Parallel sparse linear algebra (2)

Prof. Richard Vuduc
Georgia Institute of Technology

CSE/CS 8803 PNA, Spring 2008

[L.10] Thursday, February 7, 2008

Sources for today’s material

- Mike Heath (UIUC)
- CS 267 (Yelick & Demmel, UCB)
- John Gilbert (UCSB)
- Xiaoye (Sherry) Li (LBNL)
- *Direct methods for sparse matrices*, by Duff, Erisman, and Reid
- Esmond Ng (LBNL)

Algorithms for 2-D (3-D) Poisson, $N=n^2$ ($=n^3$)

Algorithm	Serial	PRAM	Memory	# procs
Dense LU	N^3	N	N^2	N^2
Band LU	N^2 ($N^{7/3}$)	N	$N^{3/2}$ ($N^{5/3}$)	N ($N^{4/3}$)
Jacobi	N^2 ($N^{5/3}$)	N ($N^{2/3}$)	N	N
Explicit inverse	N^2	$\log N$	N^2	N^2
Conj. grad.	$N^{3/2}$ ($N^{4/3}$)	$N^{1/2(1/3)} \log N$	N	N
RB SOR	$N^{3/2}$ ($N^{4/3}$)	$N^{1/2}$ ($N^{1/3}$)	N	N
Sparse LU	$N^{3/2}$ (N^2)	$N^{1/2}$	$N \log N$ ($N^{4/3}$)	N
FFT	$N \log N$	$\log N$	N	N
Multigrid	N	$\log^2 N$	N	N
Lower bound	N	$\log N$	N	

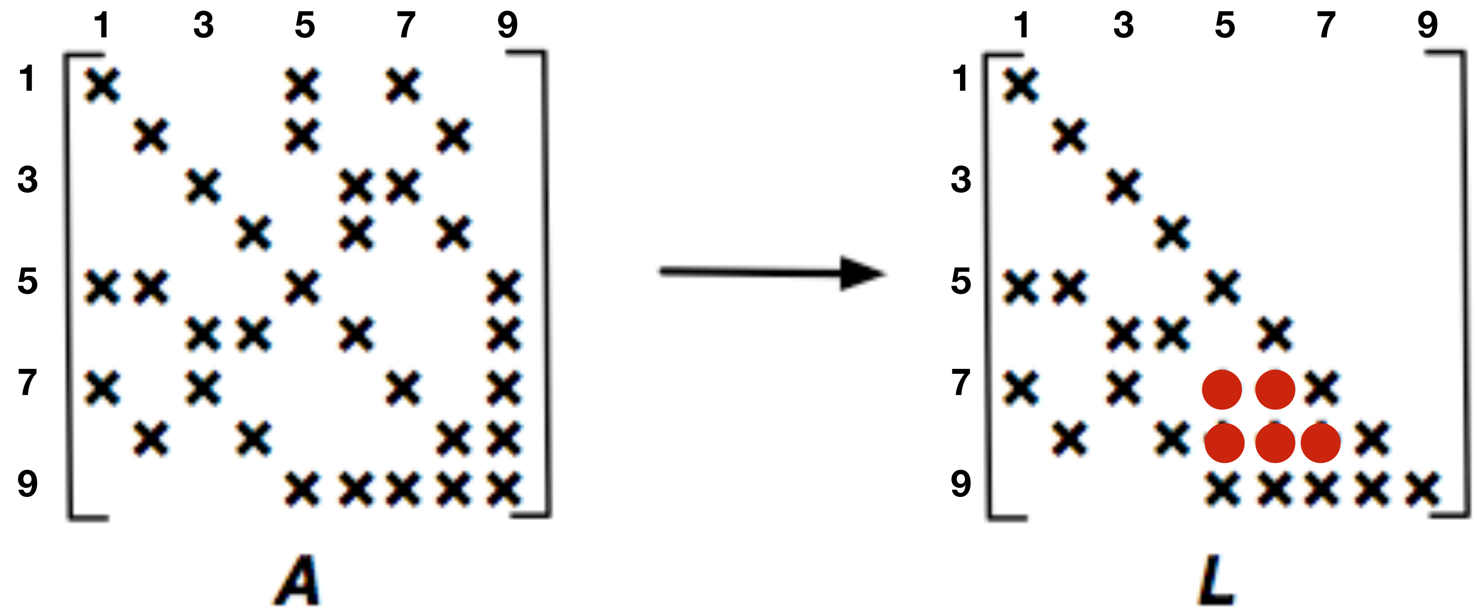
PRAM = idealized parallel model with zero communication cost.
Source: Demmel (1997)

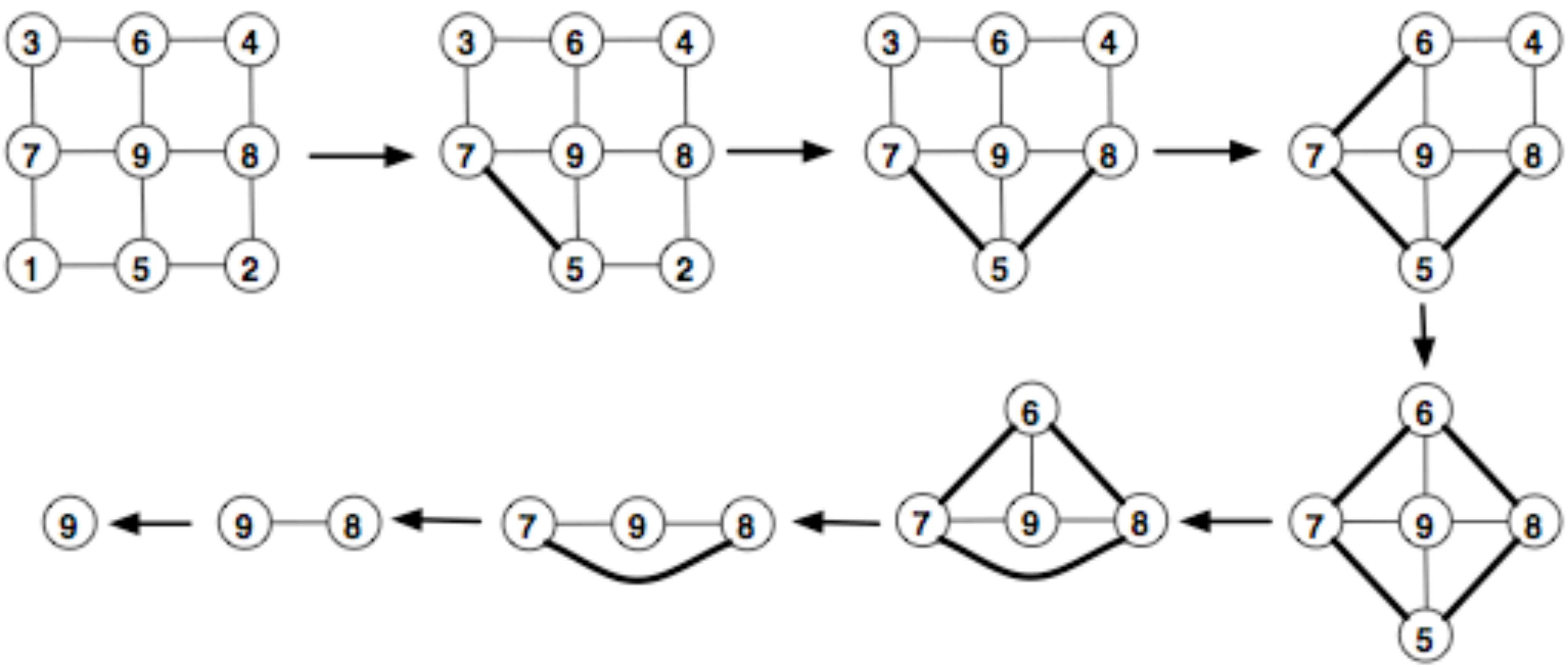
Anatomy of a sparse direct solver

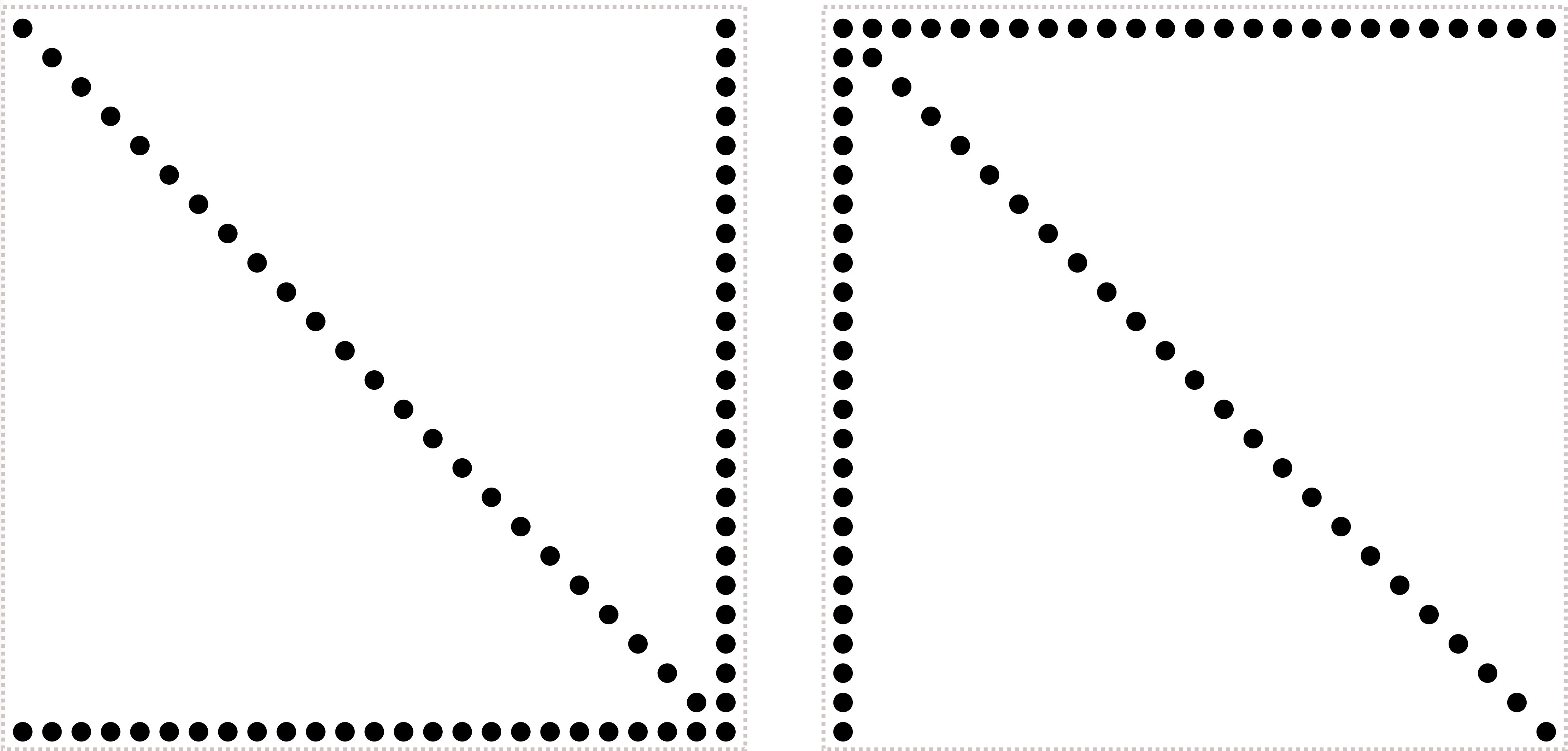
$$P_r \cdot A \cdot P_c^T = LU$$

- *Order equations & variables to minimize fill in L, U* [Combinatorial]
- Symbolic factorization [Allocate memory for L, U]
- Numerical factorization [Dominates run-time]
- Triangular solves [Small fraction, unless many RHS]

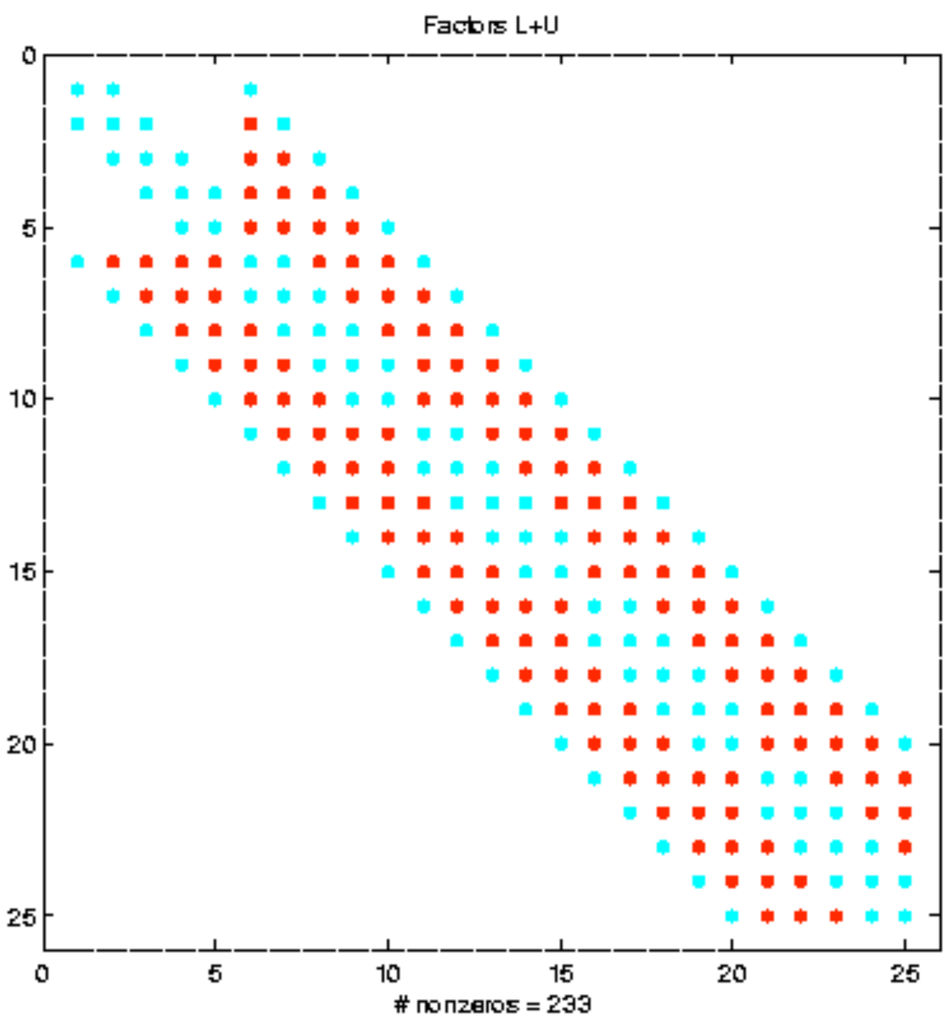
Review: Fill-reducing orderings


A
L

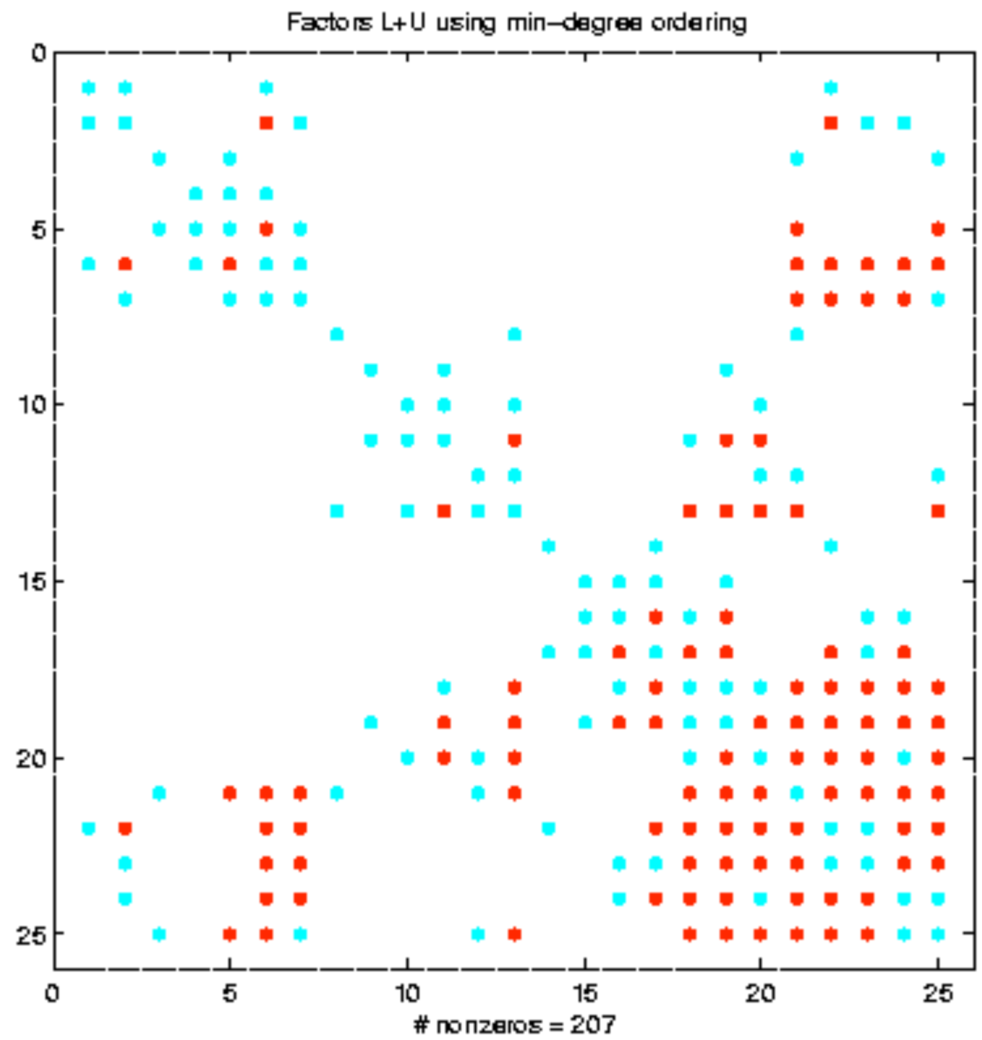




Total nnz = 233



Total nnz = 207



Ordering: Markowitz criterion ('57)

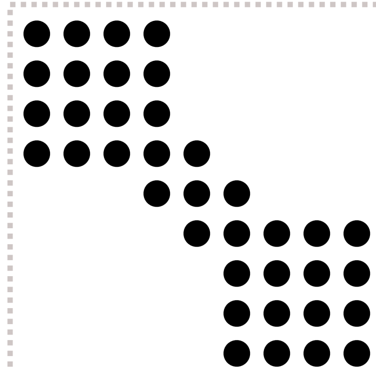
- At elimination stage k:
  - Let $r_i^{(k)}$ = nnz(row i), $c_j^{(k)}$ = nnz(col j), in uneliminated submatrix
  - Choose pivot entry a_{ij} (*i.e.*, swap row & col) that minimizes:

$$(r_i^{(k)} - 1) \times (c_j^{(k)} - 1)$$

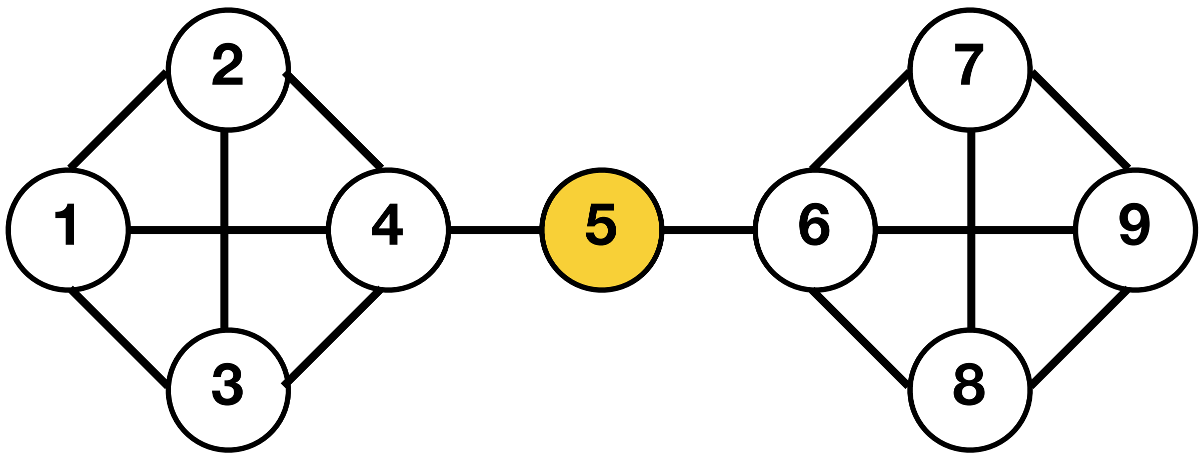
- Don't forget about numerical values, too! (*E.g.*, threshold)
- **Problem: Expensive!**

Minimum degree ordering

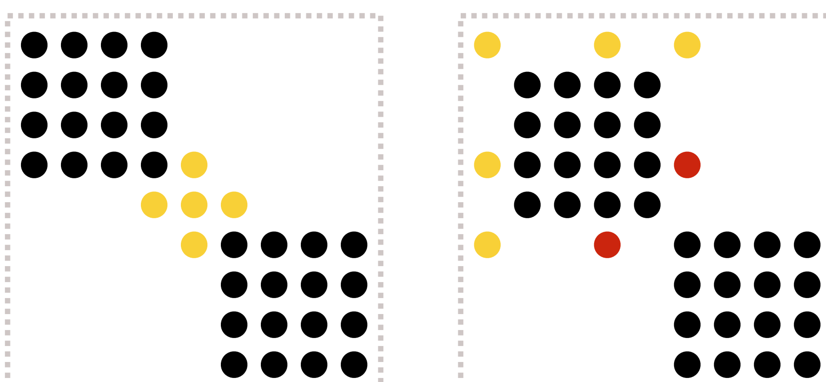
No fill in natural ordering:



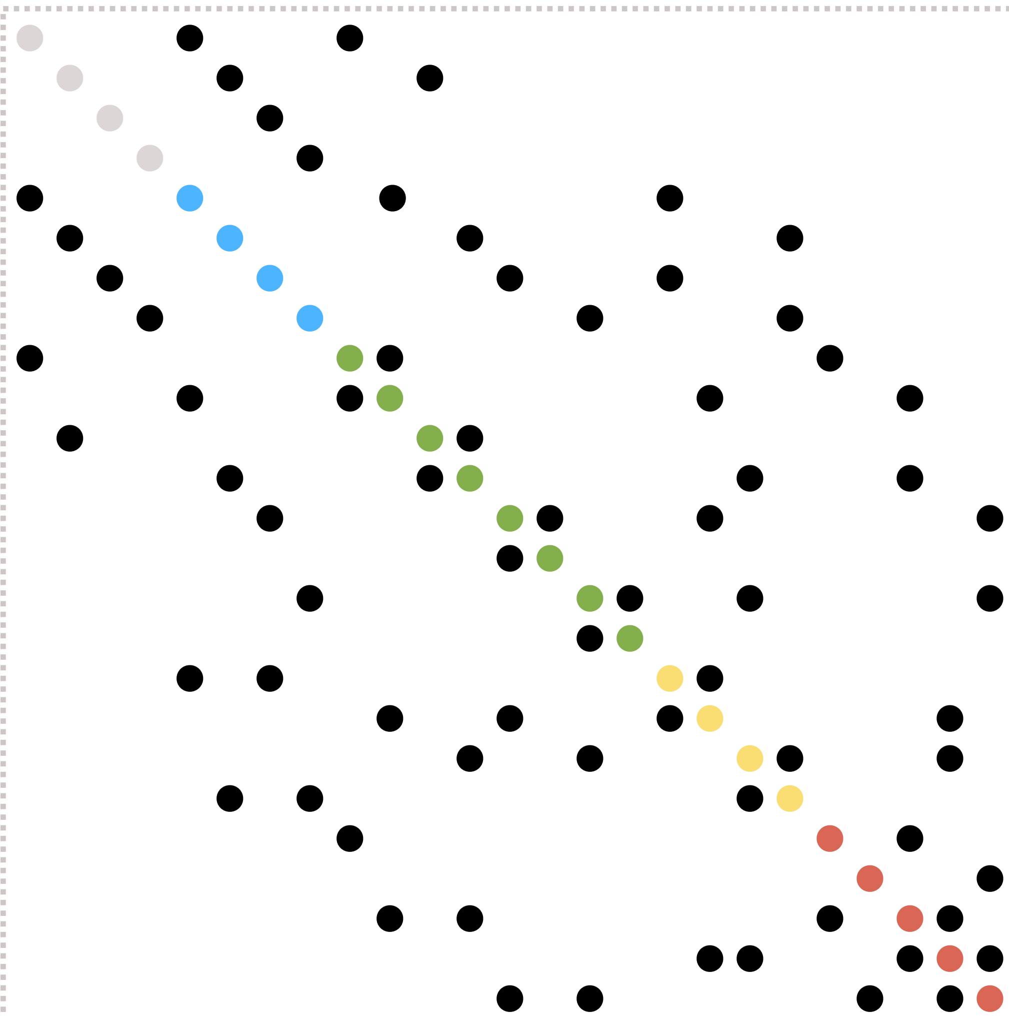
Min-degree node: 5

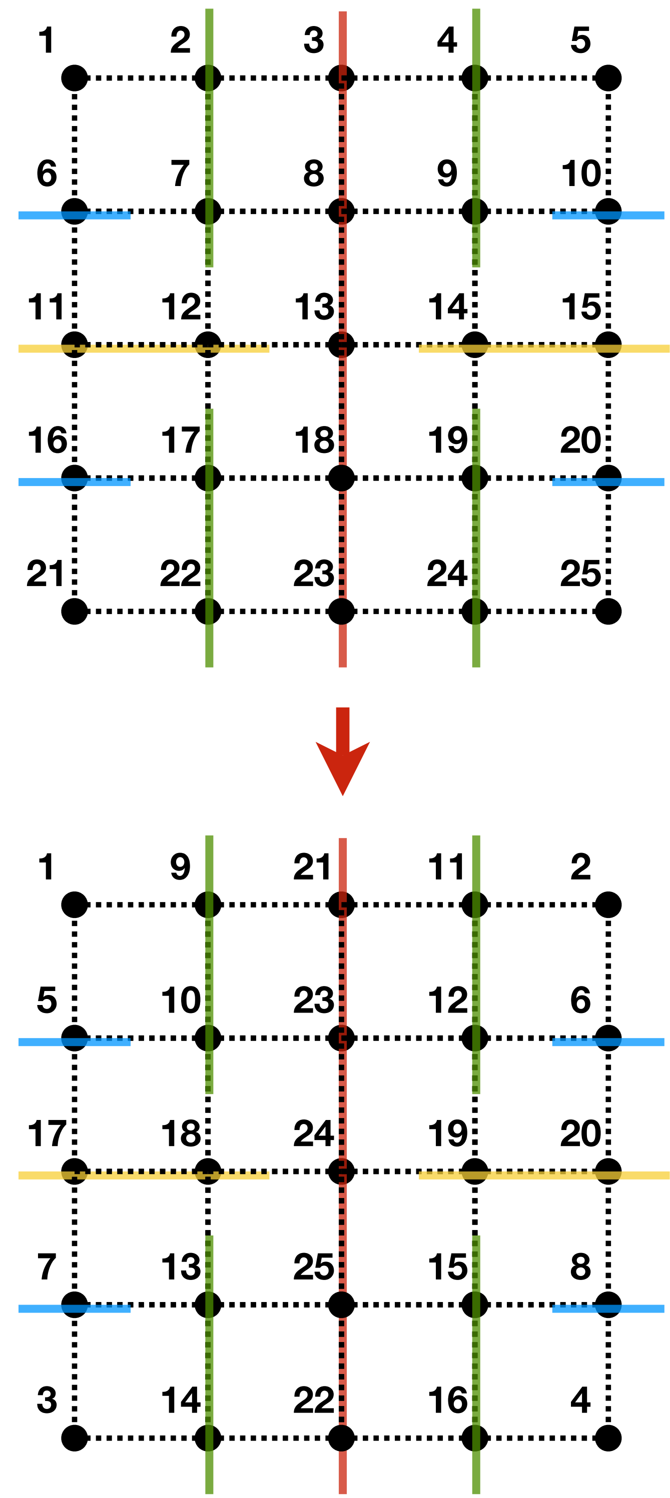


Choosing 5 leads to fill-in:



Nested dissection

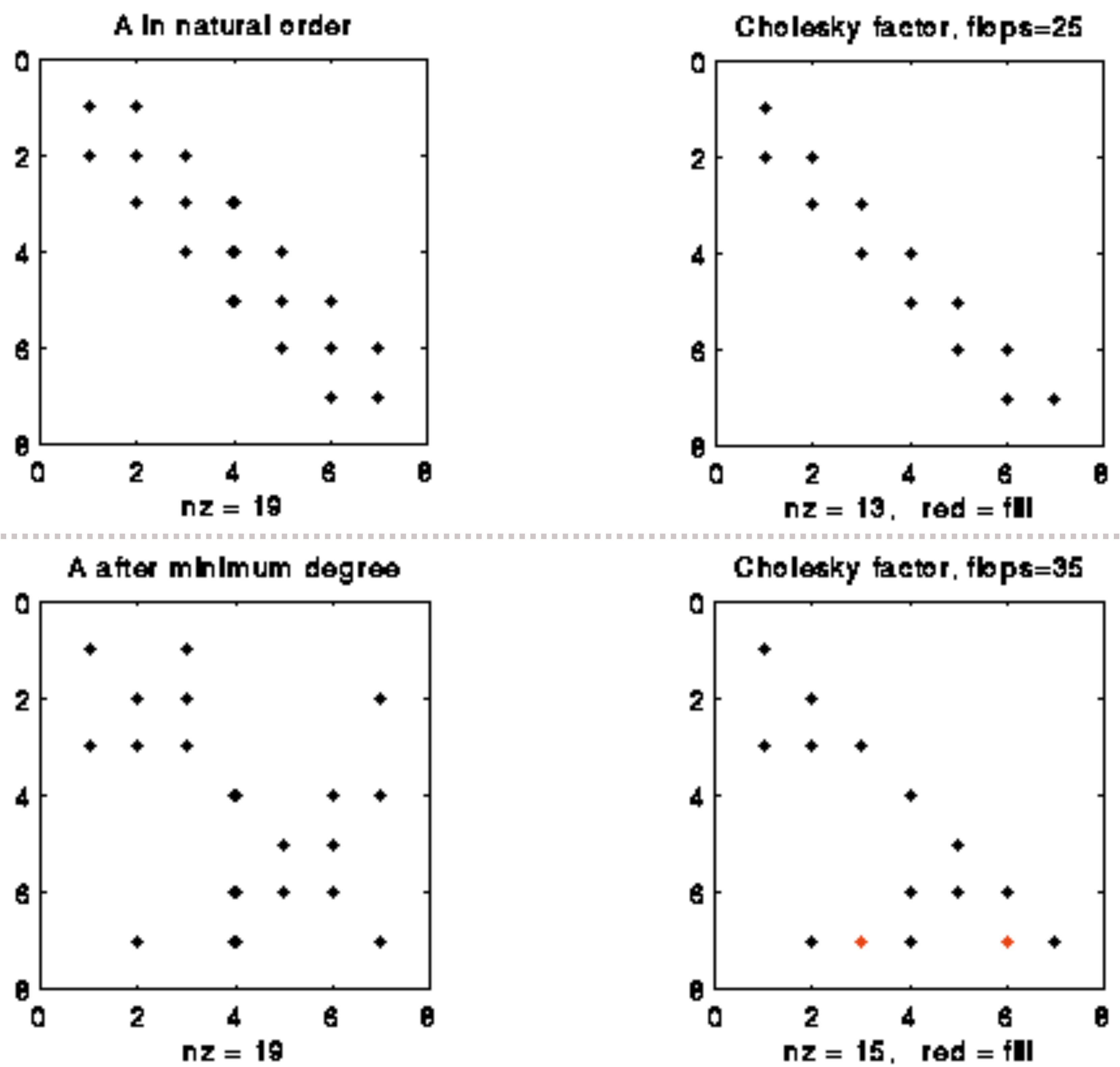


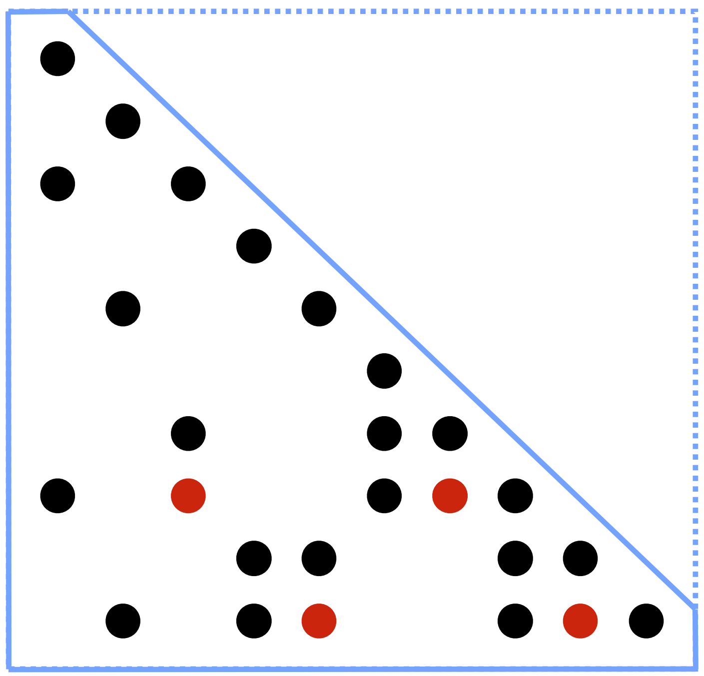


Finding good separators

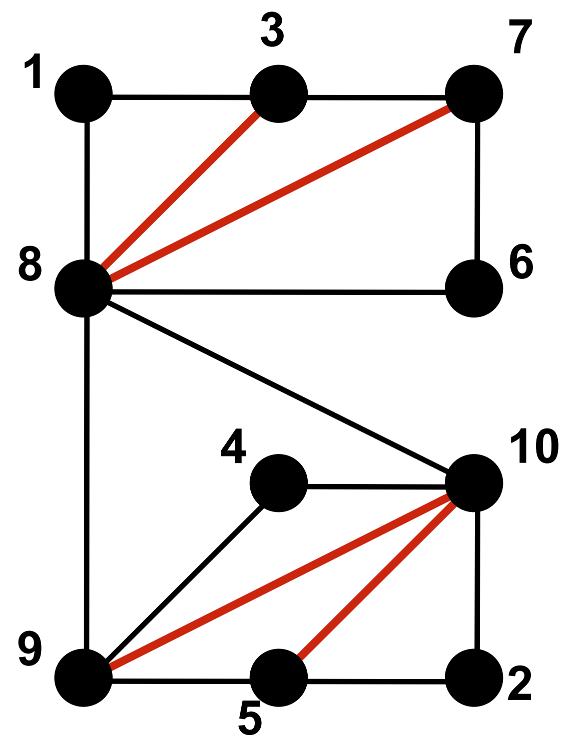
- Multilevel schemes
  - Chaco [Hendrickson & Leland ’94]
  - Metis [Karypis & Kumar ’95]
- Spectral bisection [Simon, *et al*. ’90+]
- Geometric and spectral bisection [Chan, Gilbert, & Teng ’94]

Sources of parallelism: Elimination trees

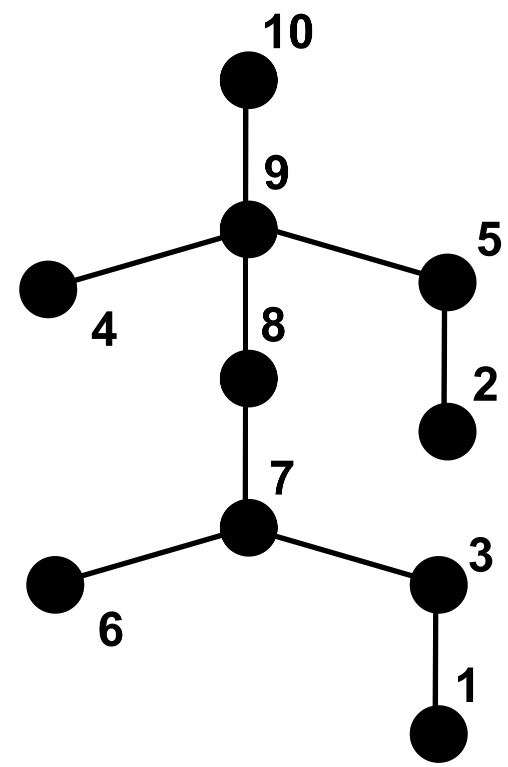

A in natural order
nz = 19
Cholesky factor, flops=25
nz = 13, red = fill
A after minimum degree
nz = 19
Cholesky factor, flops=35
nz = 15, red = fill



Cholesky factor



G⁺(A)

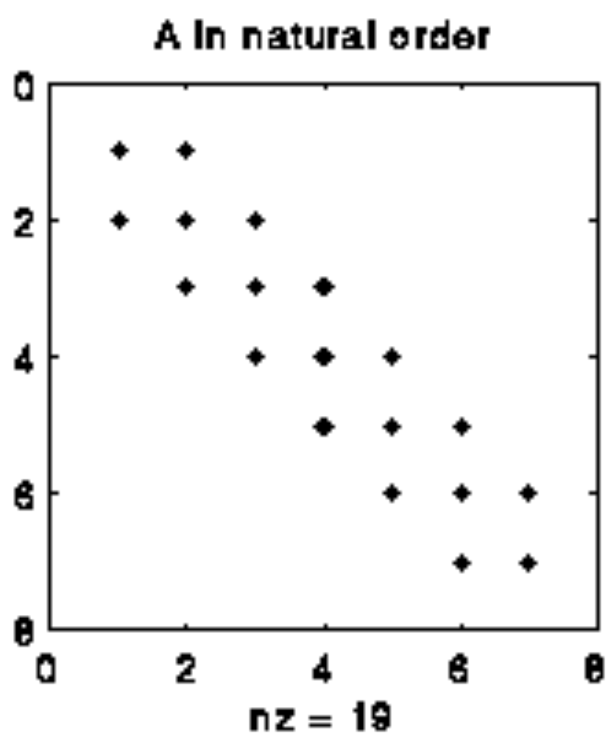


T(A)
"Elimination tree"

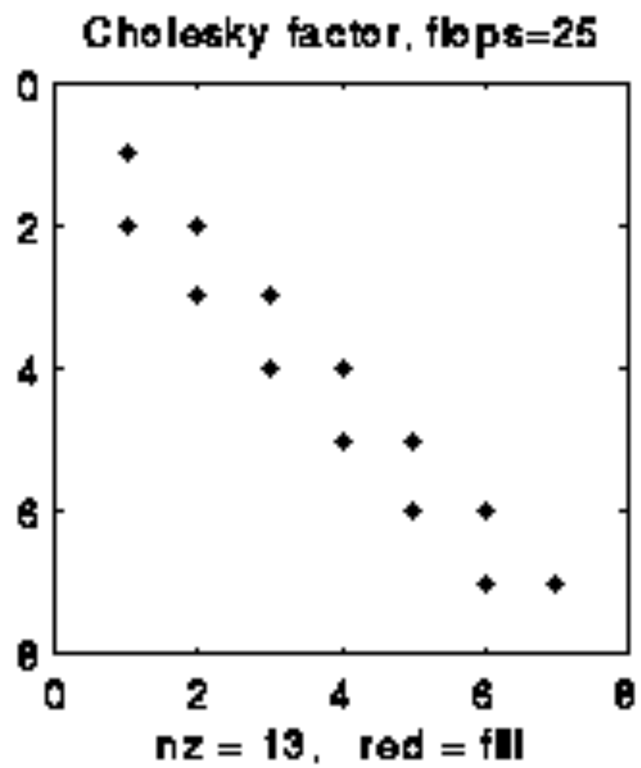
$$T(A) : \mathrm{parent}(j) = \min\{i > j : (i, j) \in G^+(A)\}$$

Elimination tree defines column dependencies in A.

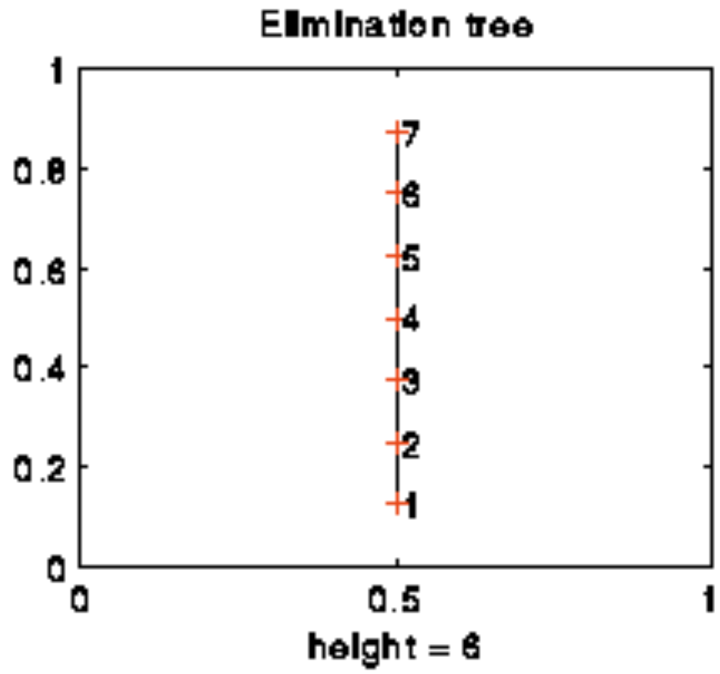
Can get T(A) from G(A) in *O(nnz * α(nnz))*, and G^+(A) from T(A) in O(nnz(L)).



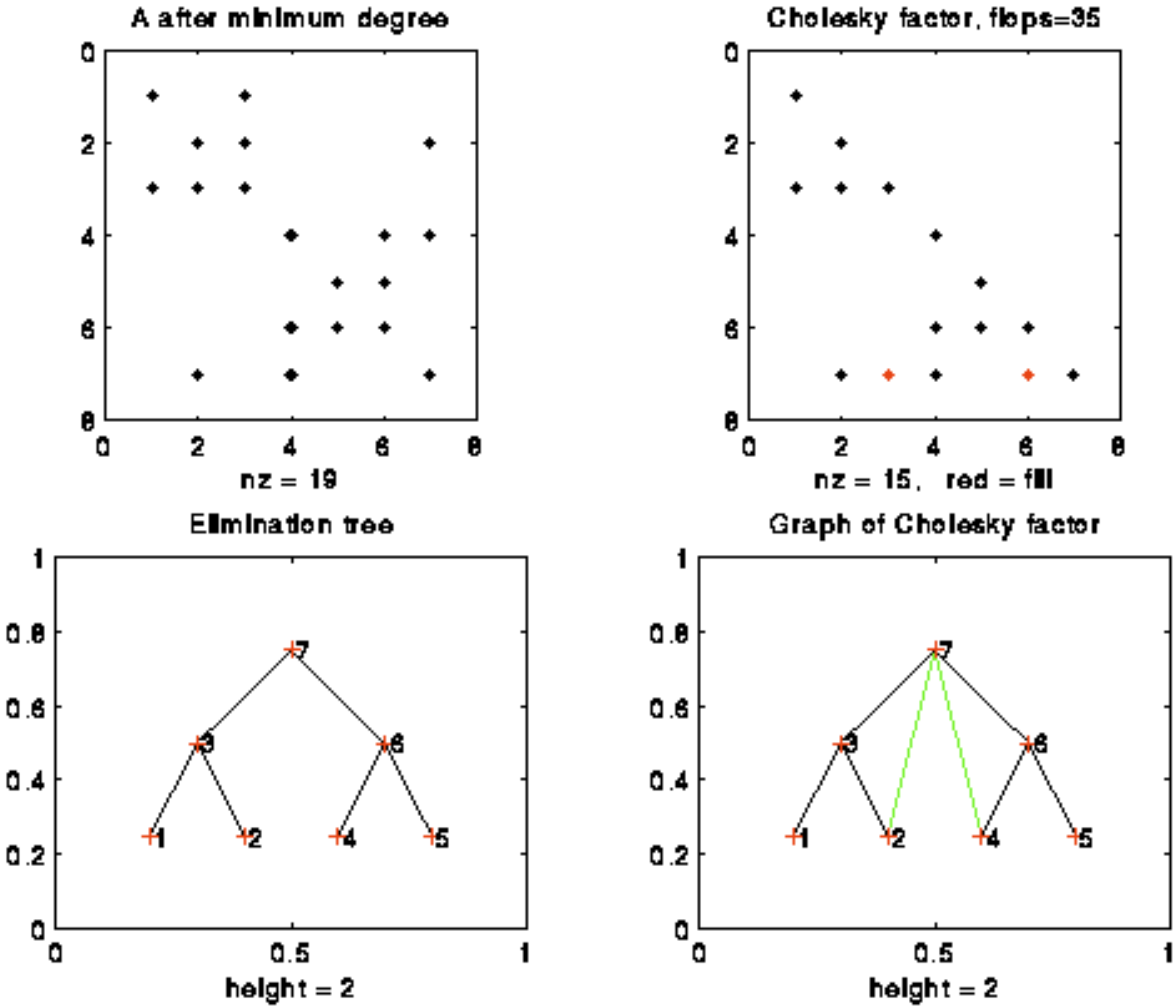
A in natural order
nz = 19

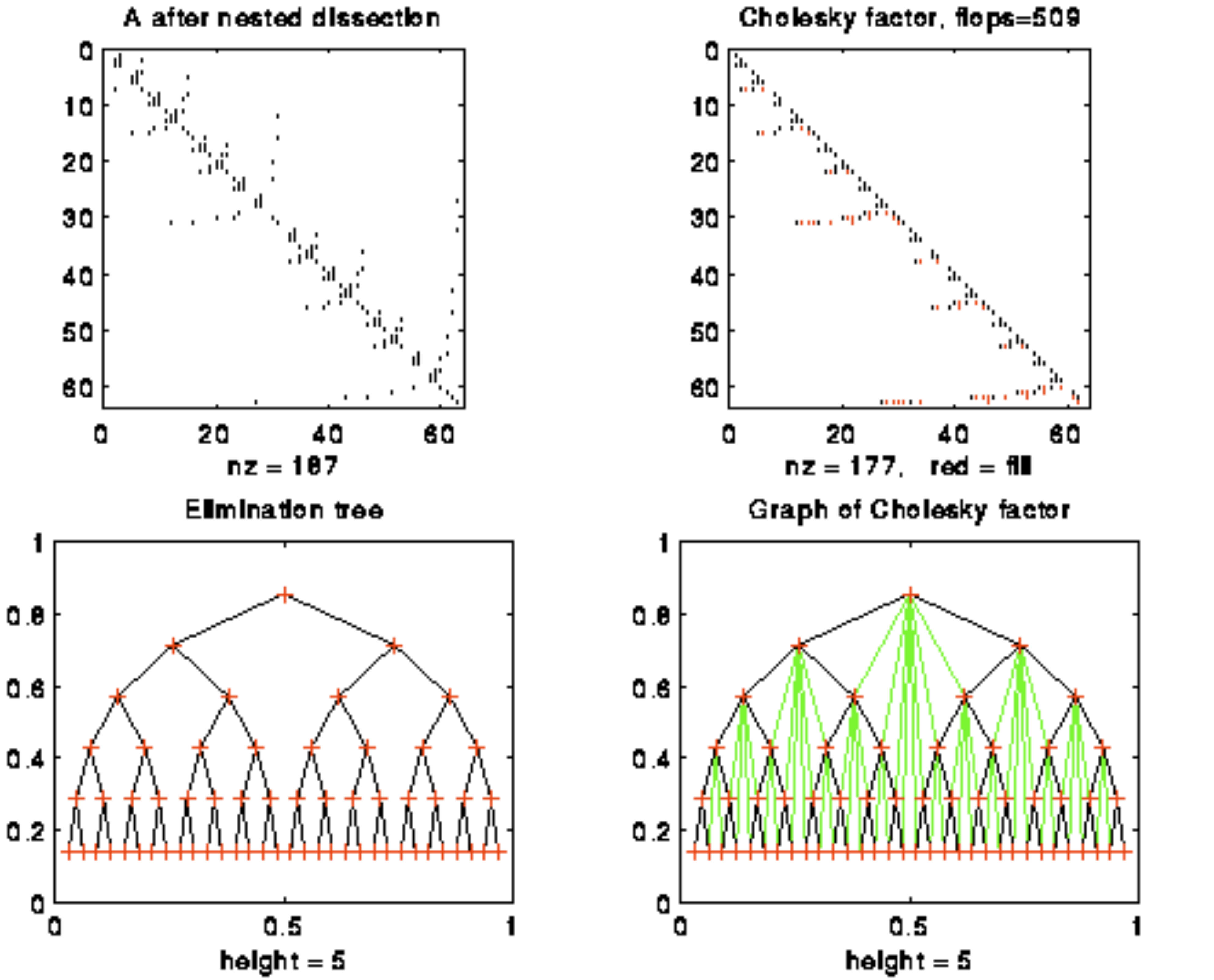


Cholesky factor, flops=25
nz = 13, red = fill



Elimination tree
height = 6

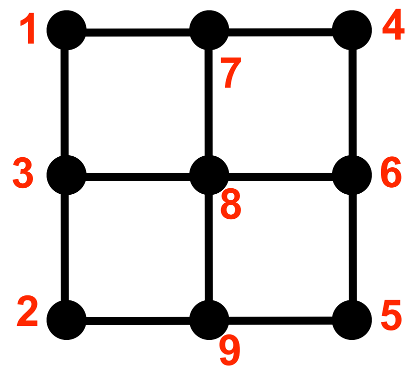

A after minimum degree
nz = 19
Cholesky factor, flops=35
nz = 15, red = fill
Elimination tree
height = 2
Graph of Cholesky factor
height = 2


A after nested dissection
nz = 187
Cholesky factor, flops=509
nz = 177, red = fill
Elimination tree
height = 5
Graph of Cholesky factor
height = 5

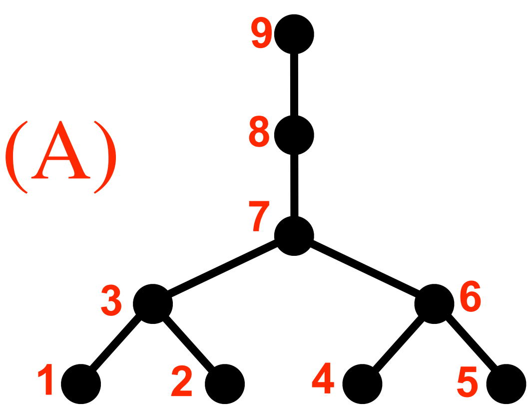
Multifrontal methods (*e.g.*, [Liu '92])

$$
\begin{aligned}
A &= \begin{pmatrix} B & V^T \\ V & C \end{pmatrix} \\
&= \begin{pmatrix} L_B & 0 \\ V L_B^{-T} & I \end{pmatrix} \begin{pmatrix} I & 0 \\ 0 & C - V B^{-1} V^T \end{pmatrix} \begin{pmatrix} L_B^T & L_B^{-1} V^T \\ 0 & I \end{pmatrix} \\
\text{where:} & \\
B &= L_B L_B^T \text{ (Cholesky factorization)} \\
\text{Observe:} & \\
-V B^{-1} V^T &= -(V L_B^{-T})(L_B^{-1} V^T) = -\sum_{k=1}^{j-1} \begin{pmatrix} l_{j,k} \\ \vdots \\ l_{n,k} \end{pmatrix} \begin{pmatrix} l_{j,k} \cdots l_{n,k} \end{pmatrix}
\end{aligned}
$$

G(A)

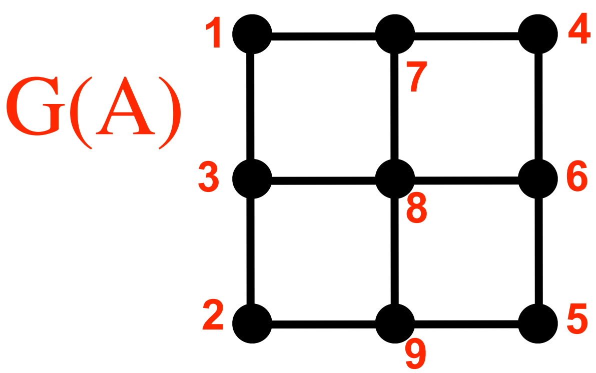


T(A)



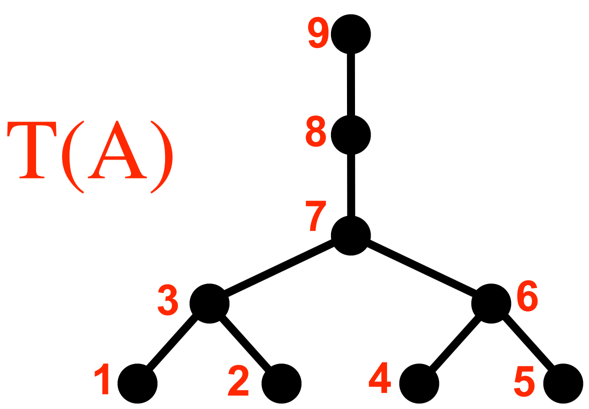
	1	2	3	4	5	6	7	8	9
1	●		●				●		
2		●	●						●
3	●	●	●					●	
4				●		●	●		
5					●	●			●
6				●	●	●		●	
7	●			●			●	●	
8			●			●	●	●	●
9		●			●			●	●

A

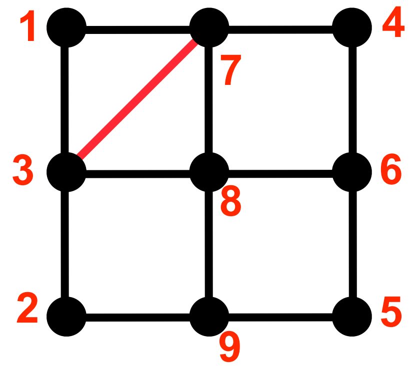


For each node of T from leaves to root:

- Sum own row/col of A with children's *Update* matrices into *Frontal* matrix
- Eliminate current variable from *Frontal* matrix, to get *Update* matrix
- Pass *Update* matrix to parent



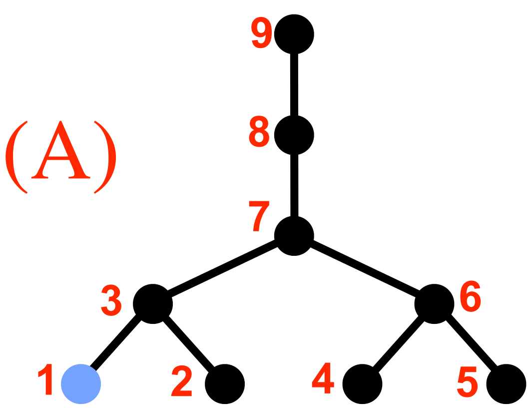
G(A)

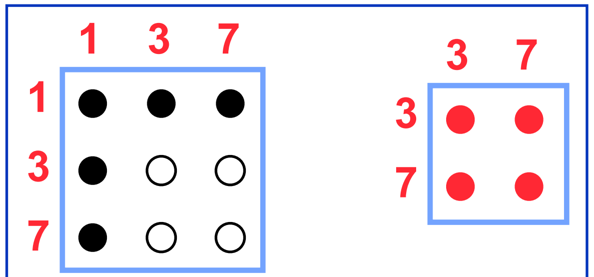


For each node of T from leaves to root:

- Sum own row/col of A with children's *Update* matrices into *Frontal* matrix
- Eliminate current variable from *Frontal* matrix, to get *Update* matrix
- Pass *Update* matrix to parent

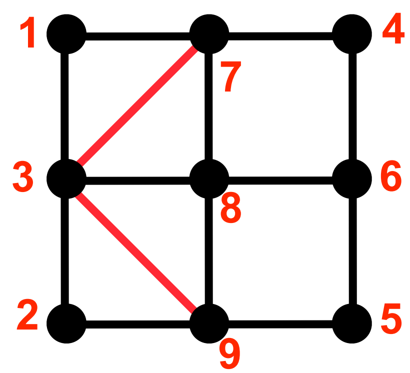
T(A)





$F_1 = A_1 \quad \Rightarrow U_1$

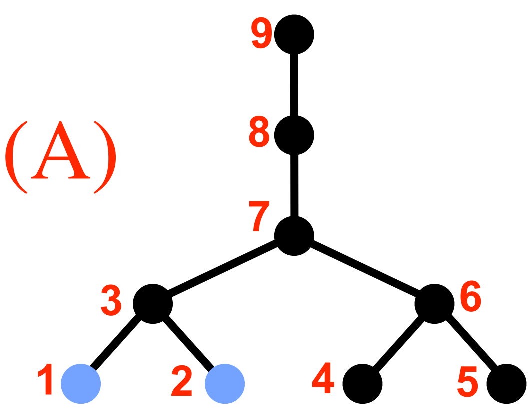
G(A)

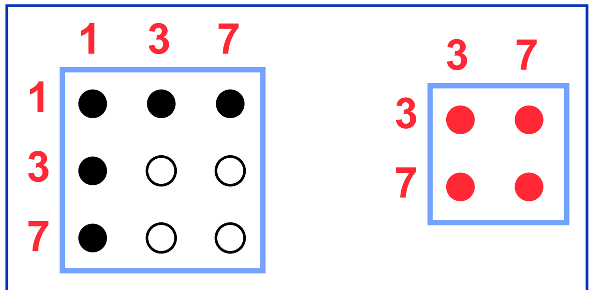


For each node of T from leaves to root:

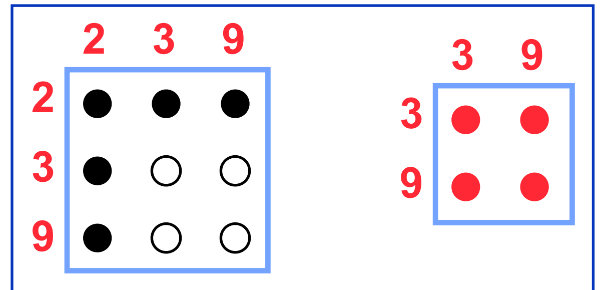
- Sum own row/col of A with children's *Update* matrices into *Frontal* matrix
- Eliminate current variable from *Frontal* matrix, to get *Update* matrix
- Pass *Update* matrix to parent

T(A)

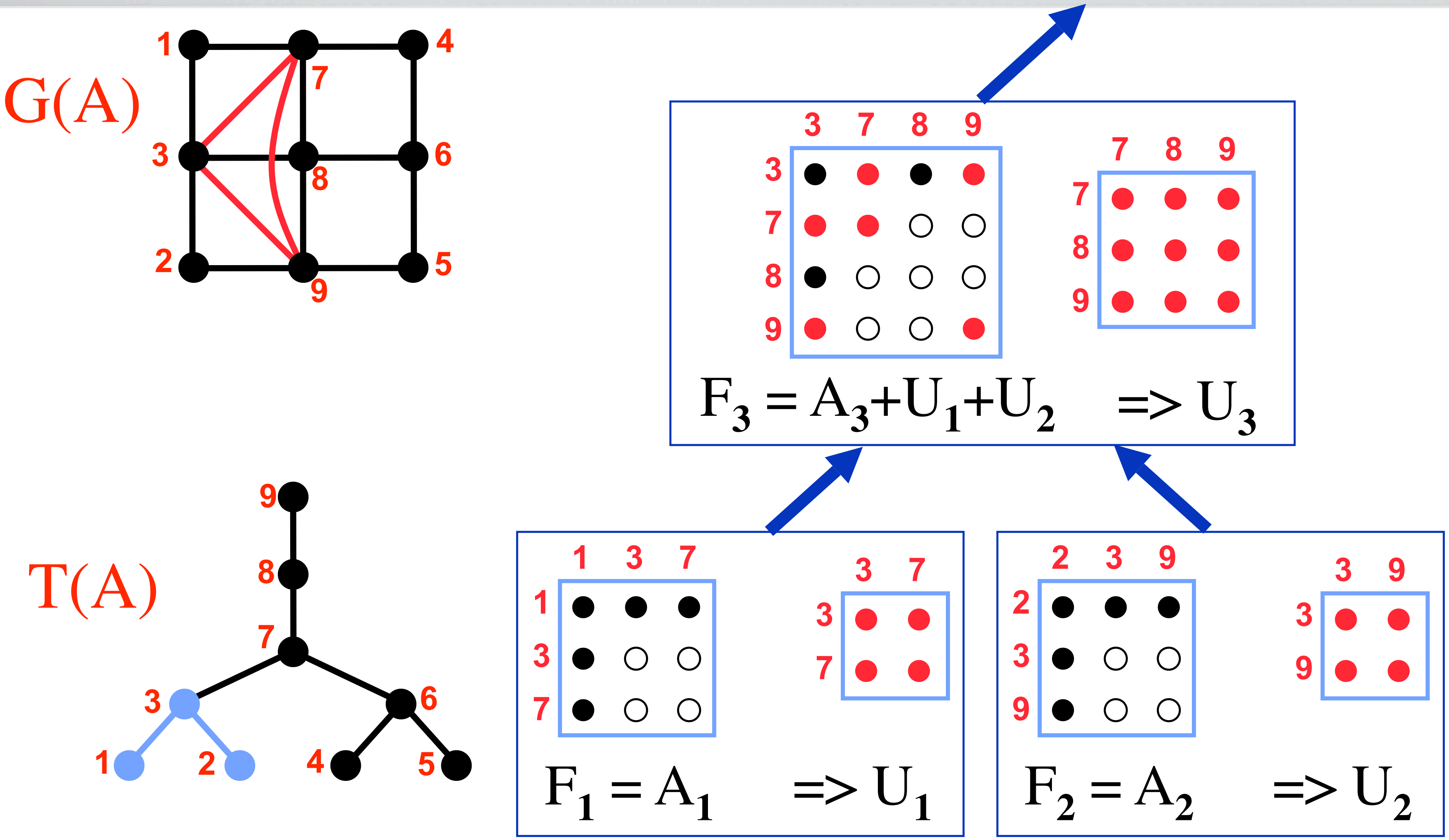




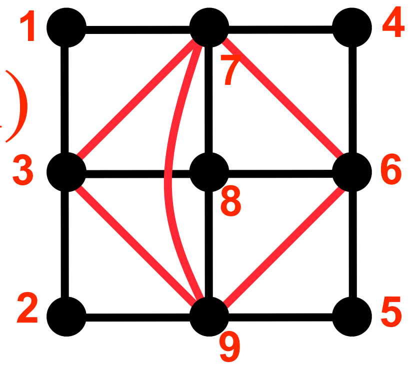
$F_1 = A_1 \Rightarrow U_1$



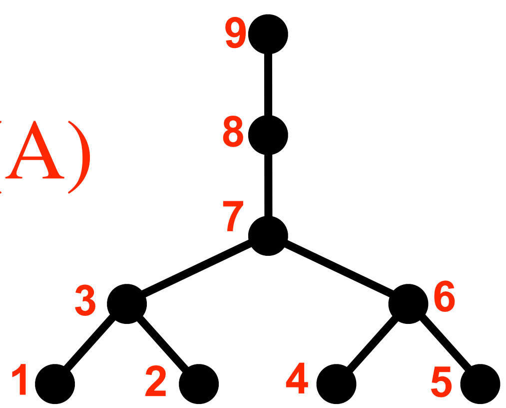
$F_2 = A_2 \Rightarrow U_2$

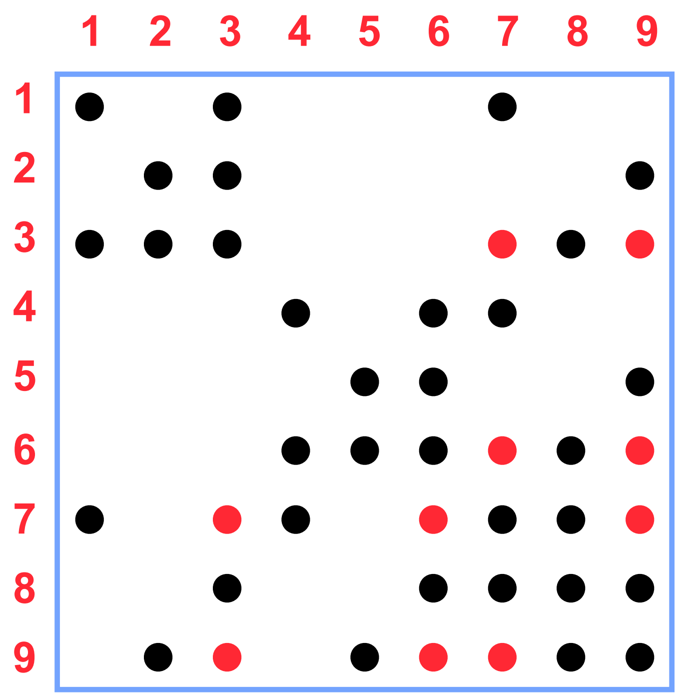

G(A)
T(A)
F3 = A3+U1+U2 => U3
F1 = A1 => U1
F2 = A2 => U2

G+(A)



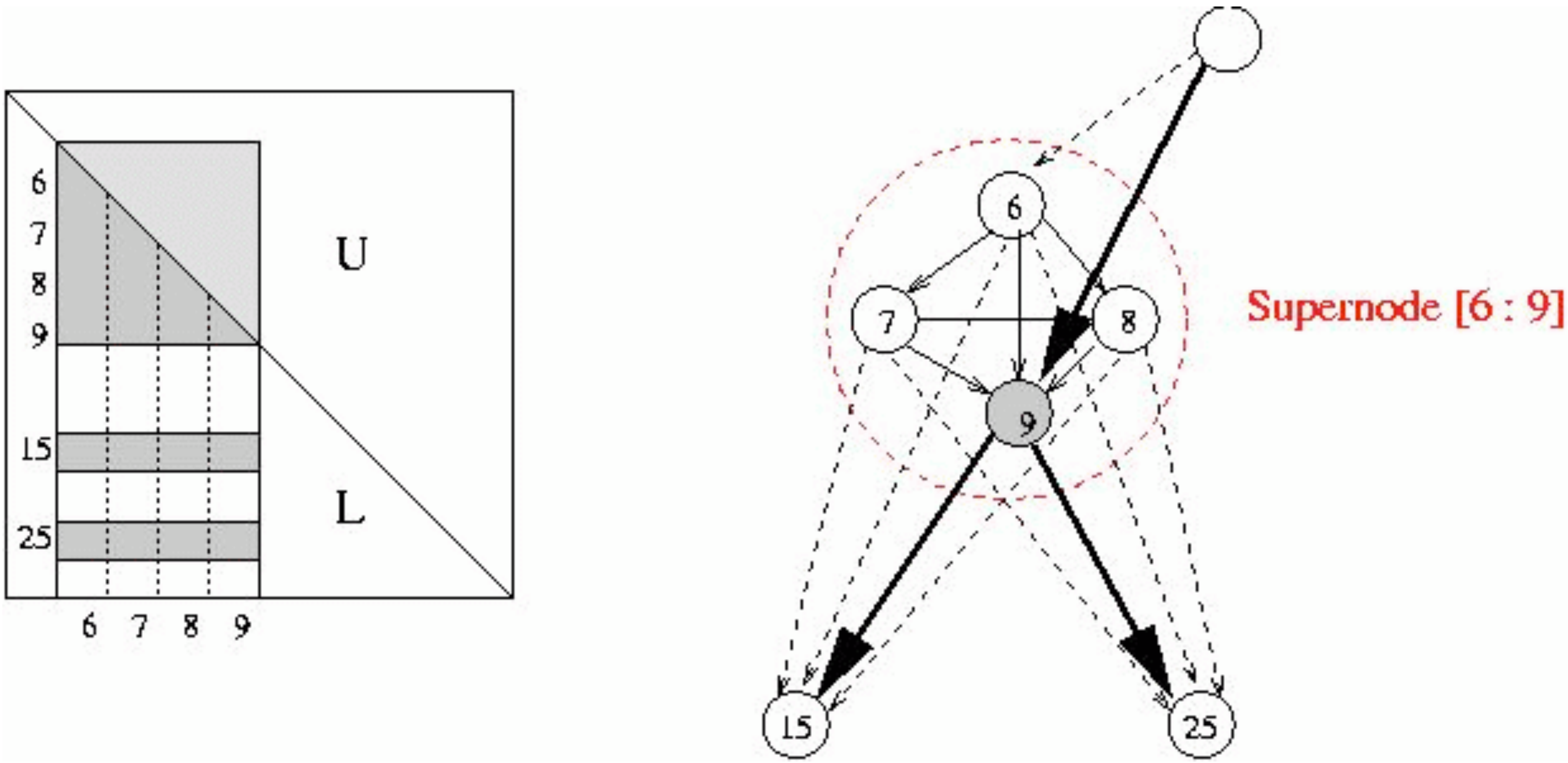
T(A)

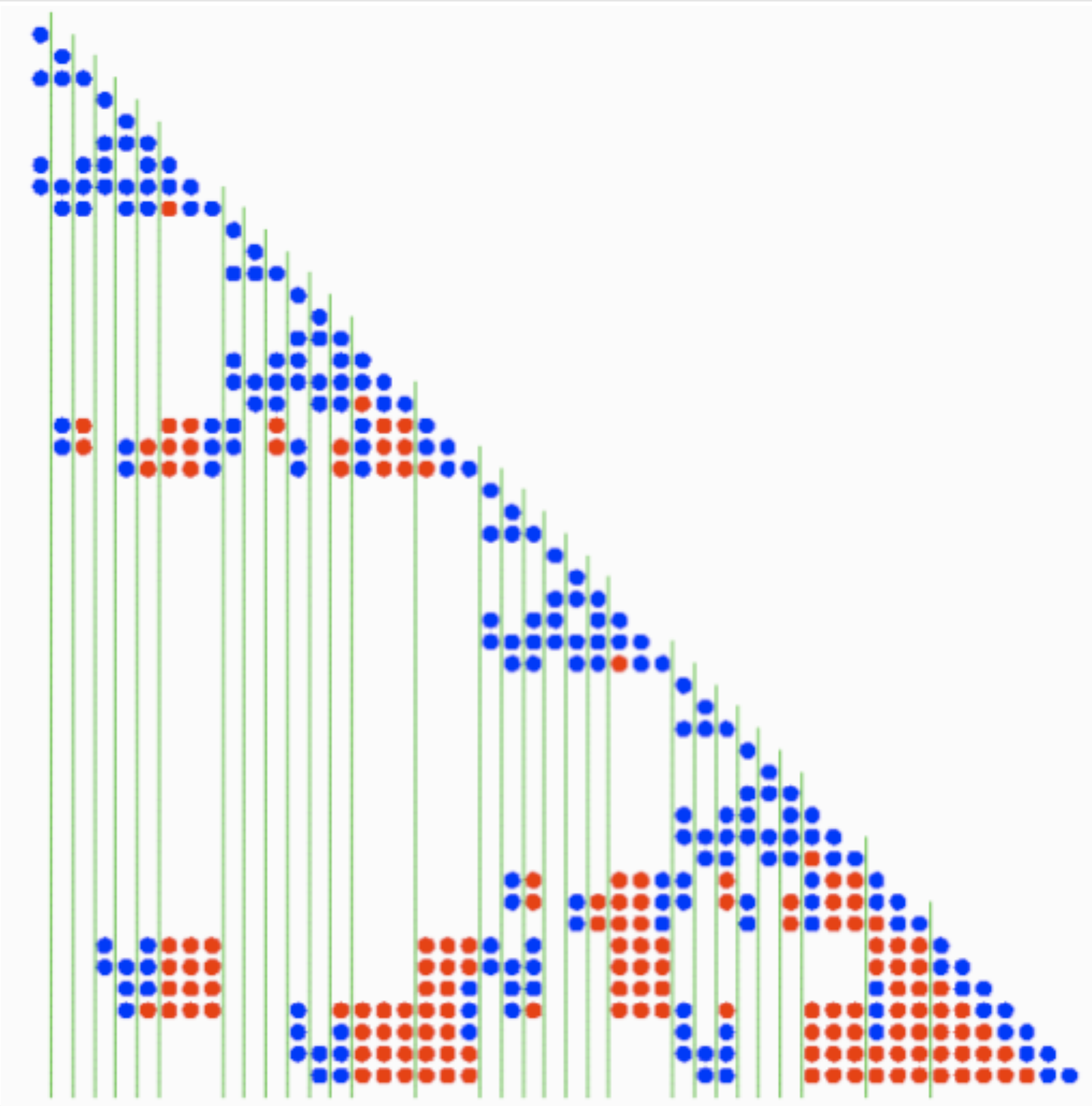


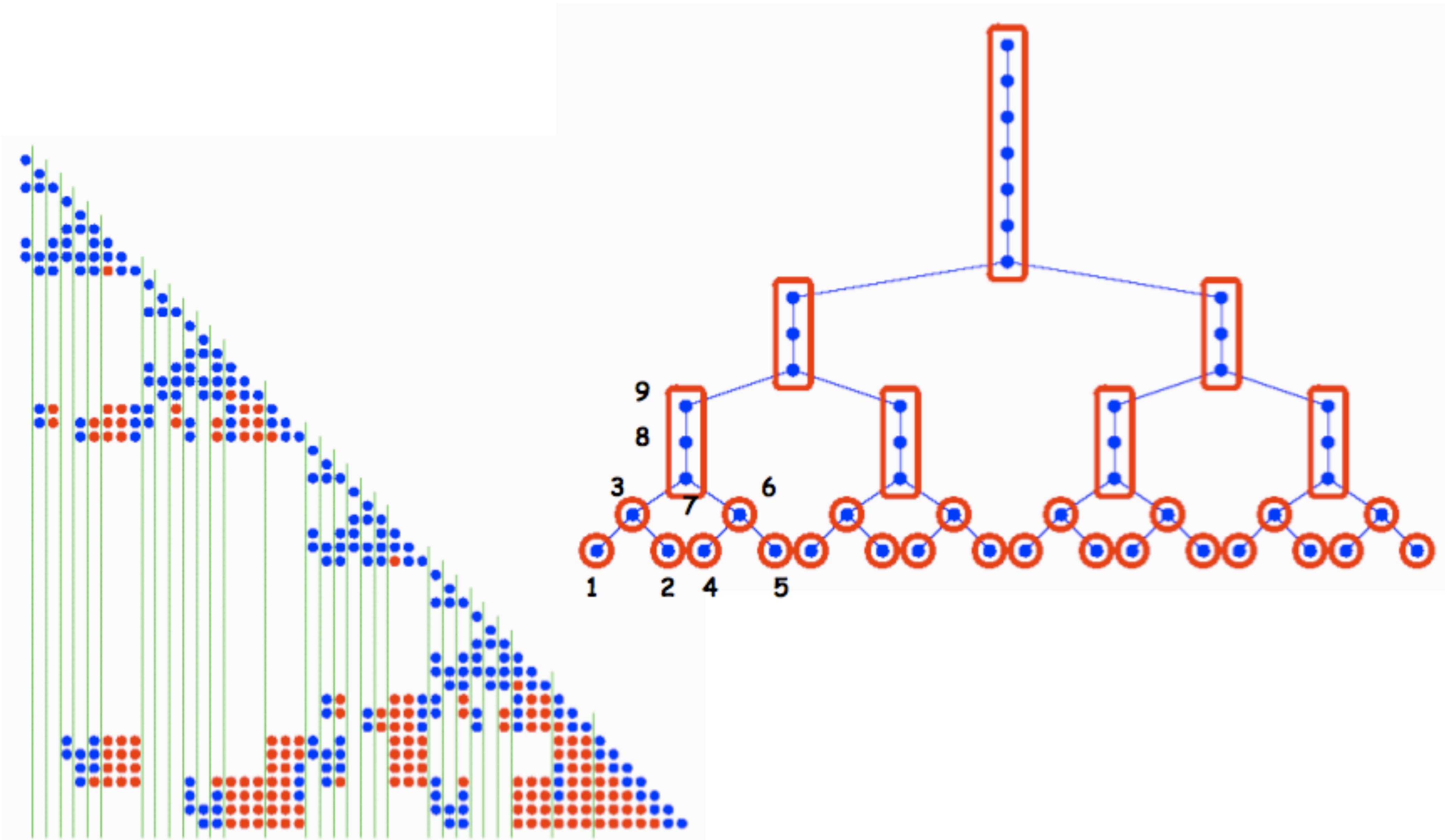


L+U

Improving memory behavior: Supernodes


6
7
8
9
15
25
6 7 8 9
U
L
6
7
8
9
15
25
Supernode [6 : 9]



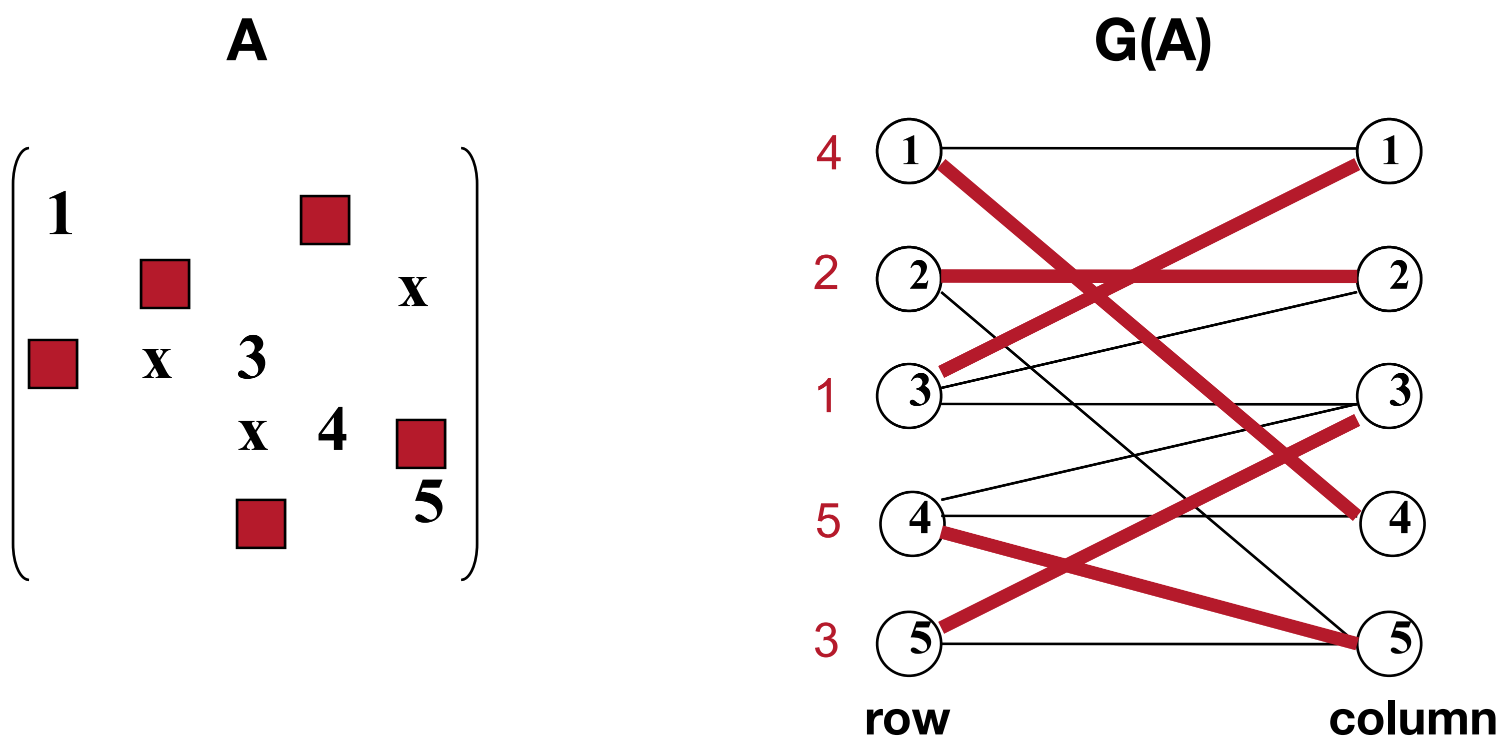

9
8
3
6
7
1
2
4
5

Complications in the unsymmetric (sparse LU) case

Need for numerical pivoting in LU

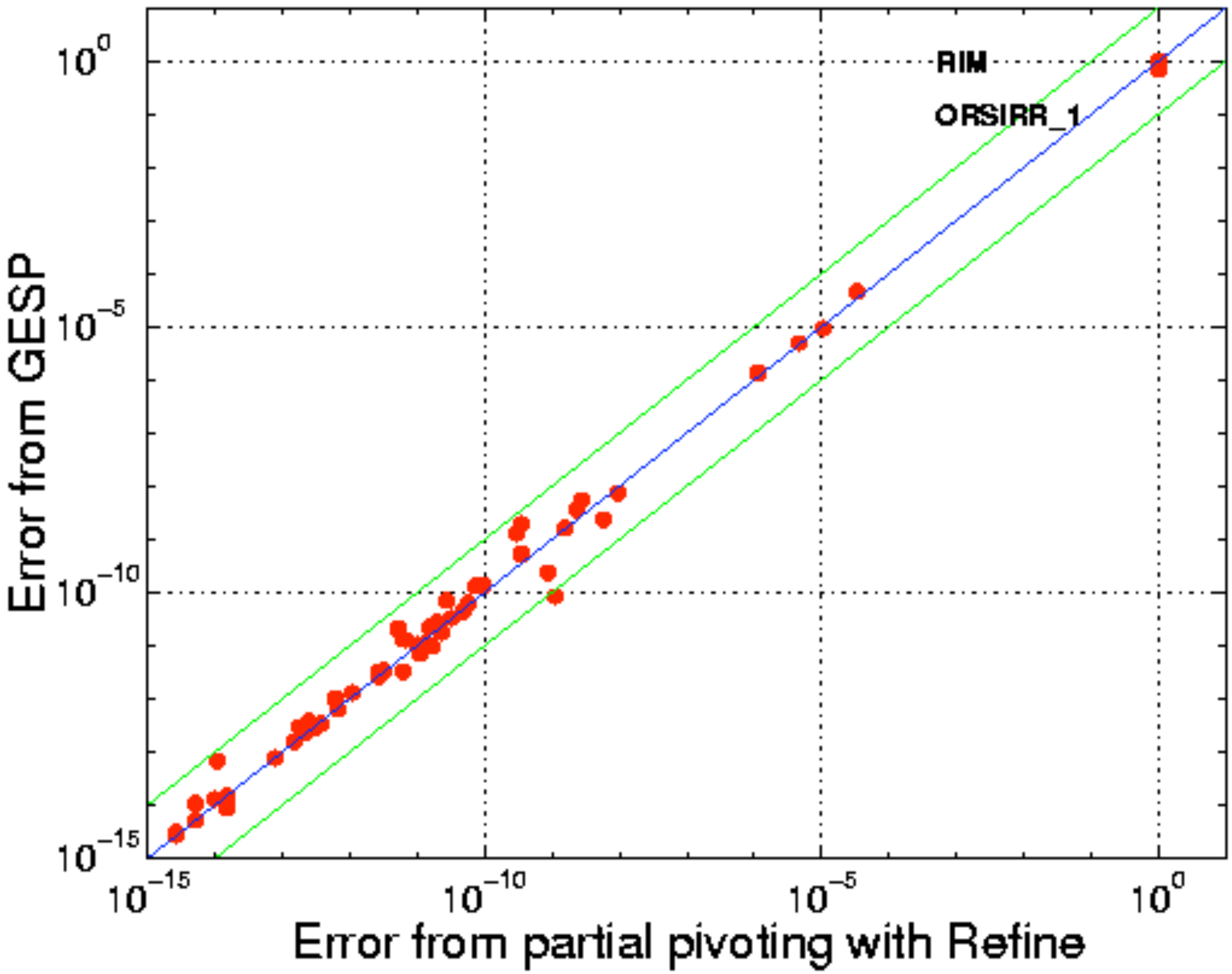
- Partial pivoting: Hard to implement scalably for sparse factorization
  - Not needed if A is SPD
  - Low cost in dense case, but not in sparse
- Alternative: **Static pivoting** (*e.g.*, SuperLU_DIST)
  - Before factor, scale and permute A to maximize diagonal: $P_r*D_r*A*D_c = A'$
  - Find P_r by **weighted bipartite matching** on G(A)
  - During factorization, replace tiny pivots with $\sqrt{\epsilon}||A||$
  - Iterative refinement if necessary (future lecture)

Static pivoting via weighted bipartite matching



- Maximize diagonal entries (sum or product [sum of logs])
  - “Hungarian” algorithm (*e.g.*, MC64): O(n*(m+n)*log n)
  - Auction algorithm (more parallel): O(n*m*log(n*C))

Source: X. Li (2006)



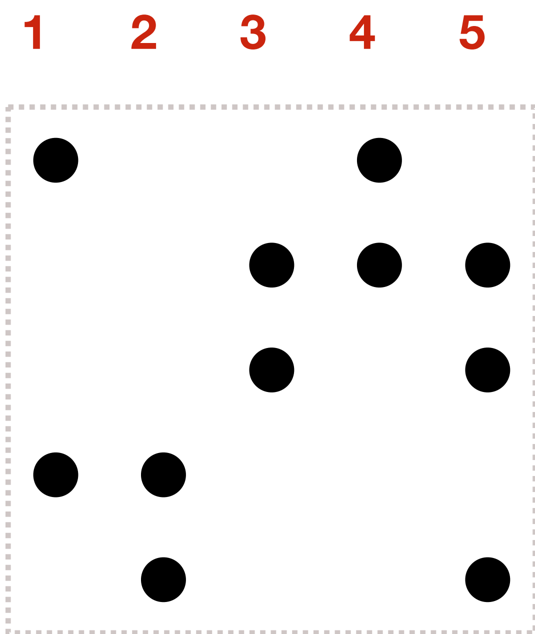
Source: X. Li (2006)

Ordering for sparse LU

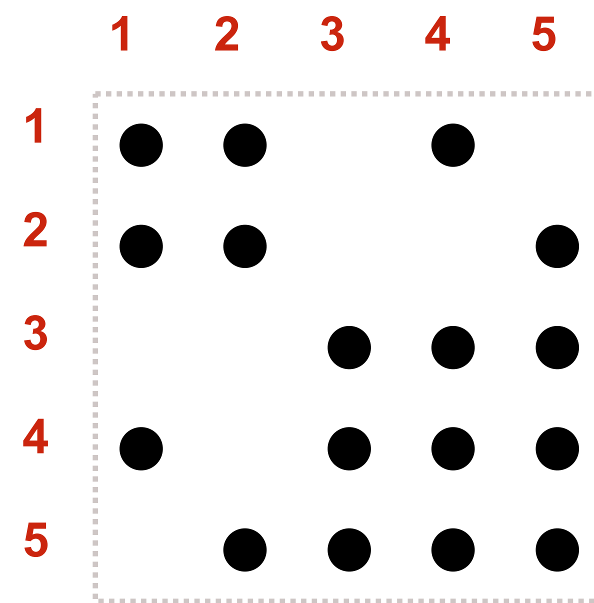
- Earlier examples implicitly show or assume symmetric structure
- Options in the unsymmetric case
  - Symmetric ordering for A^TA, based on theorem [George & Ng '87]:

$$R^T R = A^T A \quad \text{and} \quad PA = LU$$
$$\Longrightarrow \text{struct}(L + U) \quad \subseteq \quad \text{struct}(R + R^T) \text{ for any } P$$

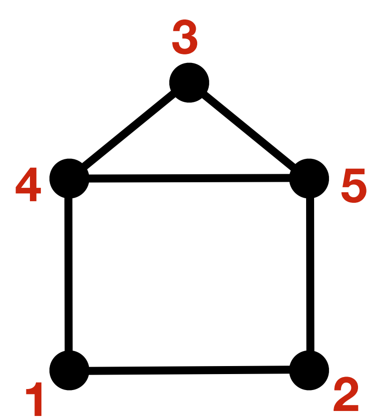
  - Static pivoting (shortly): Similar theorem holds for $R^TR = A^T+A$ (without P)
    - "Symmetrization-free" [Amestoy, Li, & Ng '03]



A



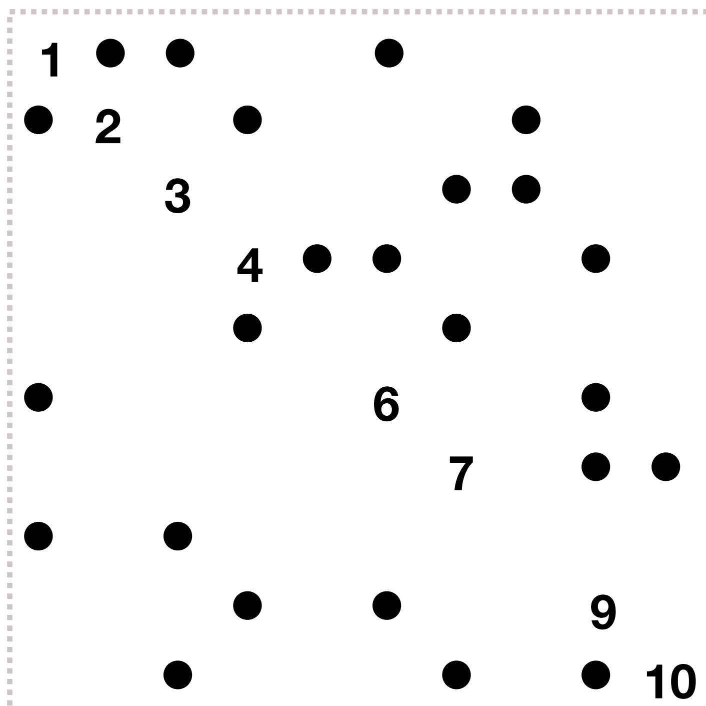
A^TA



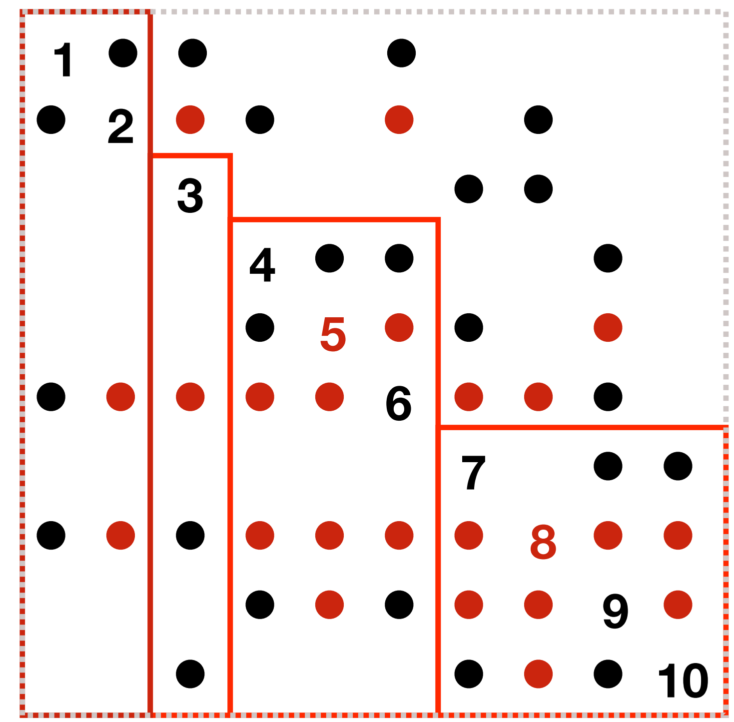
Problem: Elimination tree (DAG) not known until pivoting is done

- Symbolic and numerical factorization stages become interleaved
- Option 1: Use elimination tree of A^TA
  - Bound sparsity, parallelism
  - Update tree on-the-fly [Demmel, Gilbert, & Li ’97]
- Option 2: Pivot statically

Supernode structure in L, not U

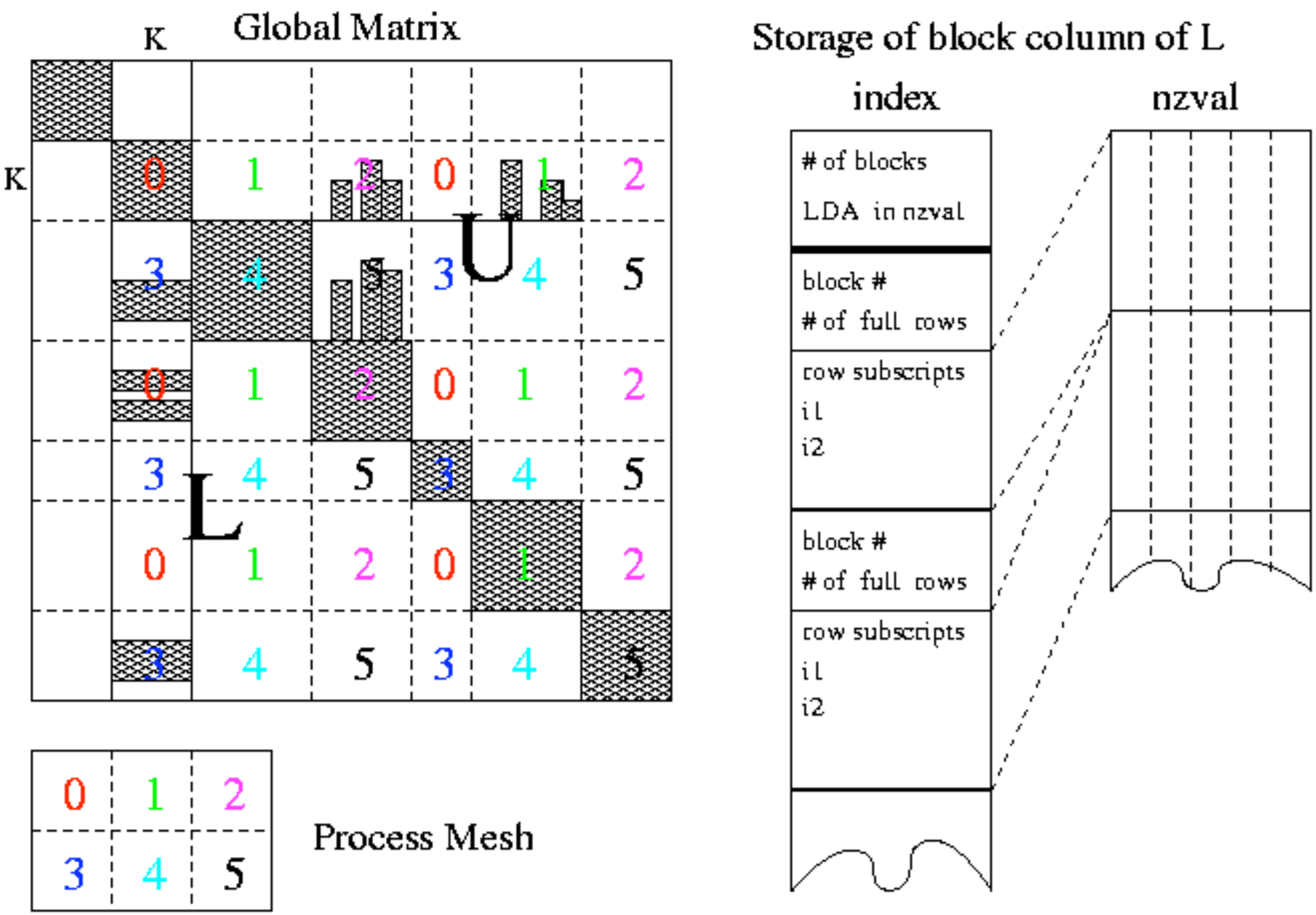


Original matrix A



Factors L+U

Data layout/distribution



Global Matrix
K
K
0
1
2
0
1
2
3
4
5
3
U
4
5
0
1
2
0
1
2
3
L
4
5
3
4
5
0
1
2
0
1
2
3
4
5
3
4
5
Process Mesh
0
1
2
3
4
5
Storage of block column of L
index
nzval
# of blocks
LDA in nzval
block #
# of full rows
row subscripts
i1
i2
block #
# of full rows
row subscripts
i1
i2

“In conclusion…”

Sparse LU software

- SuperLU [Xiaoye “Sherry” Li @ LBNL]
- MUMPS [Amestoy, *et al*.]
- UMFPACK [Davis @ UFL]
- WSMP [Gupta @ IBM]

Open problems (suggested by Sherry Li ‘07)

- Optimizing parallel performance: blocking, tuning, reducing/hiding latency
- Graph partitioning ordering for unsymmetric case
- Scalability of sparse triangular solve
- Efficient incomplete-LU (ILU)
- Optimal complexity sparse factorization
  - Fast multipole for matrix inversion — J. Xia’s dissertation (UCB, now @ UCLA)
- Latency-avoidant LU & QR

Landscape of Ax=b solvers

	Iterative y’ = Ay	Direct A = LU
General A	GMRES, BiCGStab, ...	Pivoting LU
$A = A^T$	Conjugate gradient, …	Cholesky

More general

More robust

Less storage (if sparse)

Source: Gilbert (UCSB)

Administrivia

Administrative stuff

- **New room** (dumpier, but cozier?): College of Computing Building **(CCB) 101**
- **Accounts**: Apparently, you already have them
- Front-end login node: **ccil.cc.gatech.edu** (CoC Unix account)
  - We “own” **warp43—warp56**
  - Some docs (**MPI**): http://www-static.cc.gatech.edu/projects/ihpcl/mpi.html
  - **Sign-up** for mailing list: https://mailman.cc.gatech.edu/mailman/listinfo/ihpc-lab

Homework 1: Parallel conjugate gradients

- Implement a parallel solver for Ax = b (serial C version provided)
  - Evaluate on three matrices: 27-pt stencil, and two application matrices
  - “Simplified:” No preconditioning
  - **Bonus**: Reorder, precondition
- Performance models to understand scalability of your implementation
  - Make measurements
  - Build predictive models
- Collaboration encouraged: Compare programming models or platforms