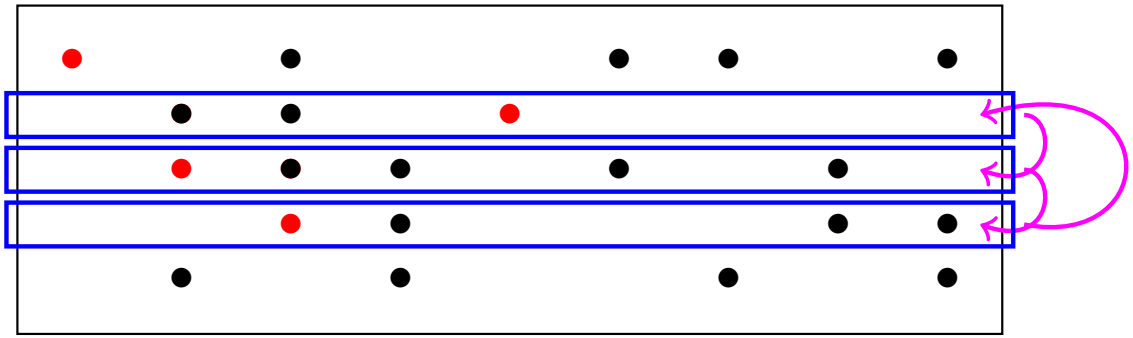
Greedy Algorithm



Choice of new pivots:

The pivotal submatrix must form a DAG $\Rightarrow$ If there is a cycle, the entry can't be selected.

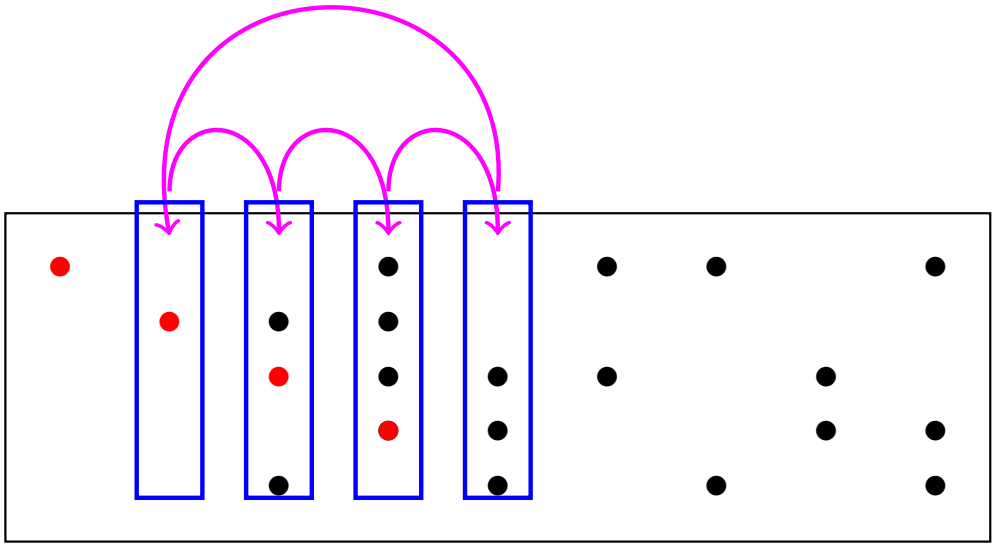
Greedy Algorithm



Choice of new pivots:

The pivotal submatrix must form a DAG $\Rightarrow$ If there is a cycle, the entry can't be selected.

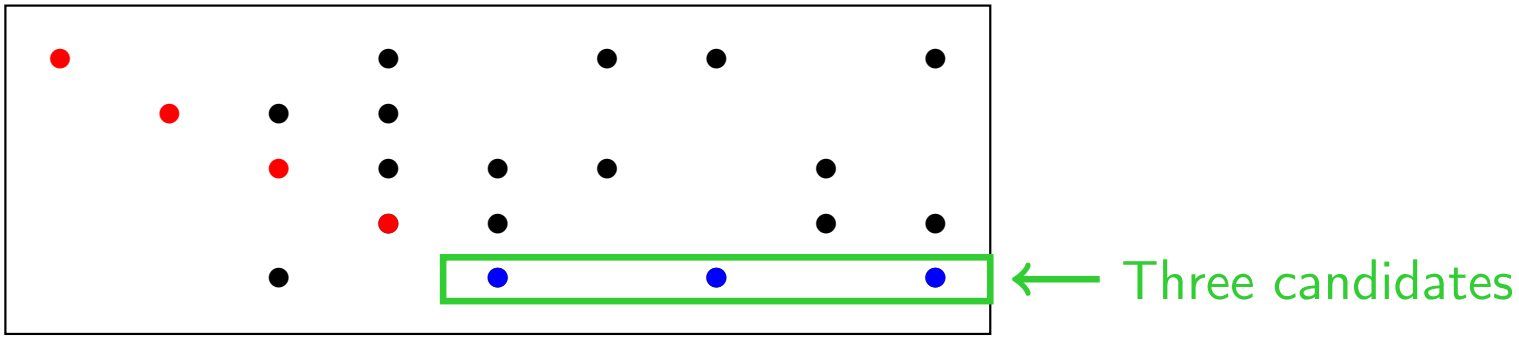
Greedy Algorithm



Choice of new pivots:

The pivotal submatrix must form a DAG $\Rightarrow$ If there is a cycle, the entry can't be selected.

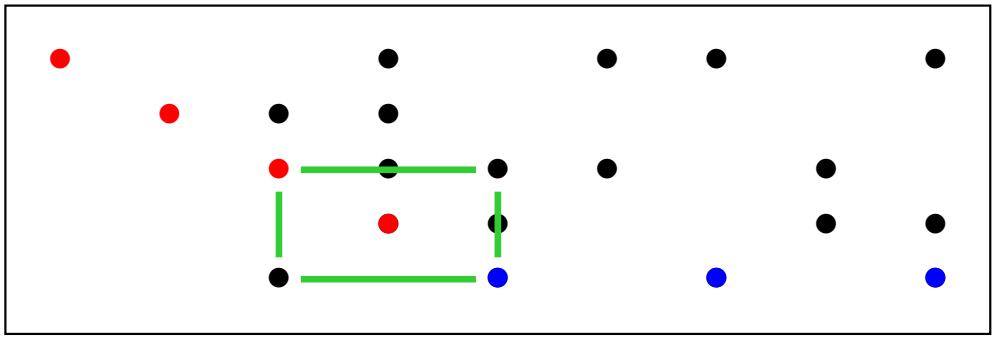
Greedy Algorithm



Choice of new pivots:

The pivotal submatrix must form a DAG $\Rightarrow$ If there is a cycle, the entry can't be selected.

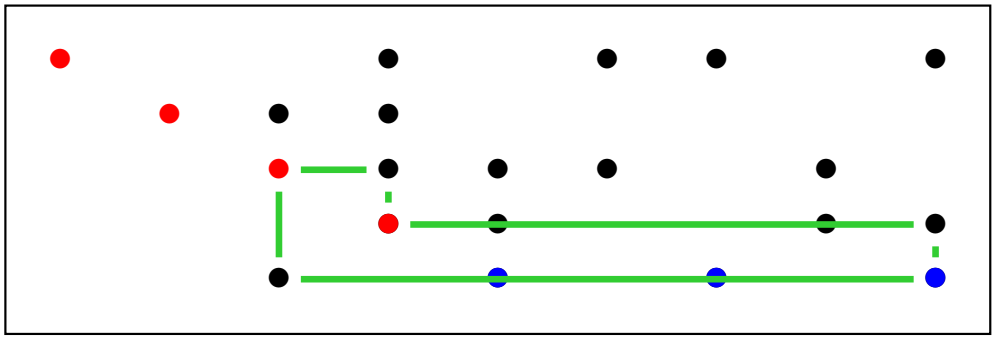
Greedy Algorithm



Choice of new pivots:

The pivotal submatrix must form a DAG $\Rightarrow$ If there is a cycle, the entry can't be selected.

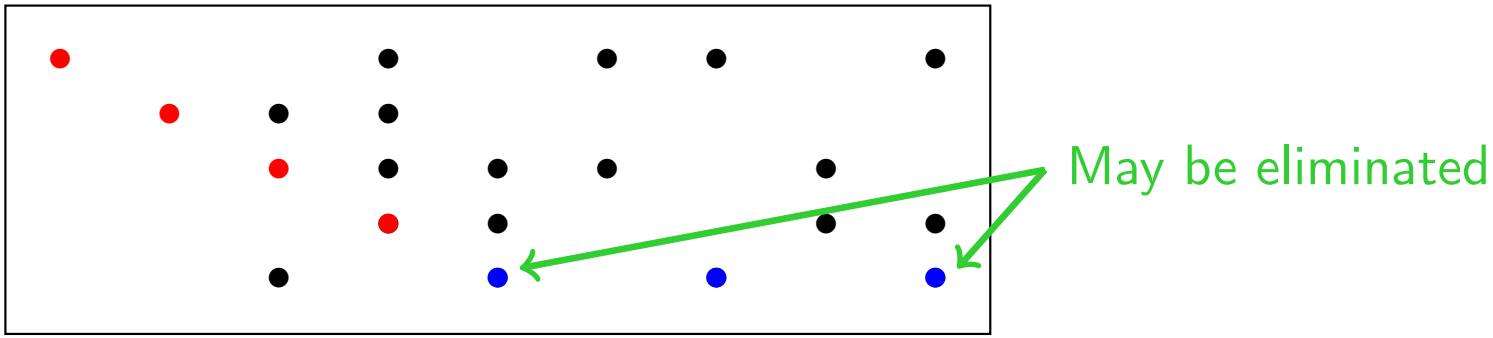
Greedy Algorithm



Choice of new pivots:

The pivotal submatrix must form a DAG $\Rightarrow$ If there is a cycle, the entry can't be selected.

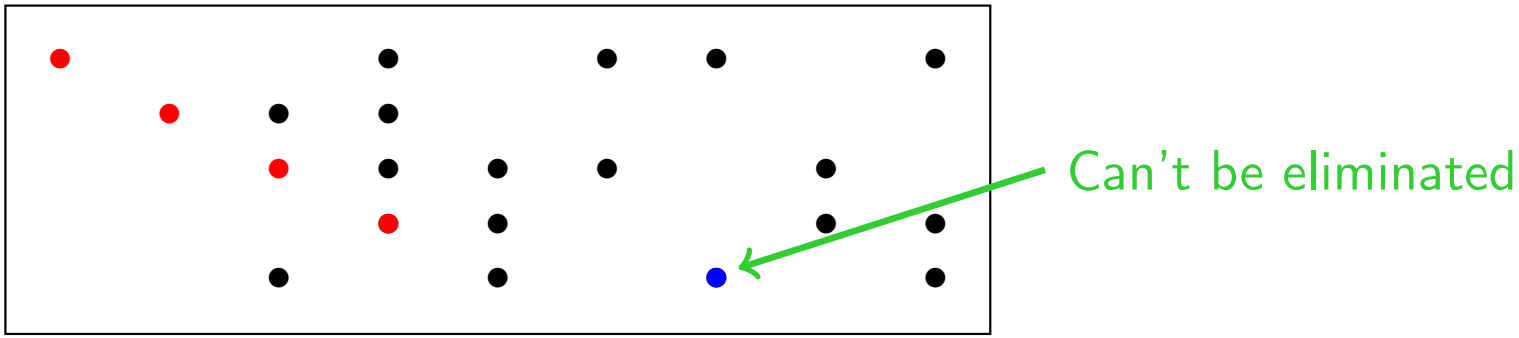
Greedy Algorithm



Choice of new pivots:

The pivotal submatrix must form a DAG $\Rightarrow$ If there is a cycle, the entry can't be selected.

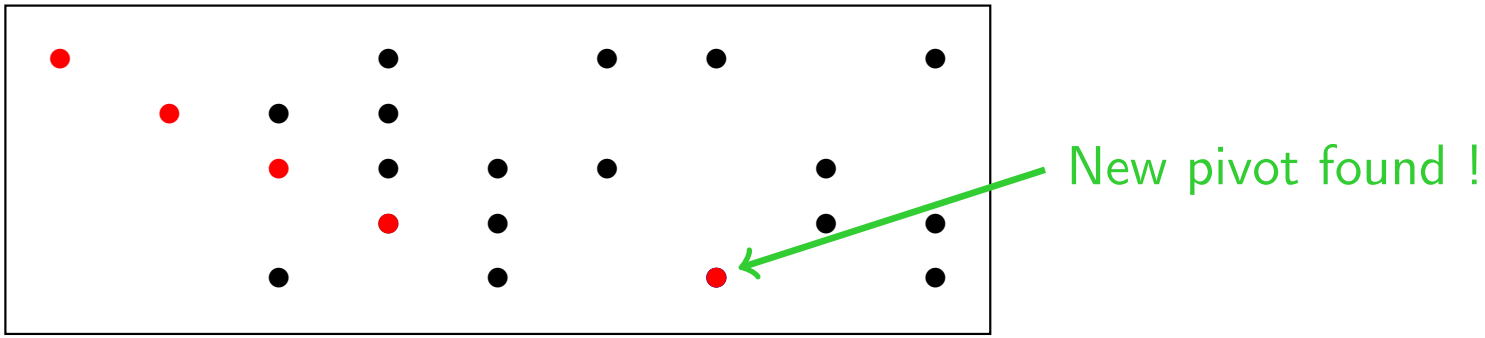
Greedy Algorithm



Choice of new pivots:

The pivotal submatrix must form a DAG $\Rightarrow$ If there is a cycle, the entry can't be selected.

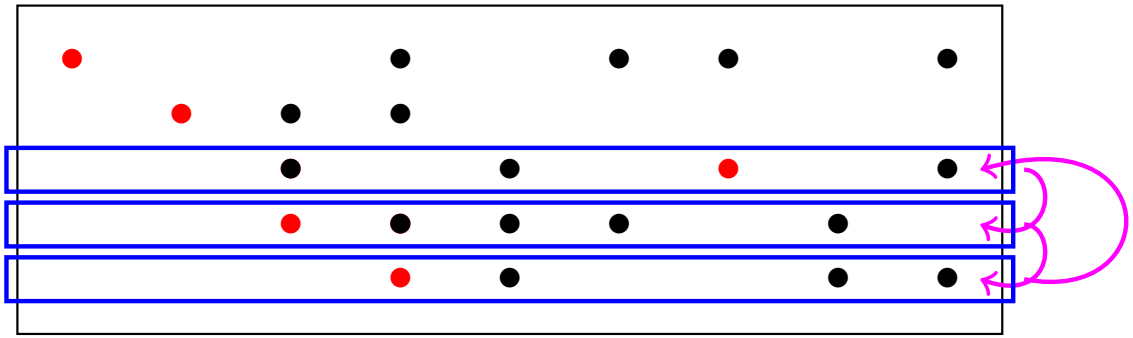
Greedy Algorithm



Choice of new pivots:

The pivotal submatrix must form a DAG $\Rightarrow$ If there is a cycle, the entry can't be selected.

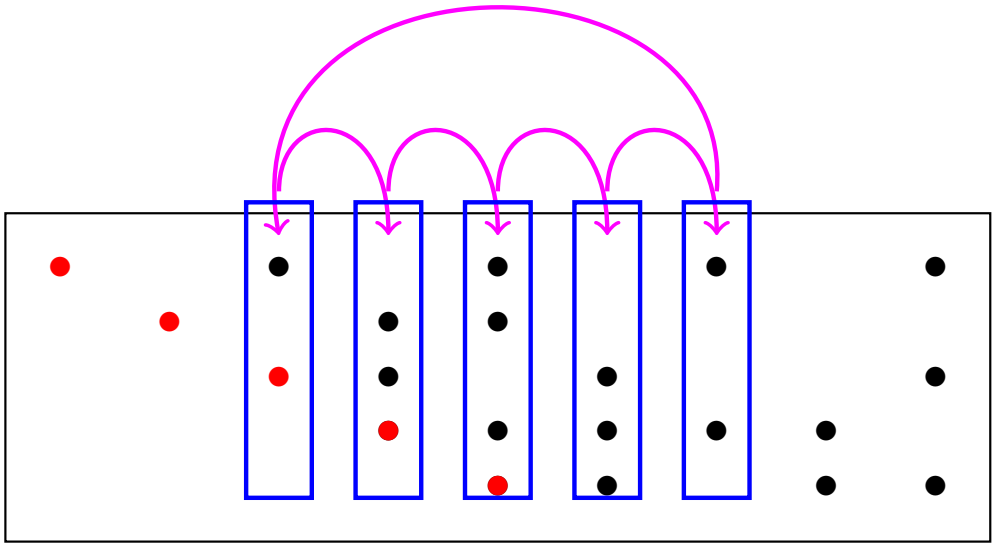
Greedy Algorithm



Choice of new pivots:

The pivotal submatrix must form a DAG $\Rightarrow$ If there is a cycle, the entry can't be selected.

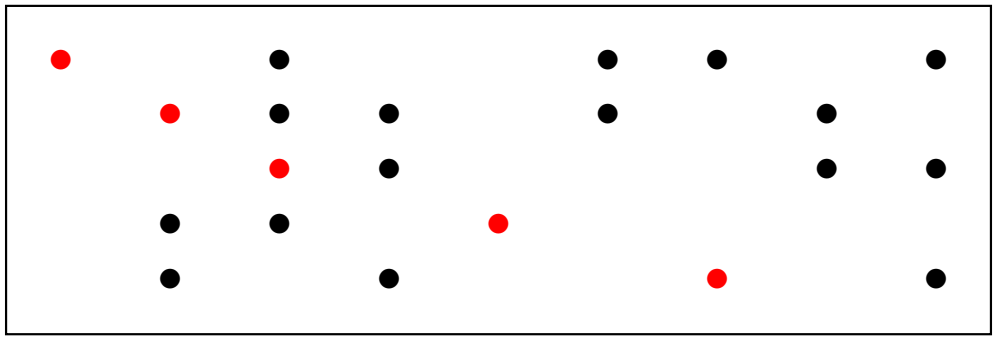
Greedy Algorithm



Choice of new pivots:

The pivotal submatrix must form a DAG $\Rightarrow$ If there is a cycle, the entry can't be selected.

Our new pivot selection algorithm



In a Nutshell:

- Find the F.-L. pivots
- Push the F.-L. pivots on the top of A.
- For all non-pivotal column j, if the upmost coefficient a_{ij} is on a non-pivotal row, select a_{ij}
- Perform a graph search to find more pivots

Graph Search

Idea

For each non pivotal row i:

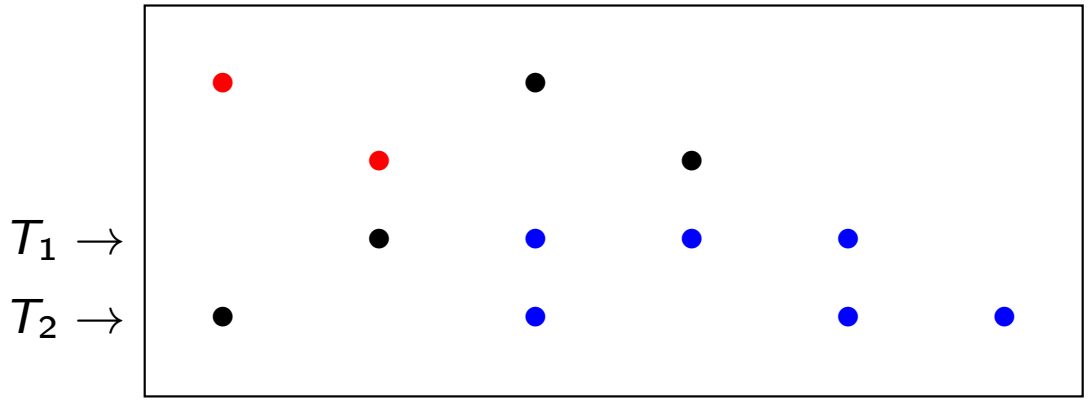
- Start a BFS from each pivotal column j such that $a_{ij} \neq 0$
- When a column k such that $a_{ik} \neq 0$ is reached, remove a_{ik} from the list of candidates.
- If there are still candidates after the BFS end, we select the leftmost one.

Remark

- This method is parallelizable.
- The worst case complexity of the search is $\mathcal{O}(n|A|)$ (same as Wiedemann)
- In practice : much faster than Wiedemann

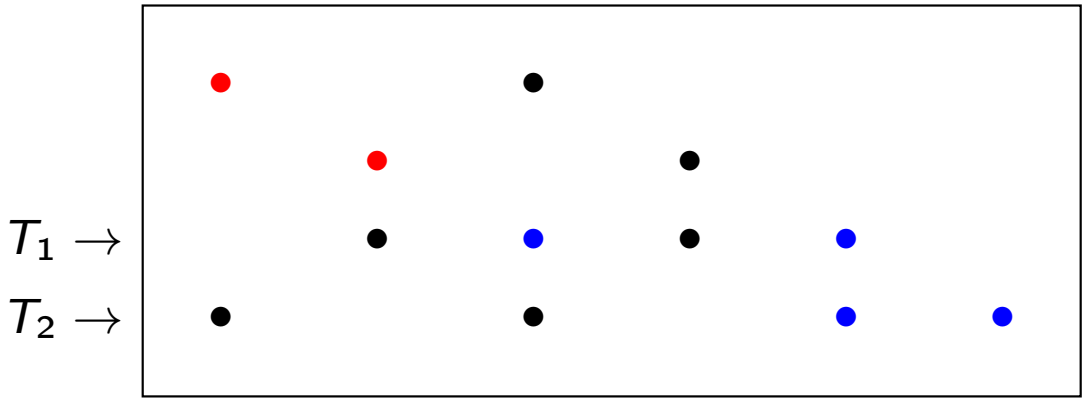
Parallelization Strategy

Idea: Process k rows simultaneously on k threads



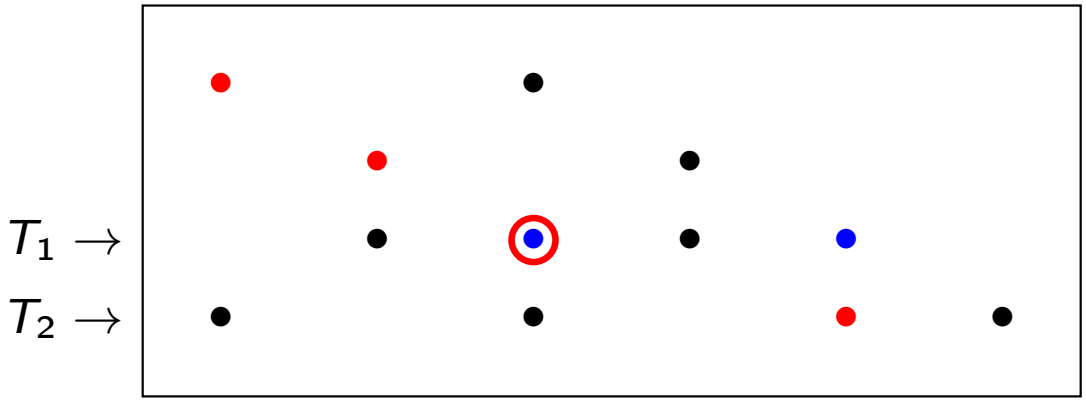
Parallelization Strategy

Idea: Process k rows simultaneously on k threads



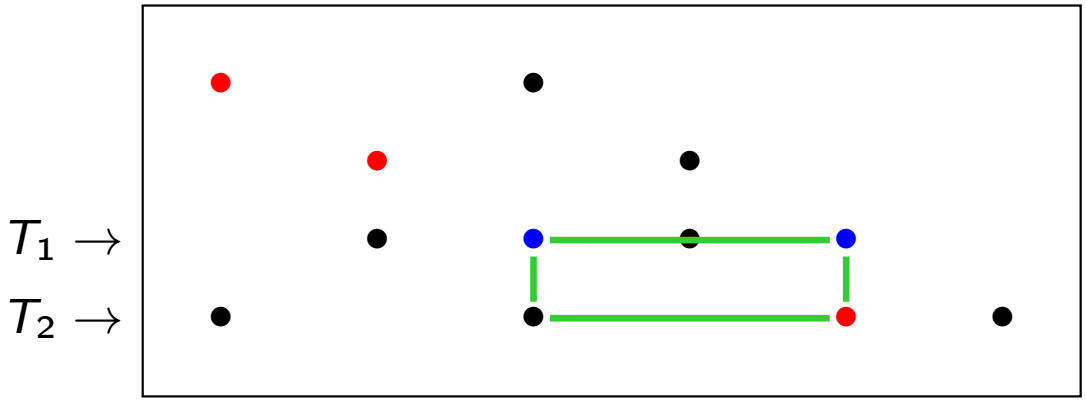
Parallelization Strategy

Idea: Process k rows simultaneously on k threads



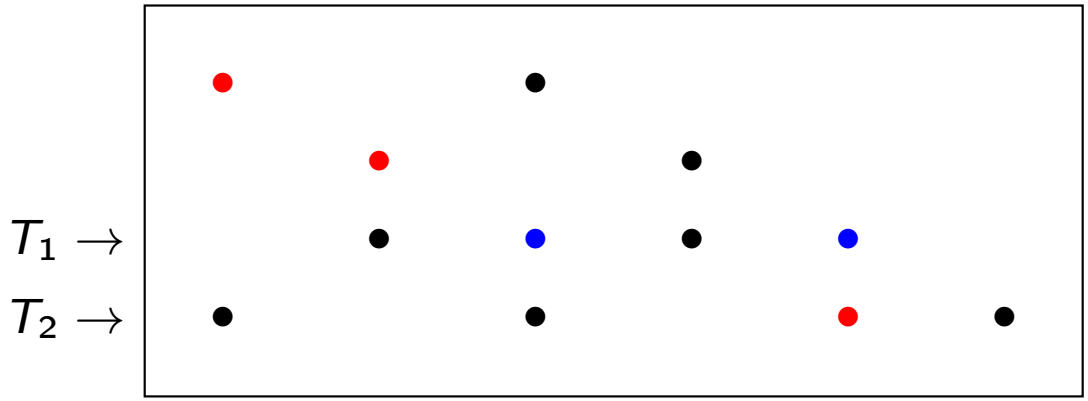
Parallelization Strategy

Idea: Process k rows simultaneously on k threads



Parallelization Strategy

Idea: Process k rows simultaneously on k threads



Observation

Issue: New cycles may appear in the meantime
$\Rightarrow$ If new pivots have been found by other threads restart the search

Benchmark

Our results

- Benchmark performed on large matrices from SIMC (*e.g.* GL7d folder)
- Find up to $99,9\%$ of the total number of pivots using our greedy algorithm
- Much faster than the Wiedemann Algorithm in some cases

Benchmark

Our results

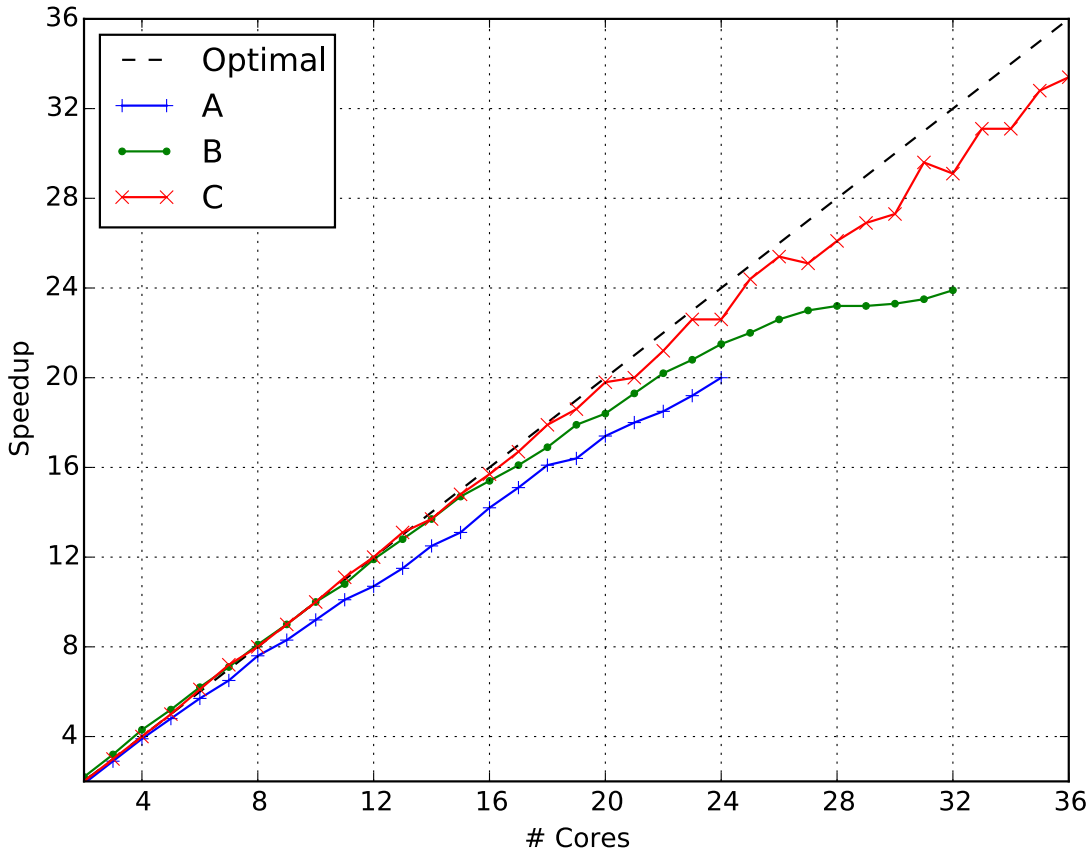
- Benchmark performed on large matrices from SIMC (*e.g.* GL7d folder)
- Find up to $99,9\%$ of the total number of pivots using our greedy algorithm
- Much faster than the Wiedemann Algorithm in some cases

Case study

- Easy case (GL7d15)
  - 3 hours using sequential Wiedemann (LinBox)
  - **10 sec on 36 cores using our Algorithm**
- Hardest case (GL7d19)
  - More than 16 days using sequential Wiedemann (LinBox)
  - **4 min on 36 cores using our Algorithm**

Parallel Efficiency

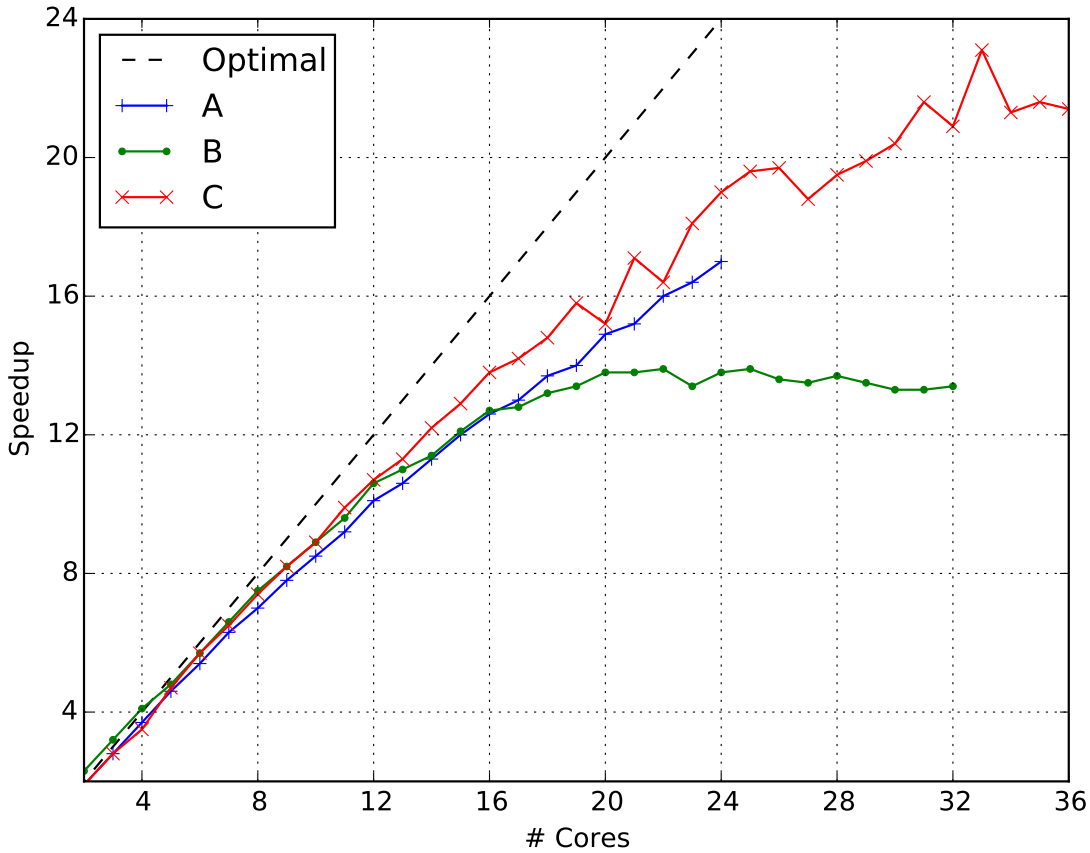
Matrix GL7d16



Machine	# CPU	# cores	L3 Cache
A	2× Xeon E5-2670 v3	12	30MB
B	4× Xeon E5-4620	8	16MB
C	2× Xeon E5-2695 v4	18	45 MB

Parallel Efficiency

Matrix GL7d19



Machine	# CPU	# cores	L3 Cache
A	2× Xeon E5-2670 v3	12	30MB
B	4× Xeon E5-4620	8	16MB
C	2× Xeon E5-2695 v4	18	45 MB

Conclusion

- We propose new pivots selections heuristic for Sparse LU Factorisation
- Implemented in C using OpenMP in the SpaSM (SPArse direct Solver Modulo p) library and publicly available at:

`https://github.com/cbouilla/spasm`

Open questions:

- Can we still improve the pivots selection?

Conclusion

- We propose new pivots selections heuristic for Sparse LU Factorisation
- Implemented in C using OpenMP in the SpaSM (SPArse direct Solver Modulo p) library and publicly available at:

`https://github.com/cbouilla/spasm`

Open questions:

- Can we still improve the pivots selection?

Thank you for your time !