

A column pre-ordering strategy for the unsymmetric-pattern multifrontal method

Tim Davis

`davis@cise.ufl.edu`

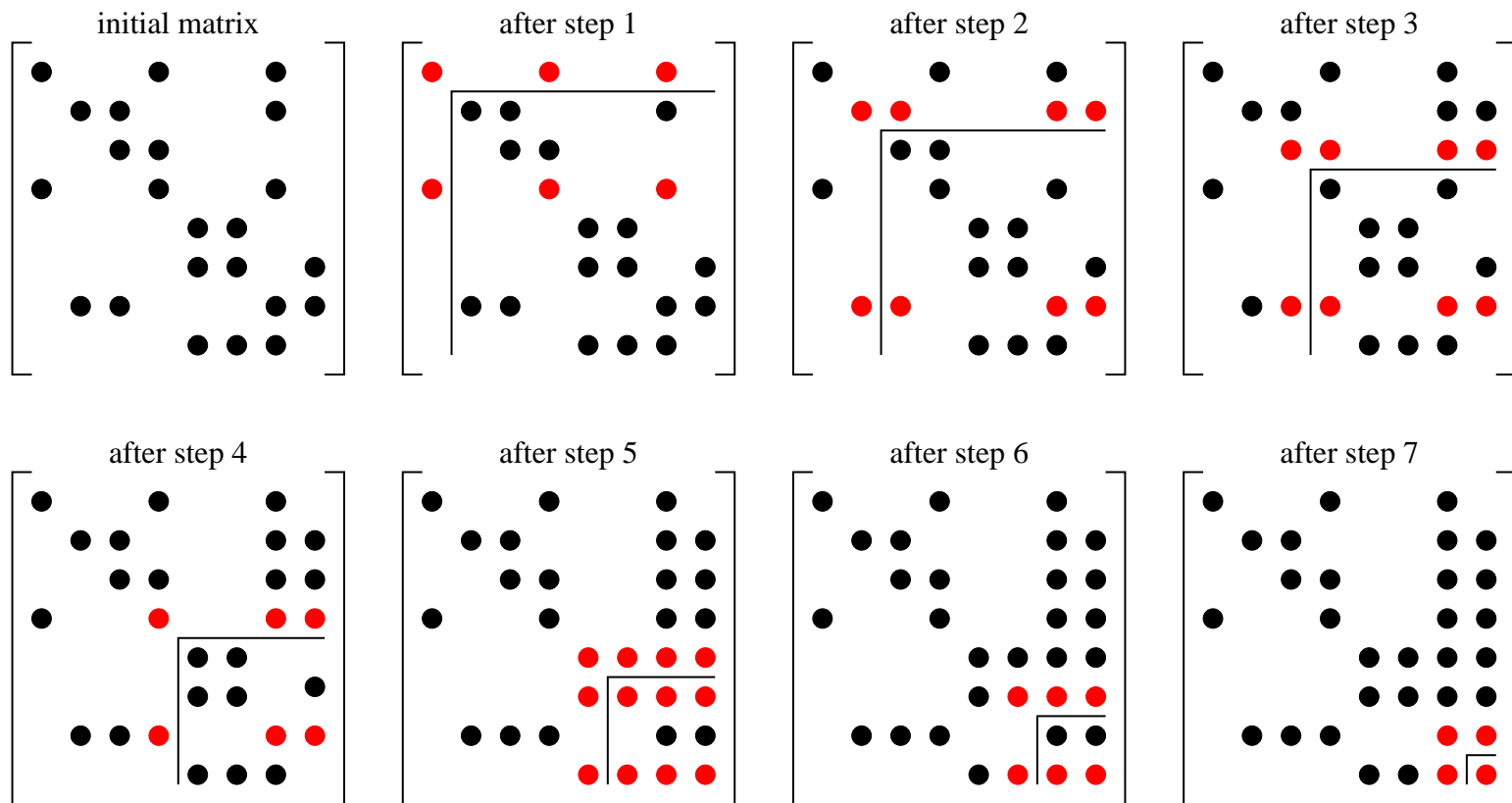
University of Florida, Stanford University, and Lawrence Berkeley National Laboratory

Outline

- symbolic LU factorization, $PA = LU$
- the column elimination tree, and supercolumns
- a multifrontal view of sparse partial pivoting
 - frontal matrices and unifrontal chains in the column elimination tree
 - UMFPACK numeric factorization
 - numerical assembly / degree update
 - representing the Schur complement
 - local pivot search
 - UMFPACK reordering and analysis phase
- experimental results

Symbolic LU factorization

- $PA = LU$, select P via partial pivoting
- example upper-bound symbolic LU factorization:



Symbolic LU factorization

- upper-bound symbolic LU factorization = a graph elimination process
- the column intersection graph
 - undirected hypergraph
 - n column nodes, each row = a hyperedge (or, a clique in a regular graph)
- at each step:
 - a new hyperedge/clique is constructed (union of all candidate pivot rows)
 - one or more prior hyperedges/cliques removed (the candidate pivot rows)
 - one column node removed
- pre-ordering: pick the column that results in the smallest new hyperedge/clique

Symbolic LU factorization

Let $\mathcal{A}_i = \text{Struct}(\mathbf{A}_{i*})$ for all i

Let $\mathcal{R}_i = \emptyset$ for all i

elimination phase:

for $k = 1$ **to** n **do**

find upper bound pattern of pivot row:

$$\mathcal{R}_k = \left(\bigcup_{k=\min \mathcal{R}_i} \mathcal{R}_i \right) \cup \left(\bigcup_{k=\min \mathcal{A}_i} \mathcal{A}_i \right) \setminus \{k\}$$

find upper bound number of nonzeros in pivot column:

$$l_k = \left(\sum_{k=\min \mathcal{R}_i} l_i \right) + |\{i : k = \min \mathcal{A}_i\}| - 1$$

regular row absorption:

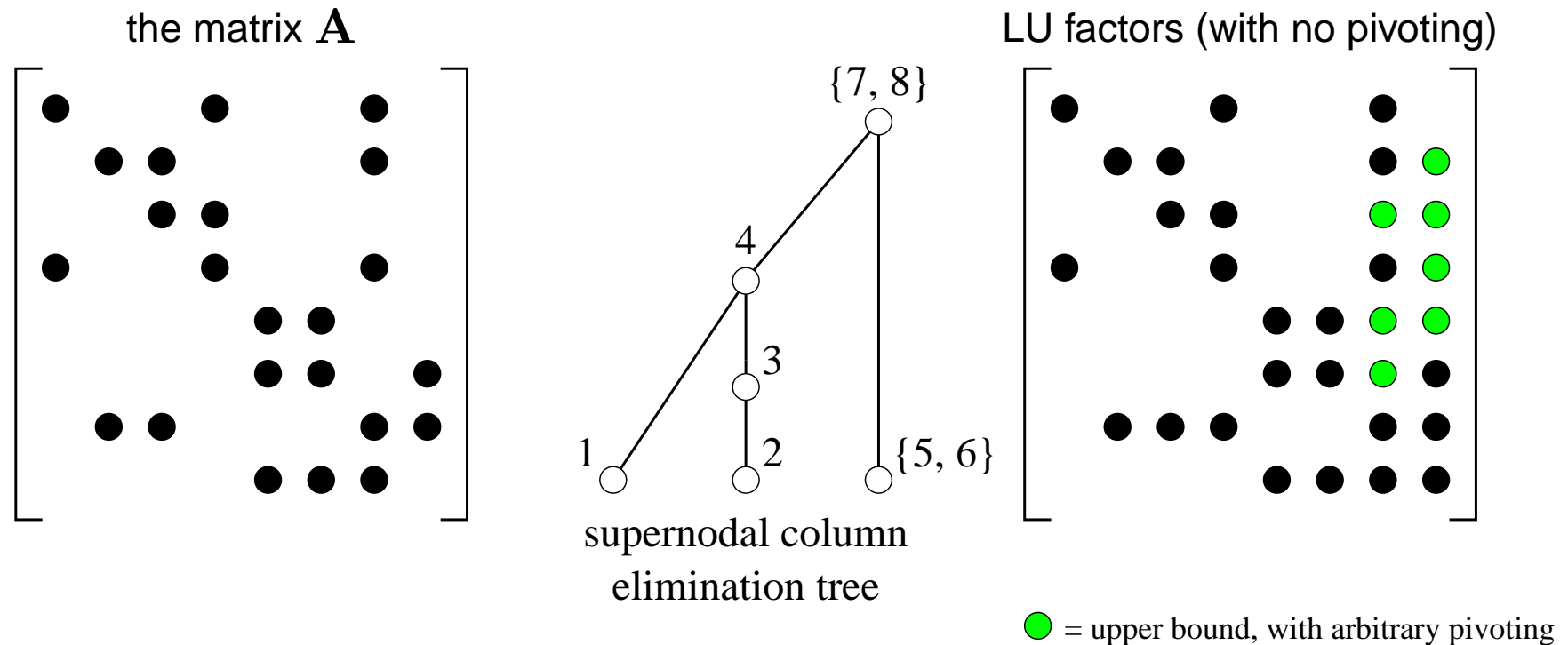
for each i **such that** $k = \min \mathcal{R}_i$ **do** $\mathcal{R}_i = \emptyset$ **end for**

for each i **such that** $k = \min \mathcal{A}_i$ **do** $\mathcal{A}_i = \emptyset$ **end for**

if $l_k = 0$ **then** $\mathcal{R}_k = \emptyset$ **end if**

end for

Column elimination tree



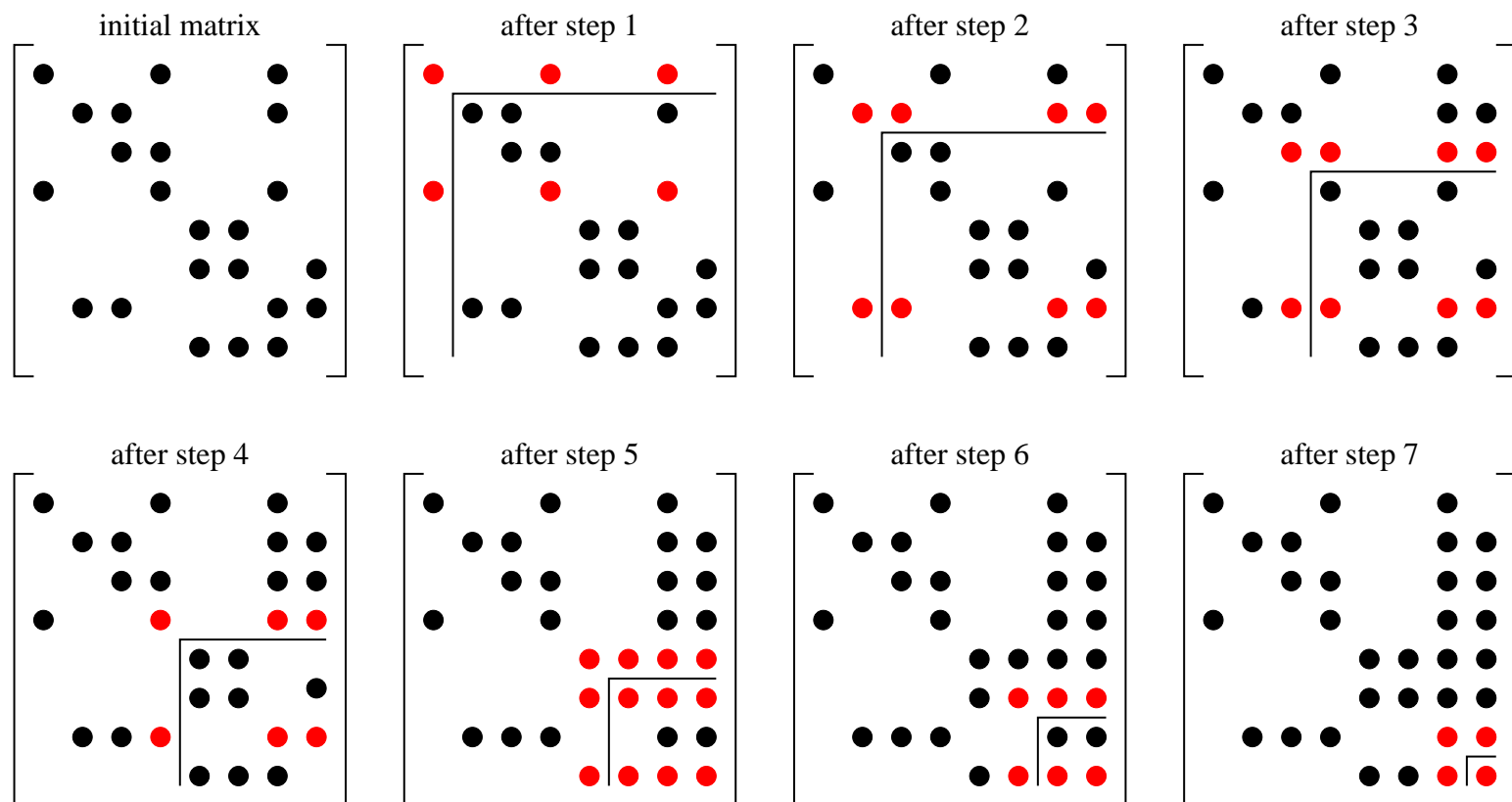
- parent of node $k = \min \mathcal{R}_k$
- row-merge tree: parent of original row i is $\min \mathcal{A}_i$

Supercolumns

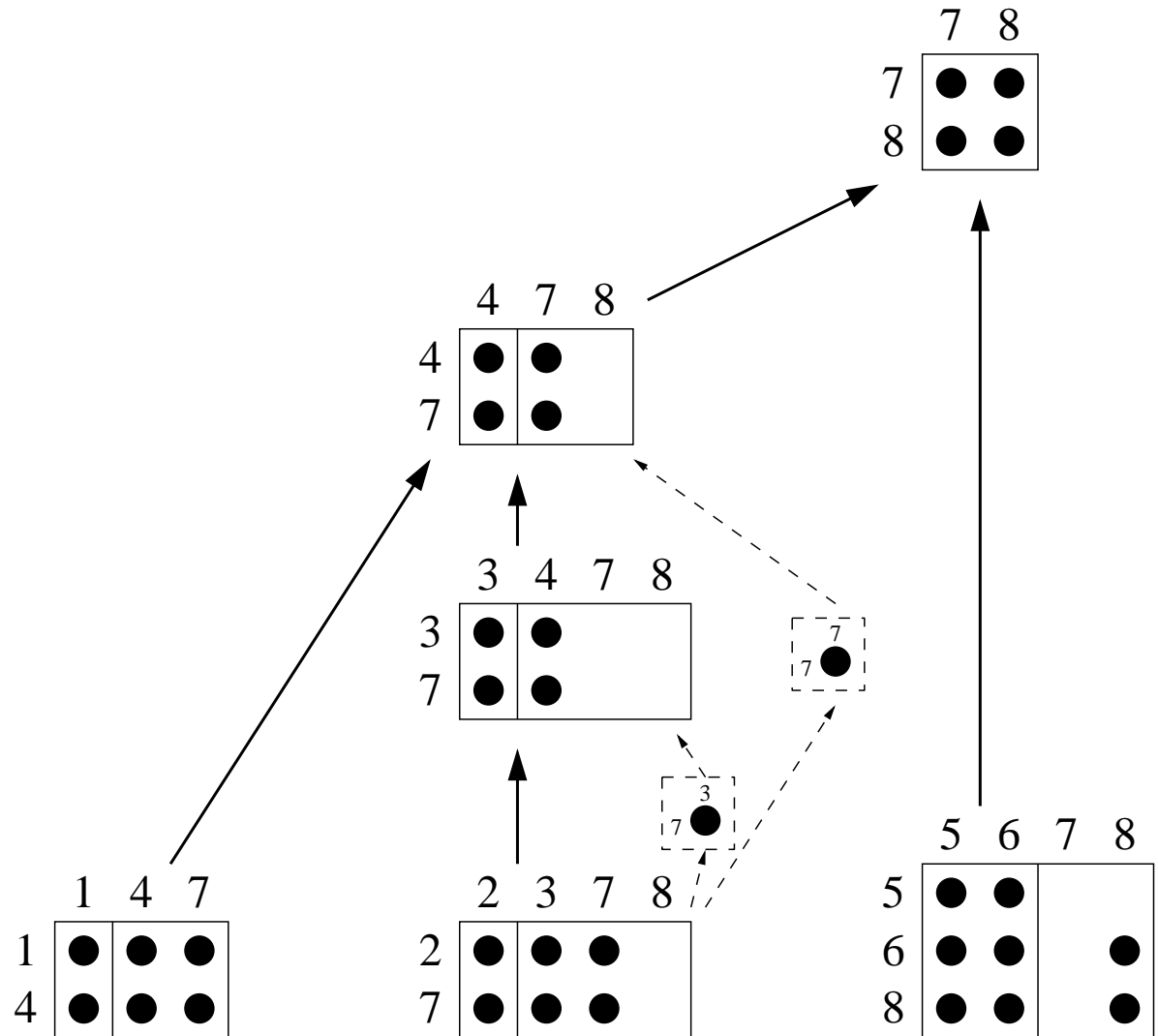
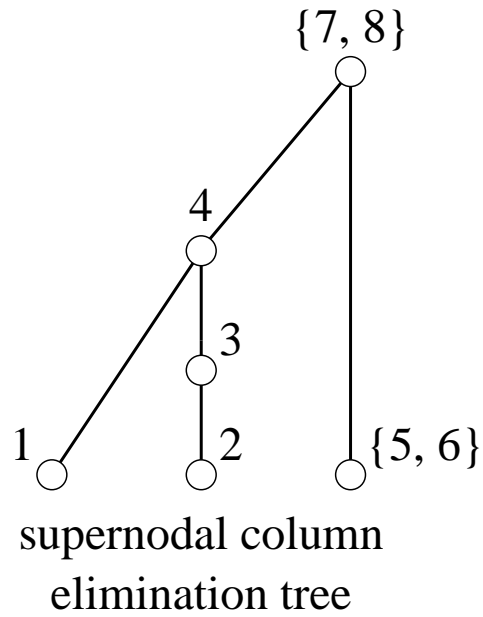
- a sequence of columns with identical upper-bound nonzero pattern
- identical upper-bound on corresponding pivot rows
 - $\mathcal{R}_k = \mathcal{R}_{k-1} \setminus \{k\}$
 - $l_k = l_{k-1} - 1$
- faster symbolic factorization, and column reordering
- allows for dense matrix BLAS in numerical factorization
 - BLAS $2\frac{1}{2}$ in SuperLU (repeated matrix-vector multiply)
 - BLAS 3 in UMFPACK (matrix-matrix multiply)
- columns within the supercolumn can be arbitrarily reshuffled without modifying the upper bound

A multifrontal view of the column etree

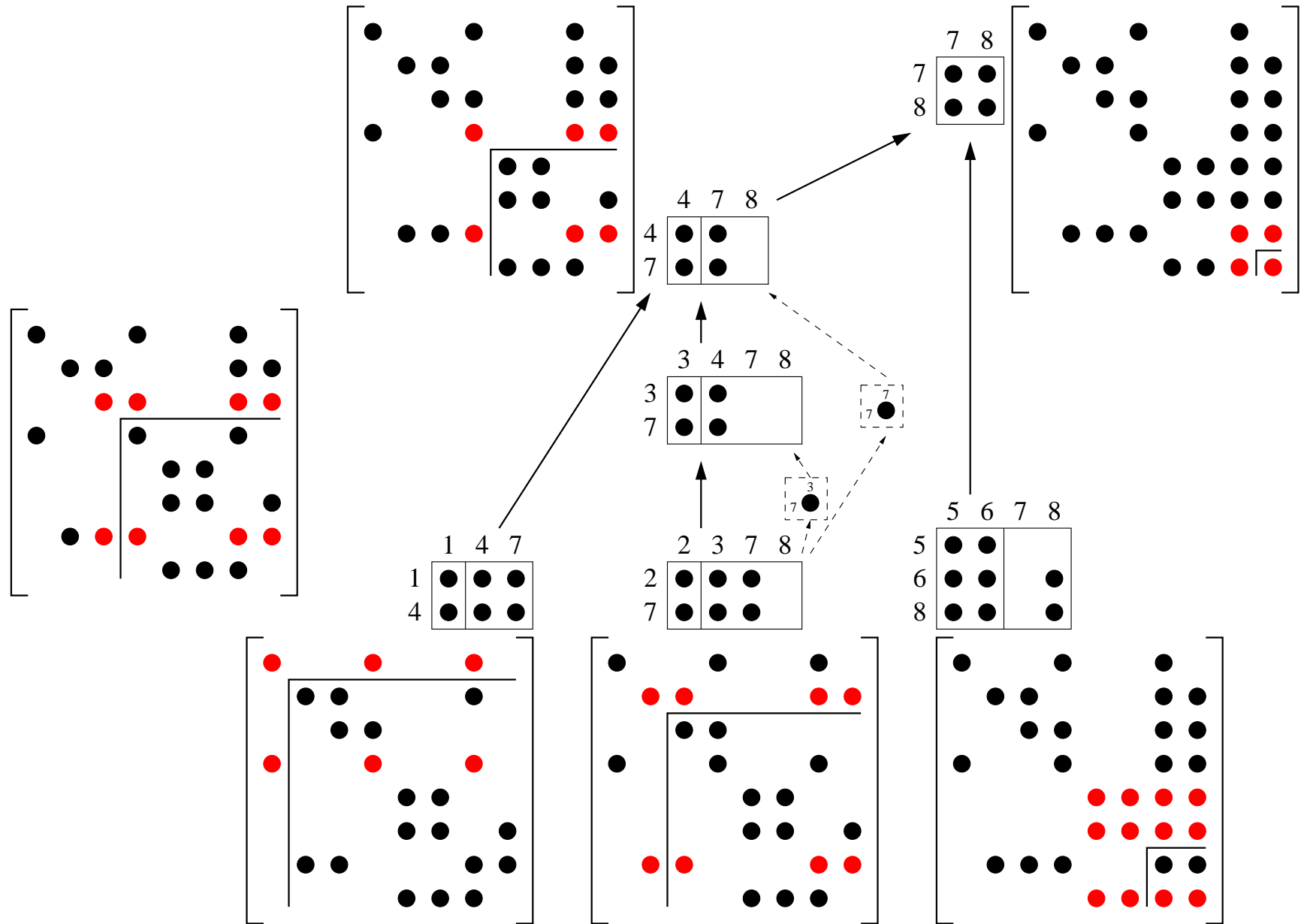
- each node in the supernodal column etree = rectangular frontal matrix
- size at most l_k -by- $|\mathcal{R}_k|$, actual size smaller



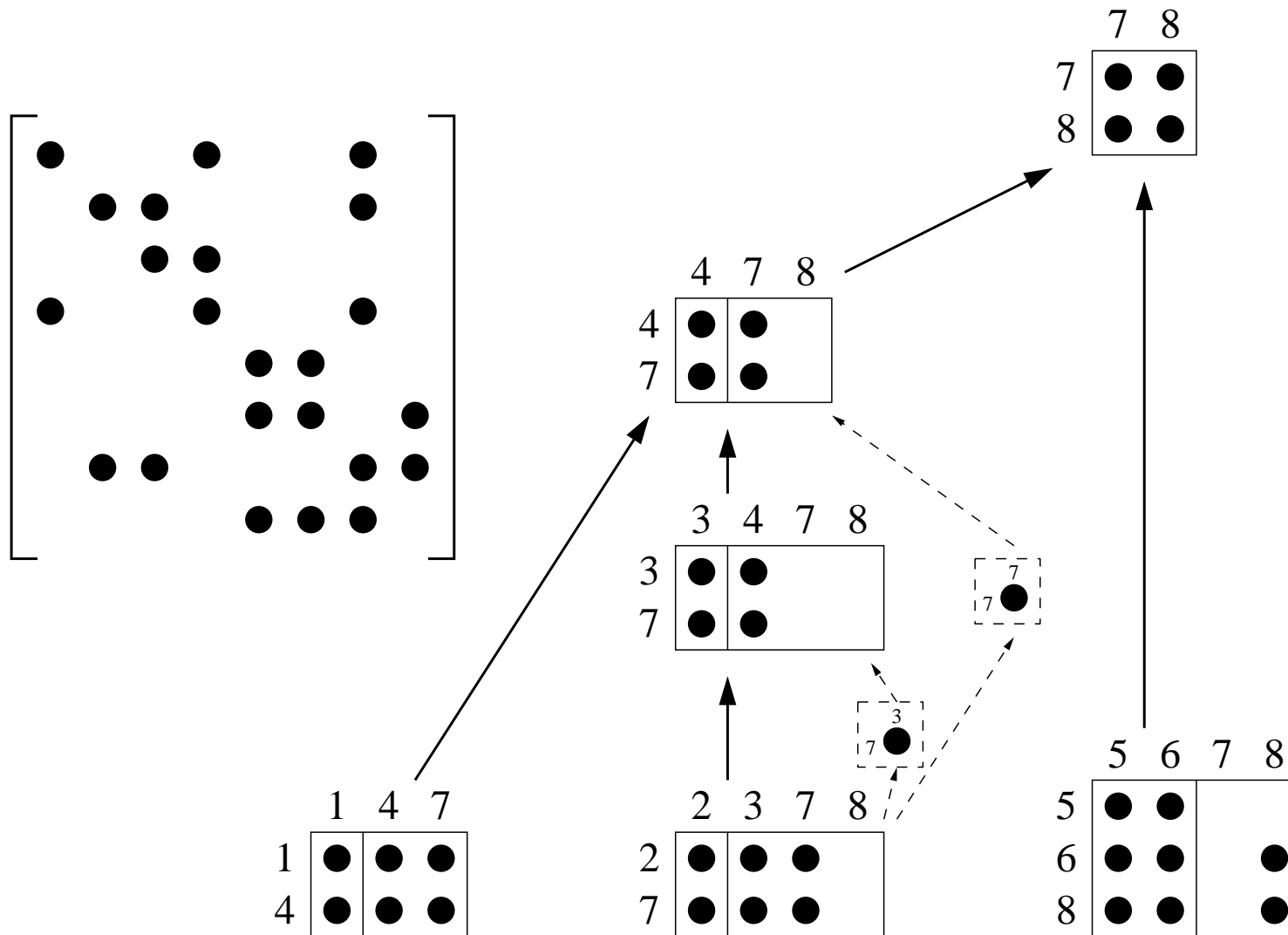
Column etree with frontal matrices



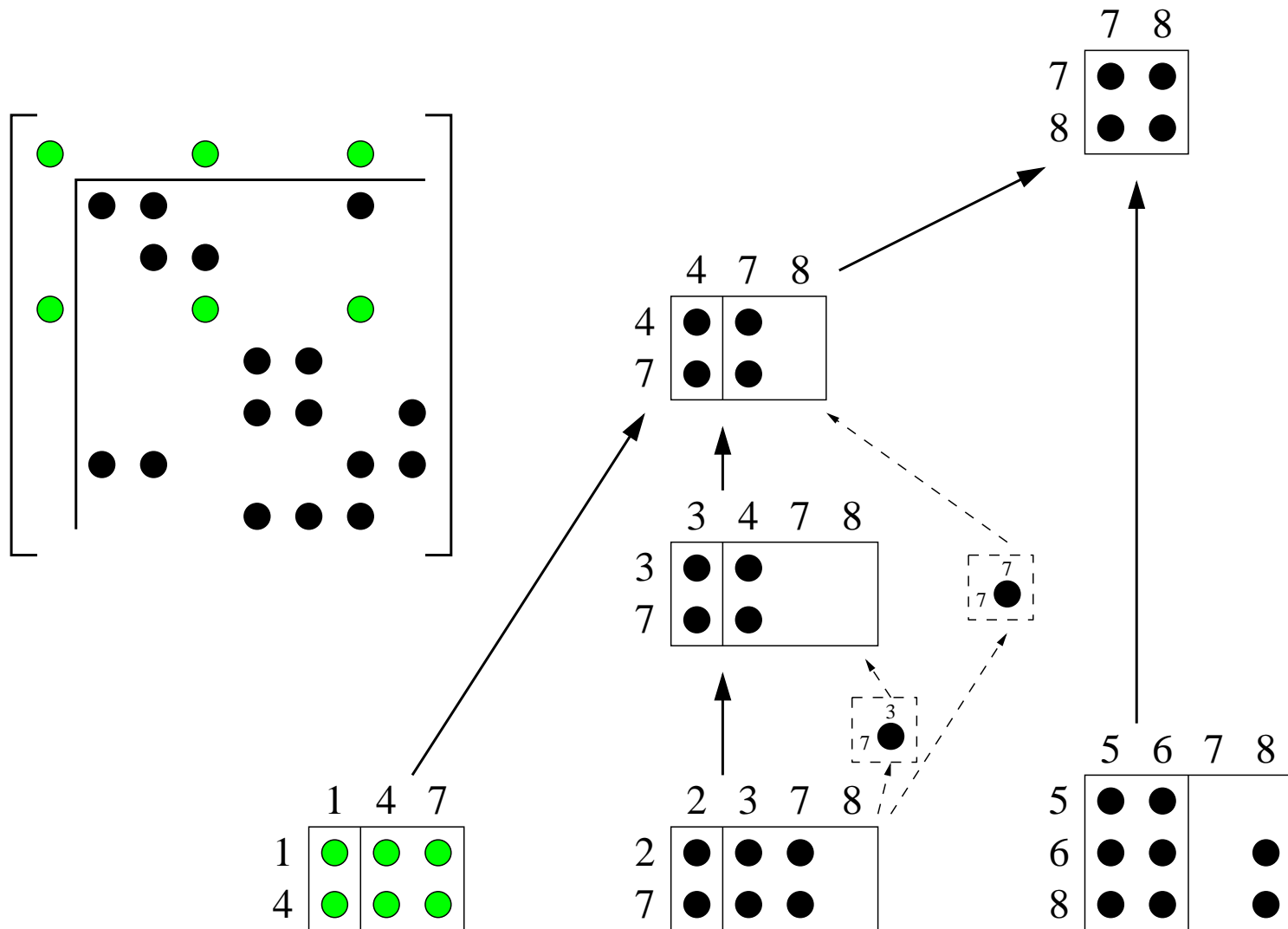
Column etree with frontal matrices



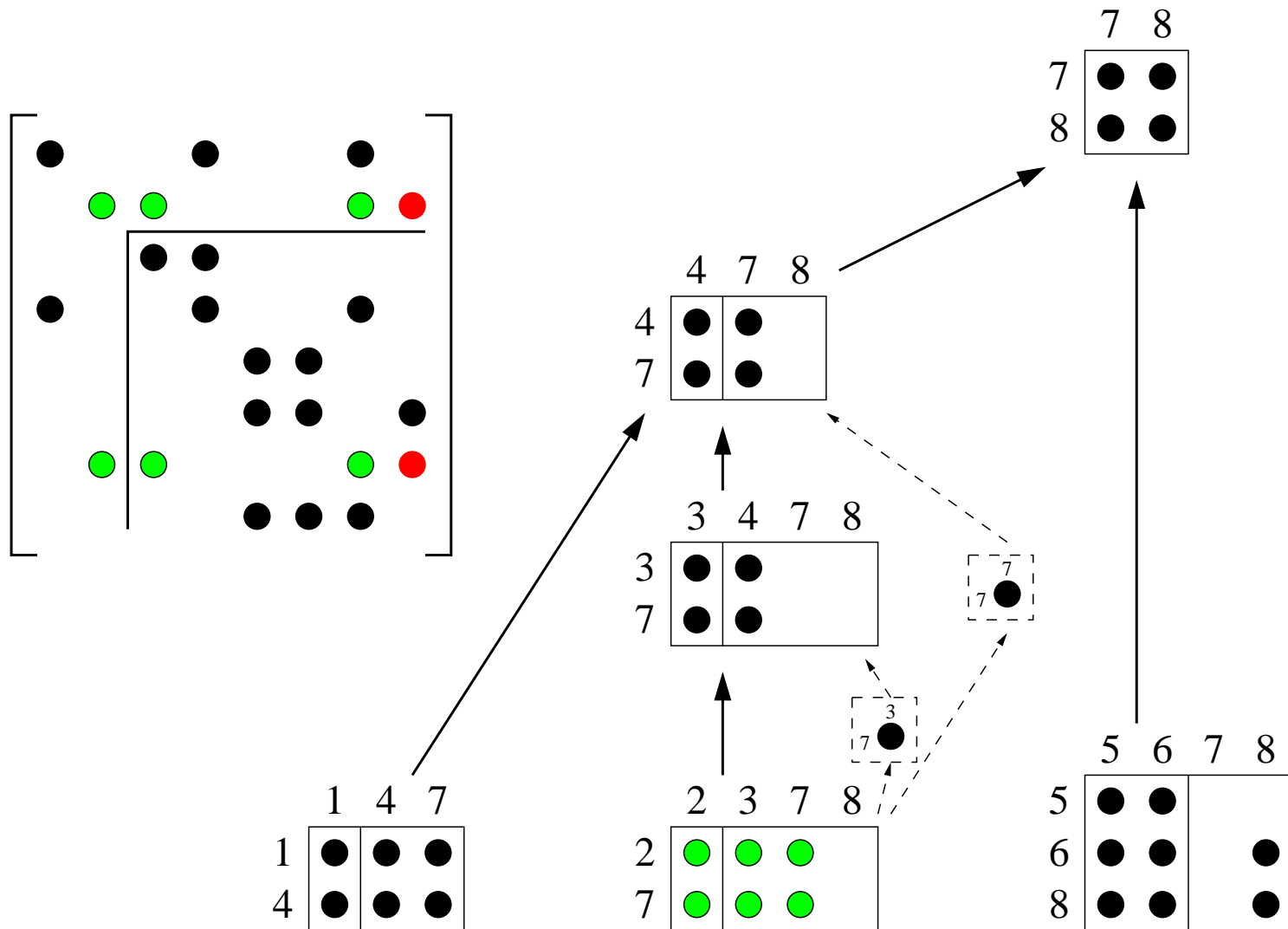
Column etree with frontal matrices



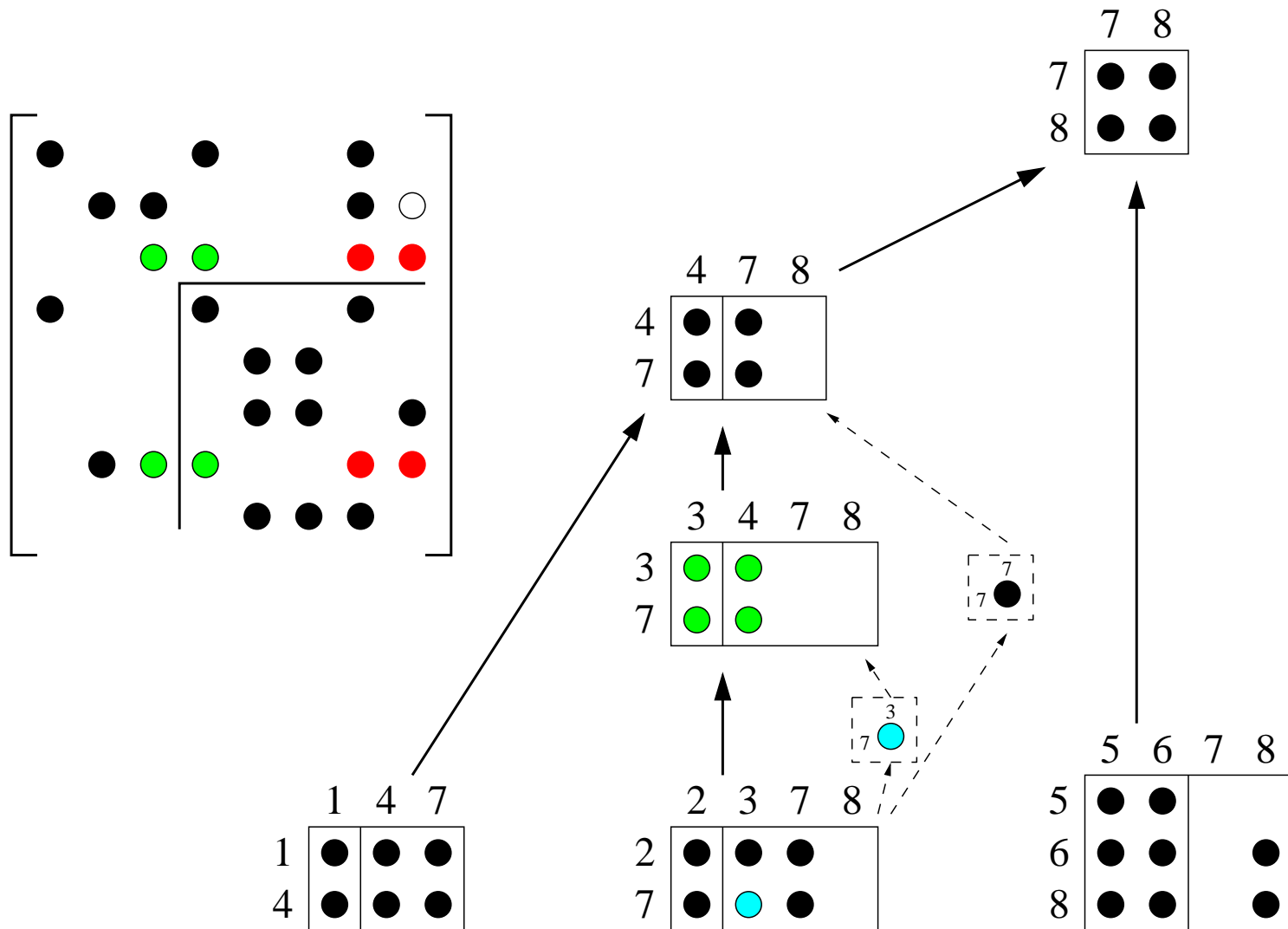
Column etree with frontal matrices



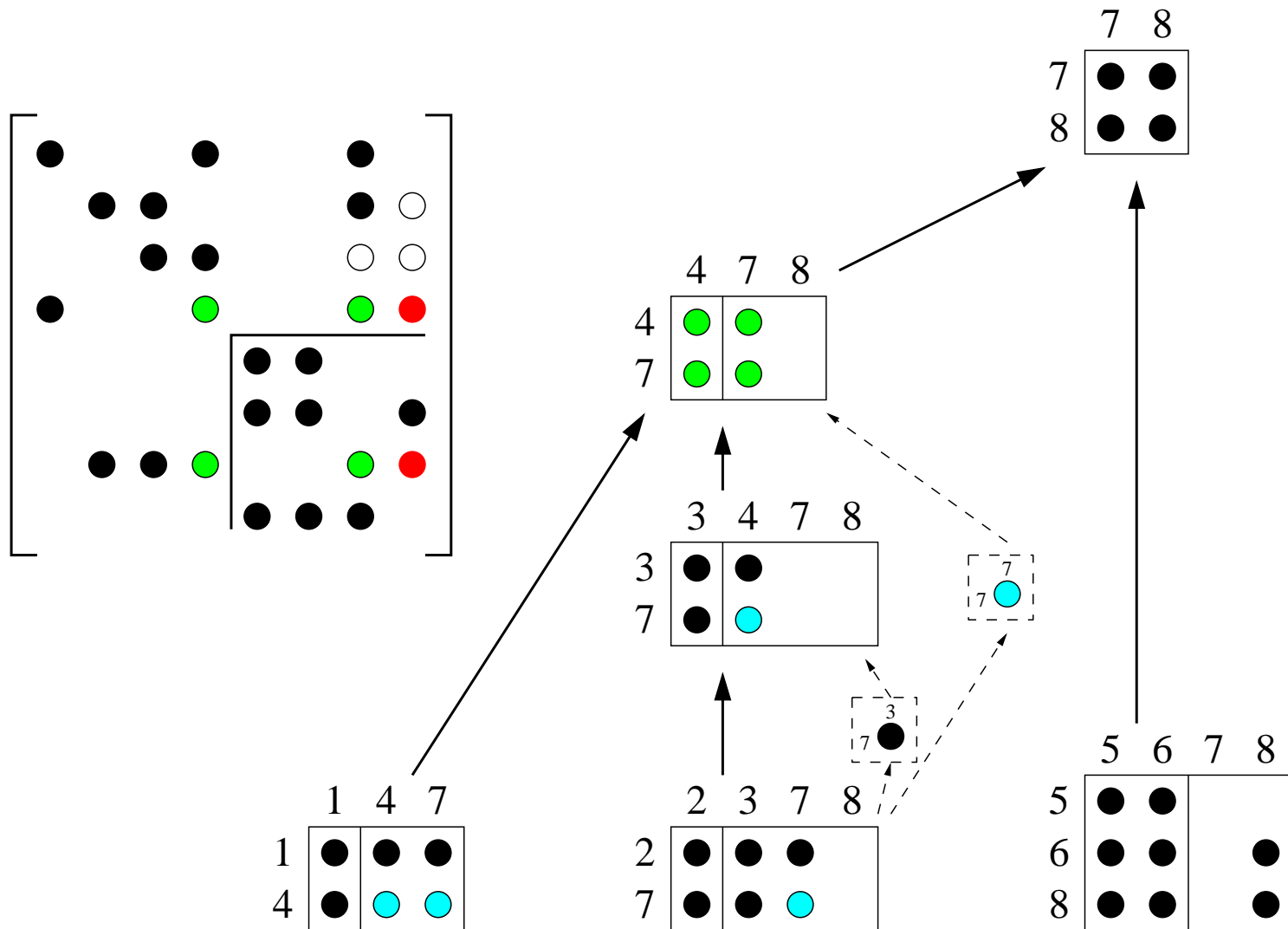
Column etree with frontal matrices



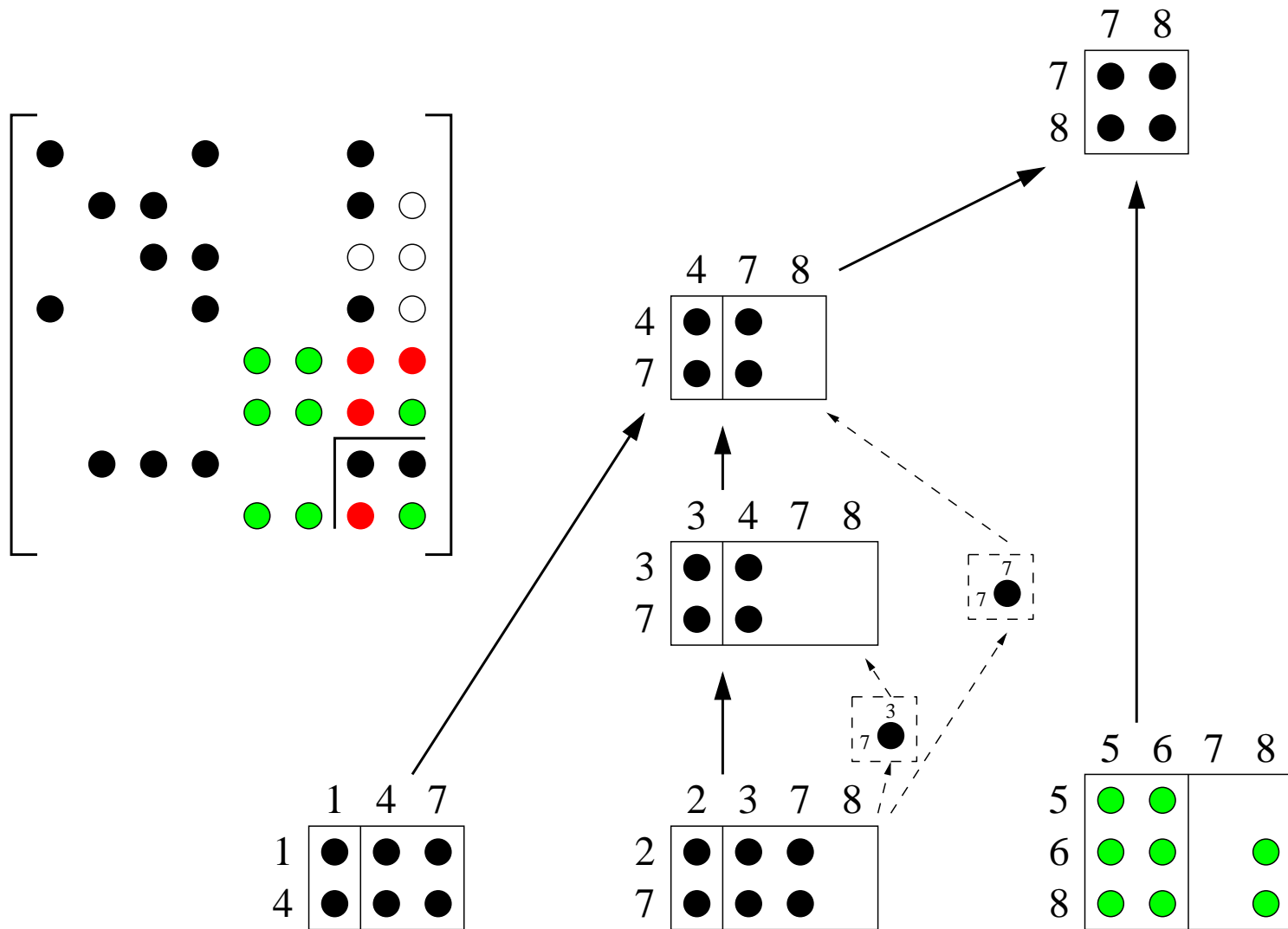
Column etree with frontal matrices



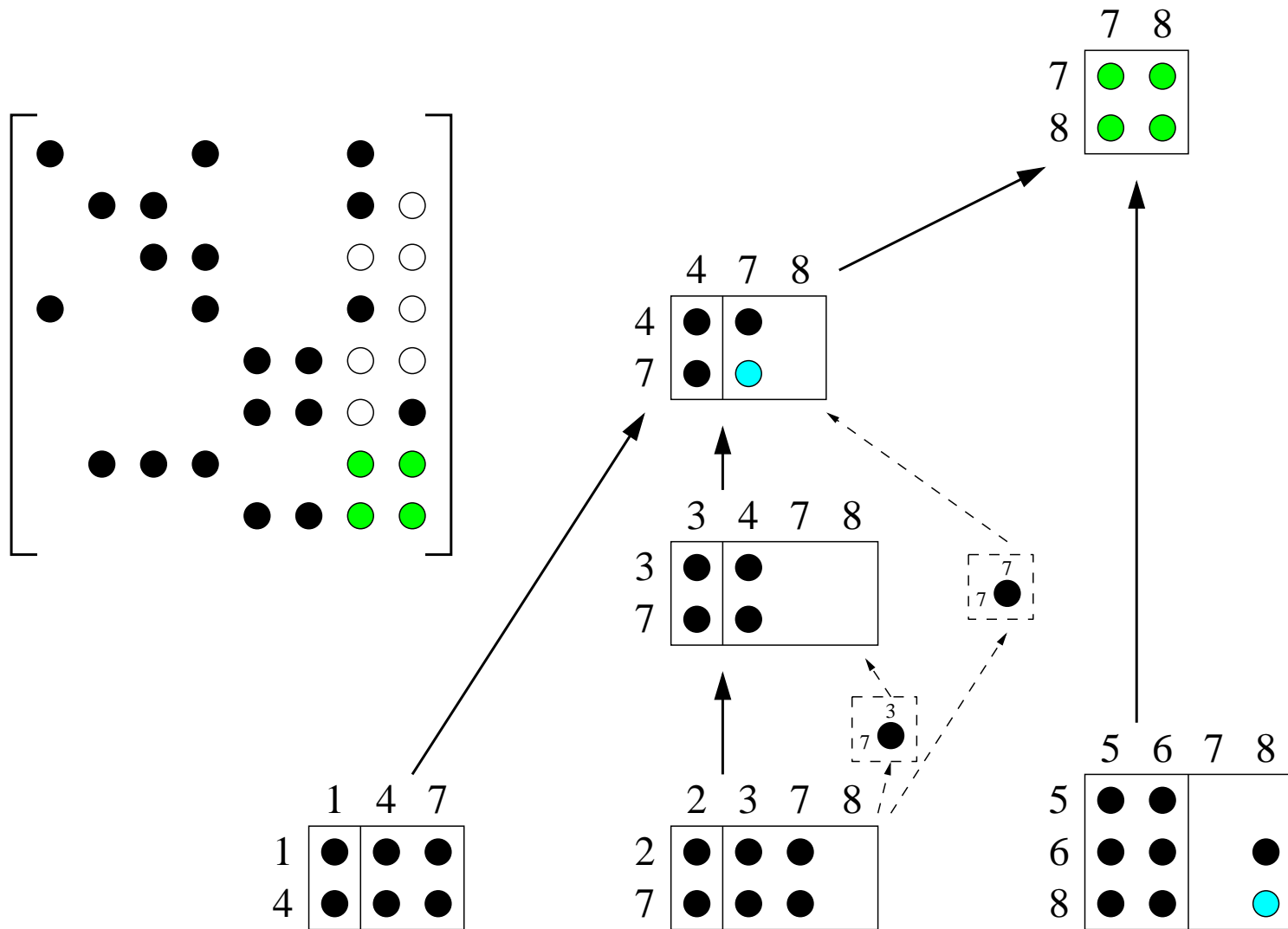
Column etree with frontal matrices



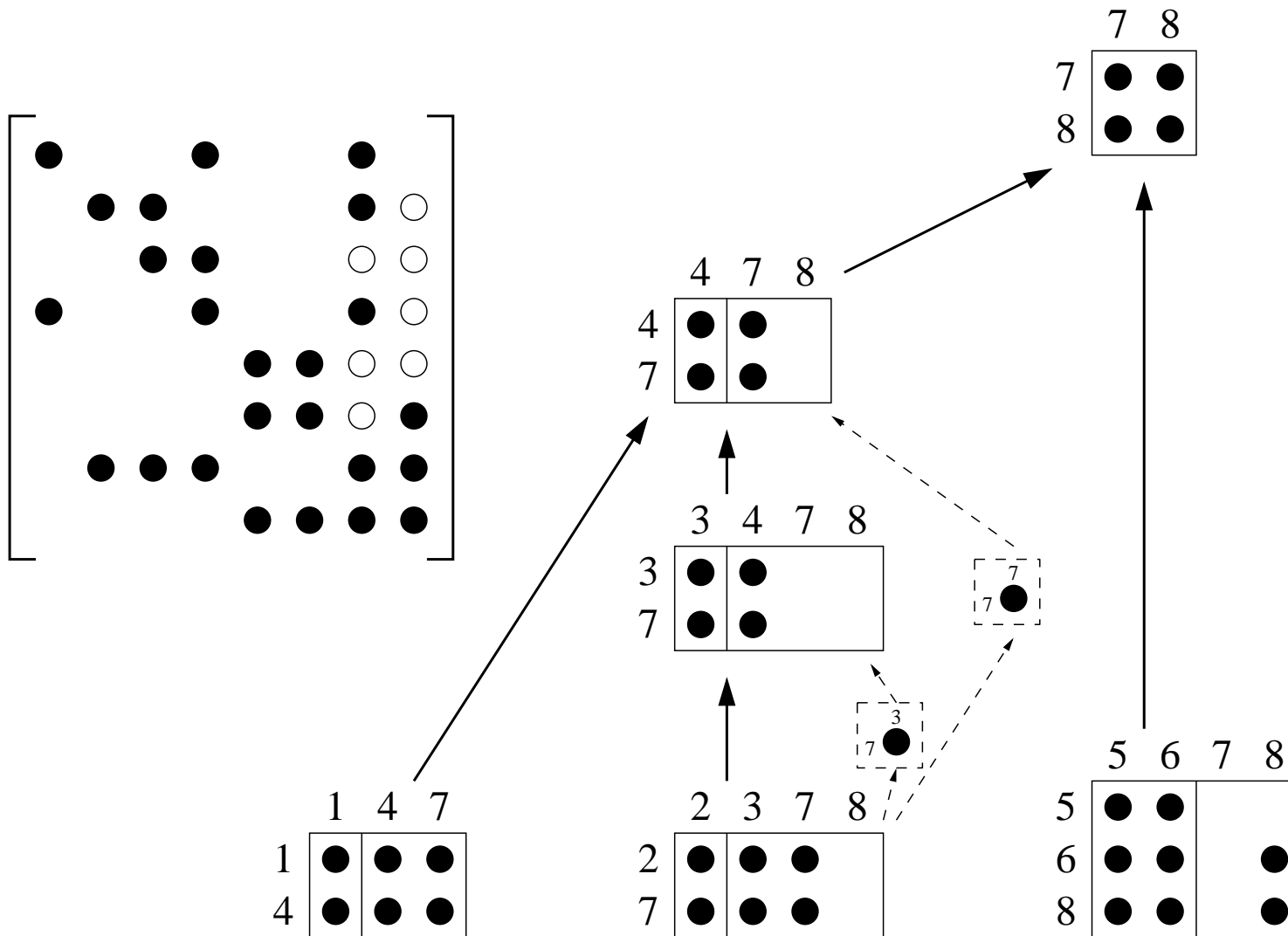
Column etree with frontal matrices



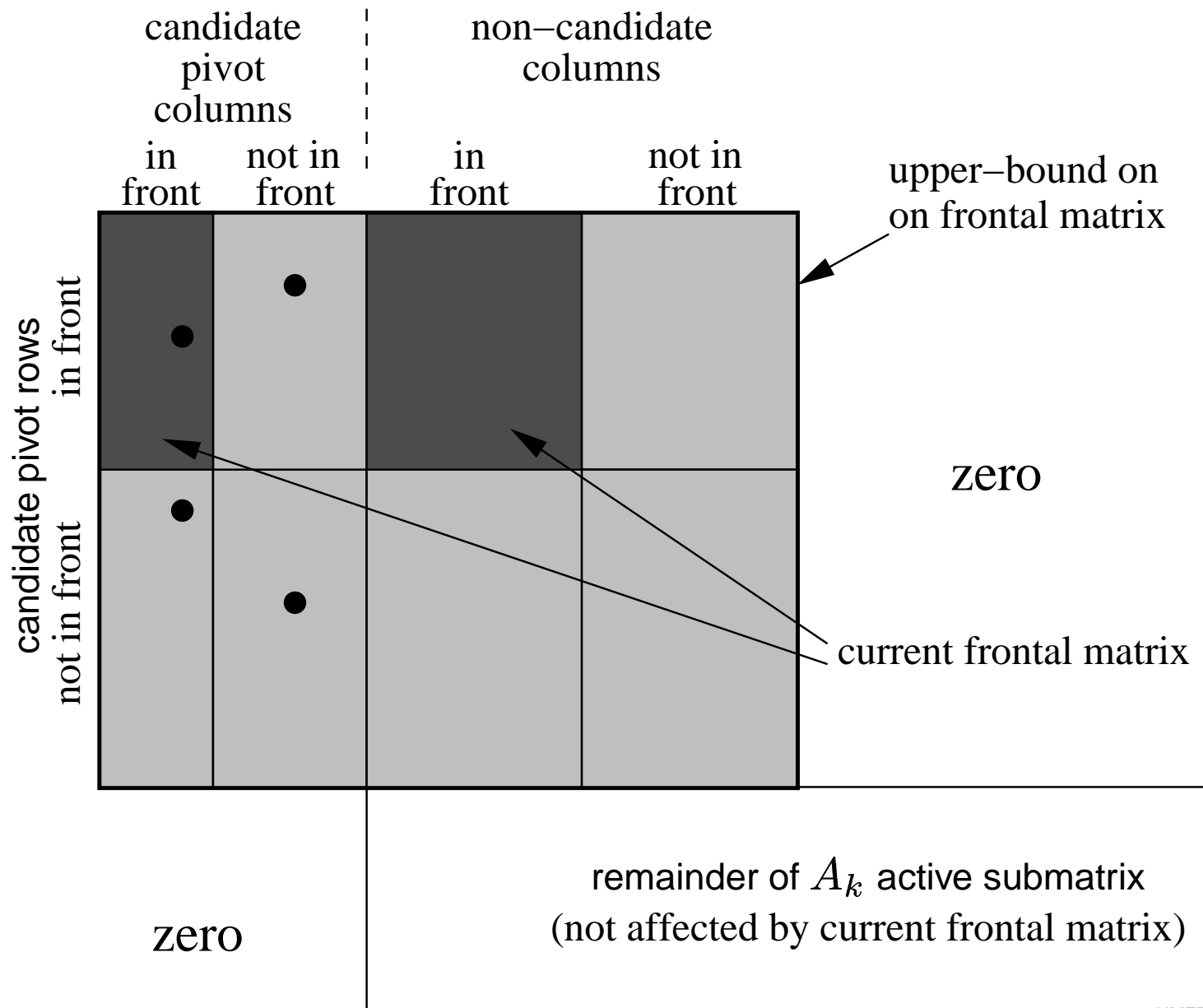
Column etree with frontal matrices



Column etree with frontal matrices

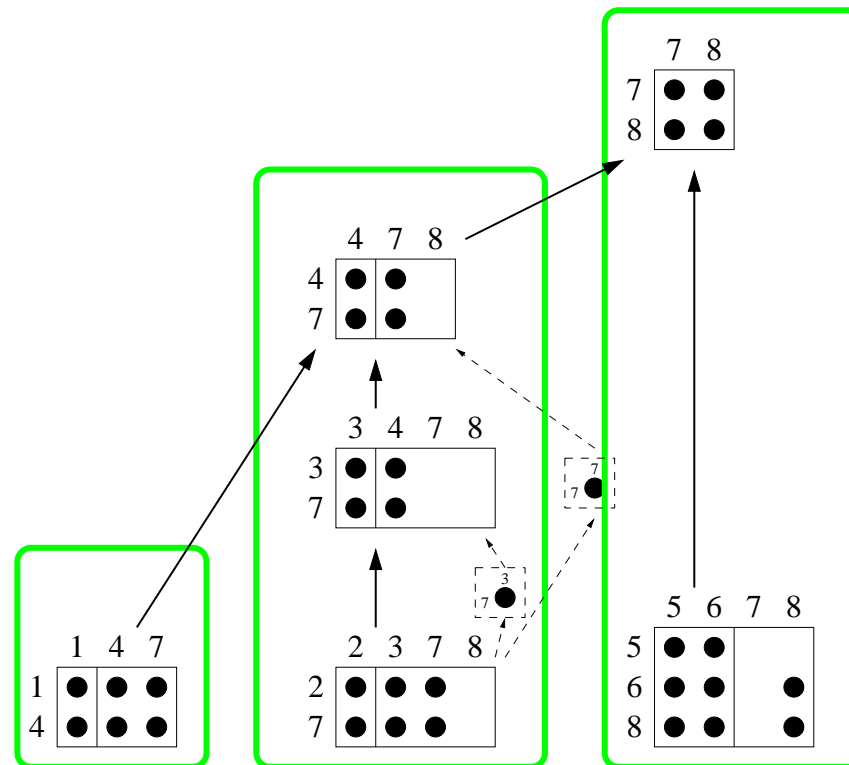


The unsymmetric frontal matrix



Unifrontal chains

- if $i + 1$ is the parent of i , then i and $i + 1$ are in the same chain.
- with a depth-first ordering, number of chains = number of leaves of column etree



UMFPACK numerical factorization

for each chain:

current frontal matrix is empty

for each frontal matrix in the chain:

for each pivot column in front

find pivot row and column

apply pending updates to pivot column

if too many zero entries in new frontal matrix (or new LU part)

apply all pending updates

if too many zero entries in new frontal matrix

create new contribution block and place on stack

start a new frontal matrix

else

extend the frontal matrix

assemble contribution blocks into current frontal matrix

scale pivot column

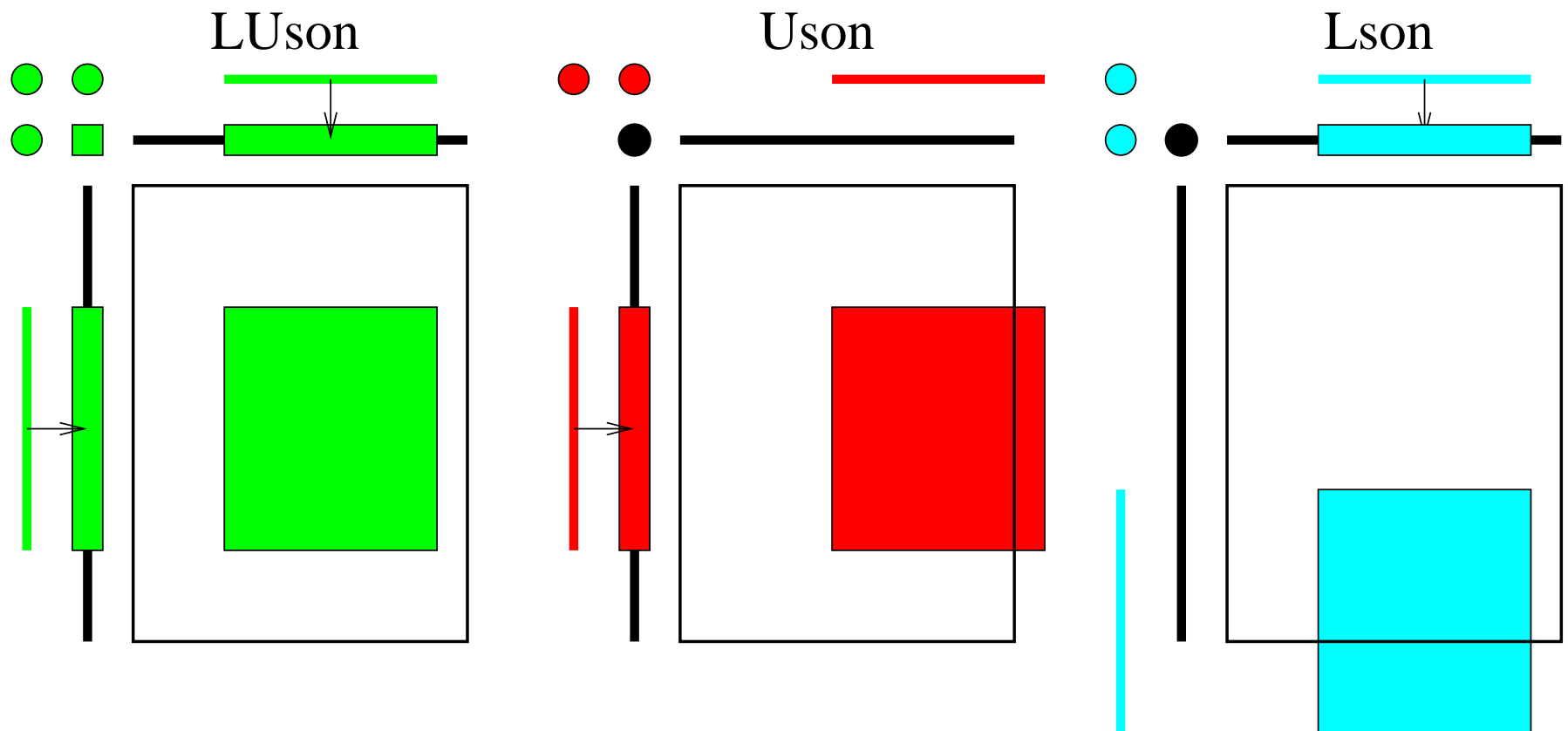
if # pivots in current frontal matrix ≥ 32

apply all pending updates

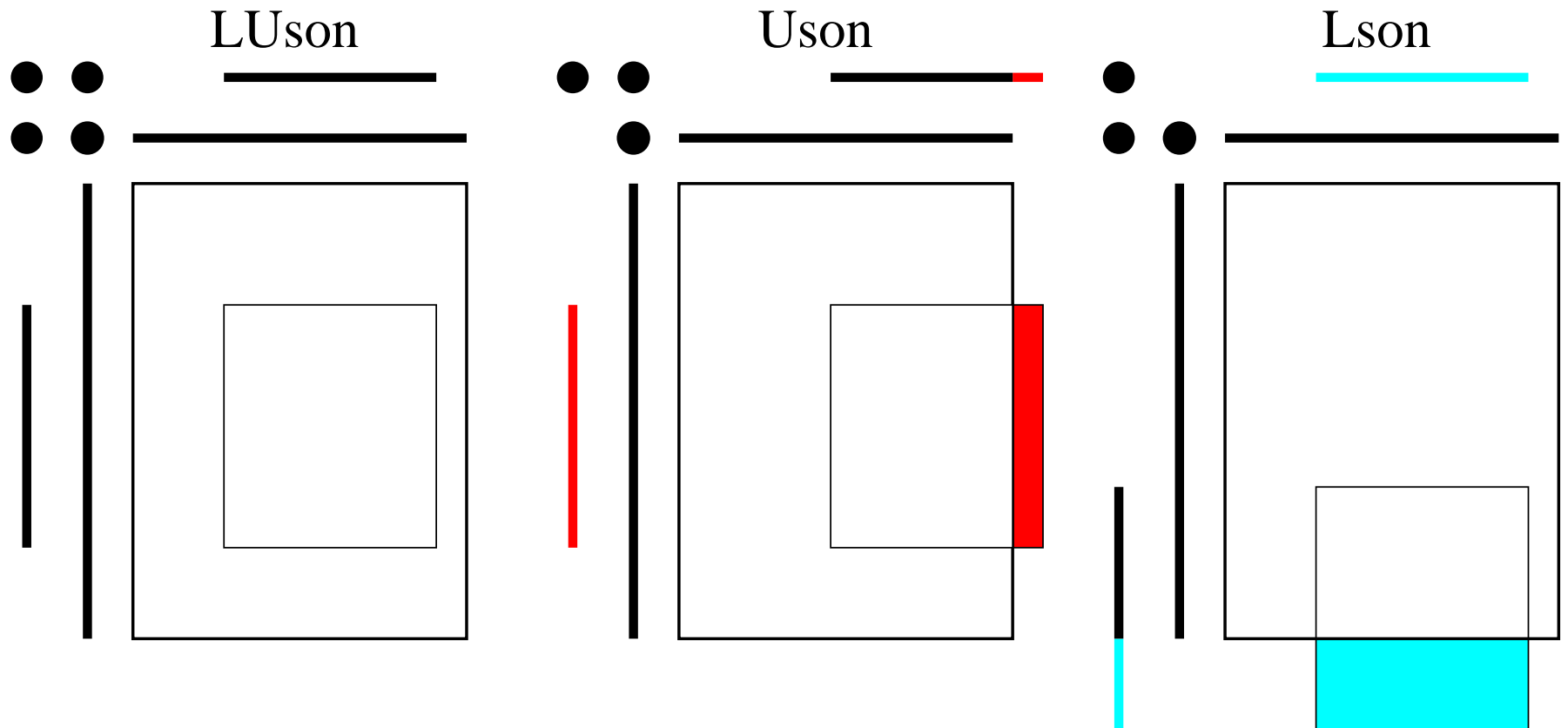
apply all pending updates

create new contribution block and place on stack

Numerical assembly / degree update



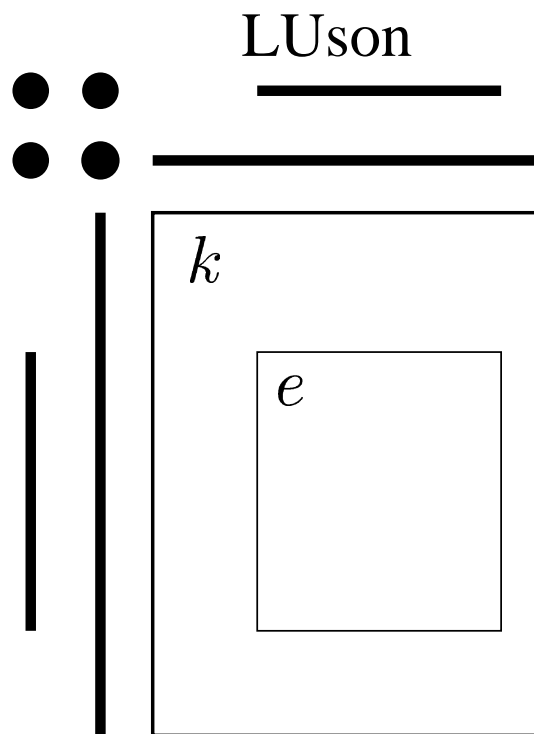
Numerical assembly / degree update



Representing the Schur complement

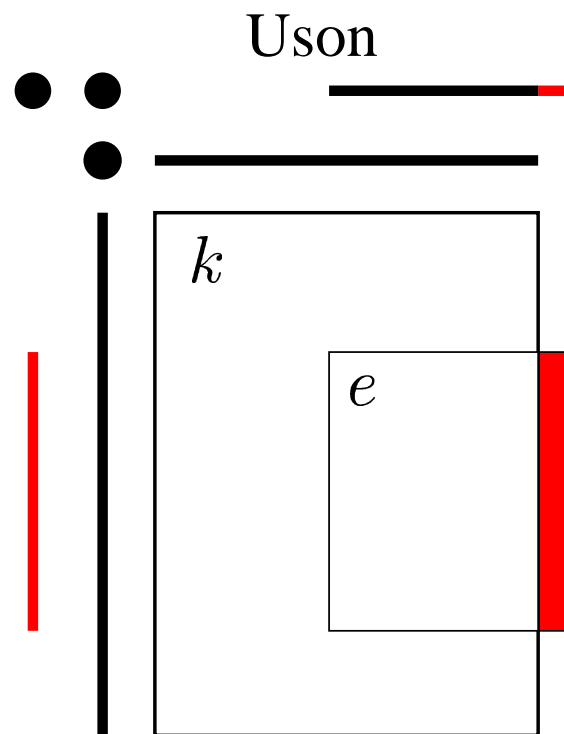
- a set of rectangular submatrices (“elements”)
- k th element:
 - \mathcal{L}_k : list of remaining row indices
 - \mathcal{U}_k : list of remaining column indices
 - C_k : a dense rectangular matrix
- row \mathcal{R}_i : list of (e, s) pairs; i is the s th entry in \mathcal{L}_e
- column \mathcal{C}_j : a list of (e, s) pairs; i is the s th entry in \mathcal{U}_e
- analogous to the symmetric quotient graph
- integer pattern not in-place, but nearly so in practice
- unsymmetric analog of approximate minimum degree
- can search for sparse pivot columns / pivot rows

Numerical assembly / degree update



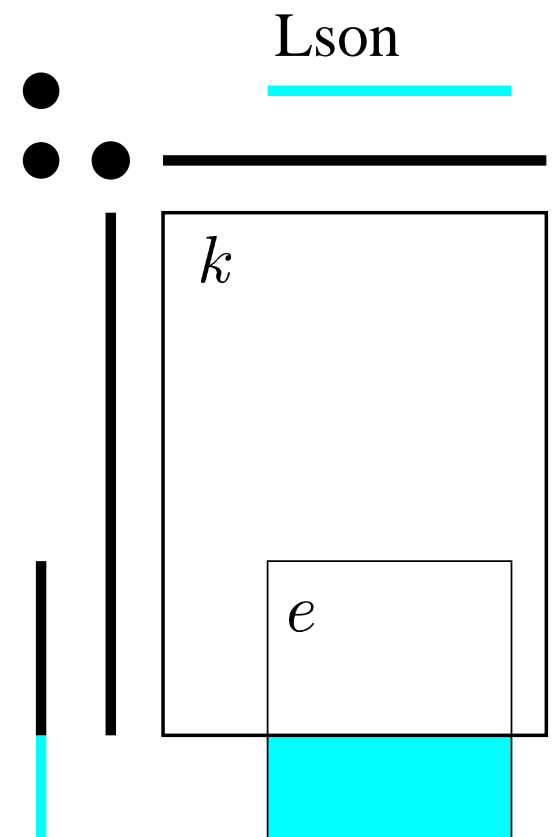
$$\mathcal{L}_e \setminus \mathcal{L}_k = \emptyset$$

$$\mathcal{U}_e \setminus \mathcal{U}_k = \emptyset$$



$$\mathcal{L}_e \setminus \mathcal{L}_k = \emptyset$$

$$\mathcal{U}_e \setminus \mathcal{U}_k \text{ not empty}$$



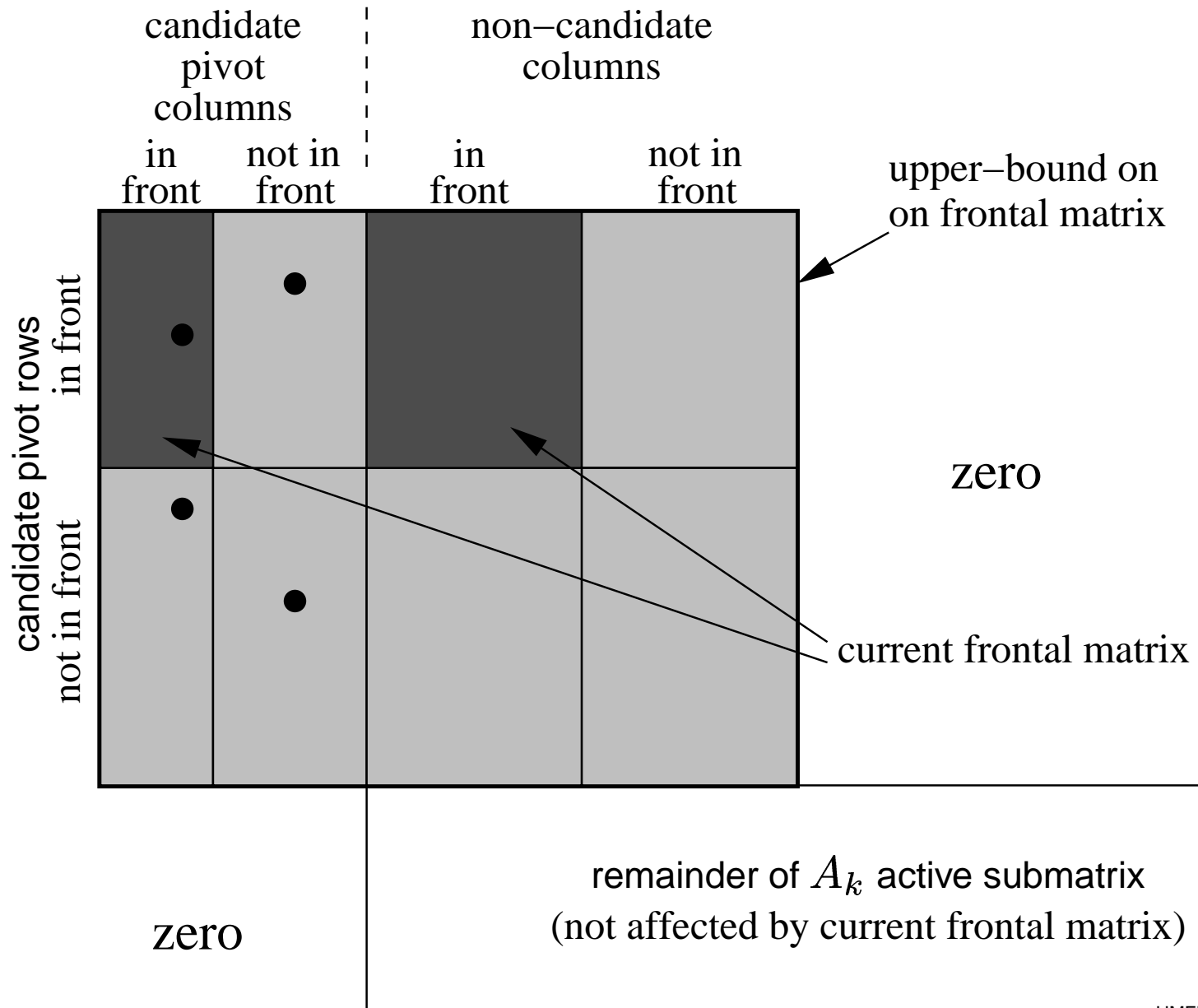
$$\mathcal{L}_e \setminus \mathcal{L}_k \text{ not empty}$$

$$\mathcal{U}_e \setminus \mathcal{U}_k = \emptyset$$

Local pivot search

- Existing left-looking methods:
 - do not construct the Schur complement
 - cannot pick a sparse pivot row
- UMFPACK:
 - keeps an unsymmetric quotient graph
 - approximate degree update / assembly phase
 - can reshuffle columns within each supercolumn
 - can pick a sparse pivot row
 - result: less fill-in than left-looking methods

Local pivot search



Preordering and analysis phase

UMFPACK analyzes the matrix A and automatically selects one of three column pre-ordering strategies:

- unsymmetric
- symmetric
- 2-by-2

Preordering and analysis phase

Unsymmetric strategy:

- for matrices with very unsymmetric nonzero pattern
- pre-ordering and analysis: column preordering via COLAMD, reduces a “pessimistic” upper bound fill-in. column etree postordering (longer unifrontal chains).
- during numerical factorization: reshuffles columns within each supercolumn. threshold partial pivoting; selects sparse pivot rows.
- only strategy in UMFPACK v4.0
(MATLAB 6.5, $x = A \setminus b$)

Preordering and analysis phase

Symmetric strategy:

- for matrices with zero-free diagonal and roughly symmetric nonzero pattern
- pre-ordering and analysis: symmetric pre-ordering via approximate minimum degree (AMD) on $\mathbf{A} + \mathbf{A}^T$, reduces an “optimistic lower-bound” on fill-in. symmetric etree postordering.
- during numerical factorization: does not reshuffle columns within each supercolumn. attempts to select diagonal entries.
- much better for finite-element matrices and circuit matrices

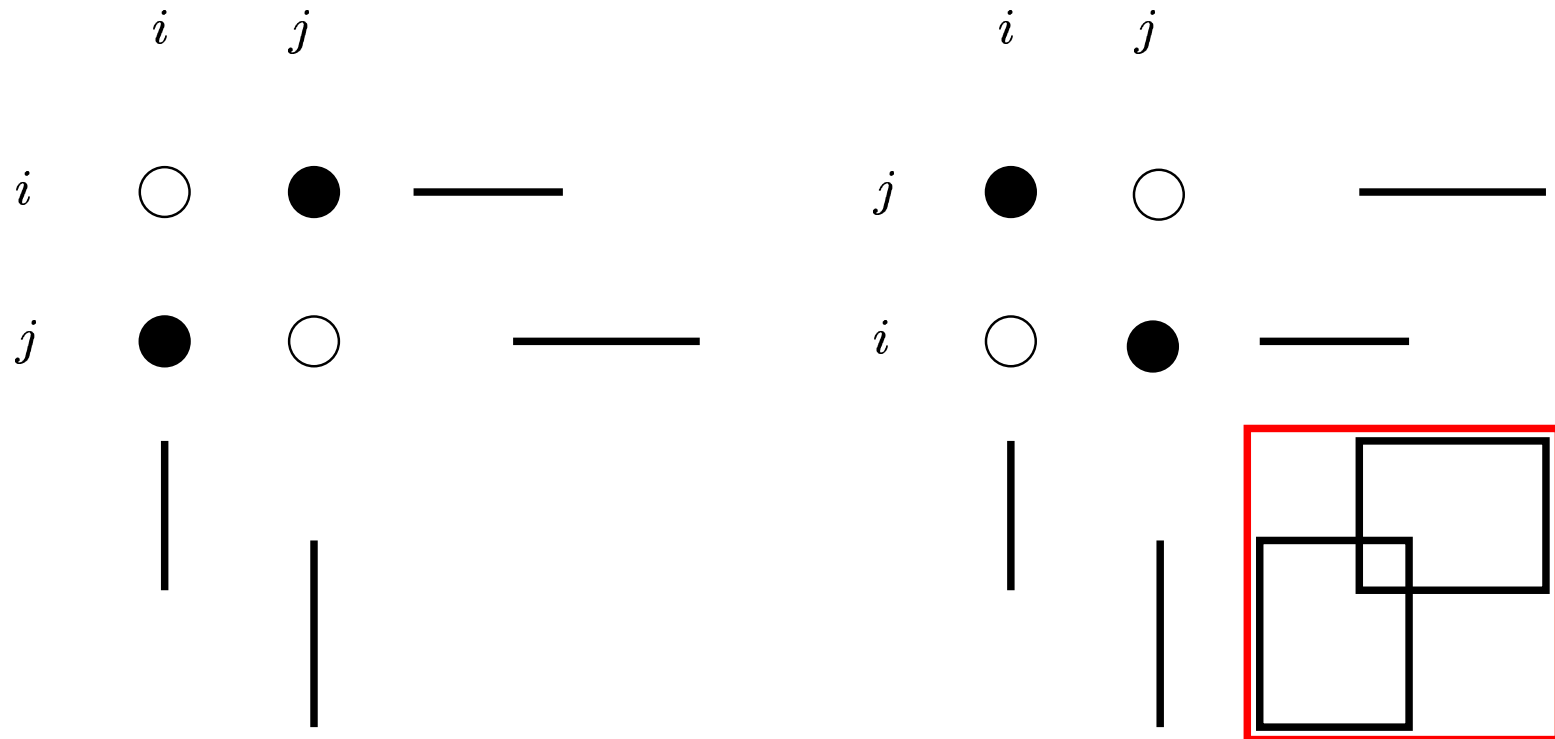
Preordering and analysis phase

2-by-2 strategy:

- permute the rows of \mathbf{A} , attempt to find a zero-free while also attempting to maintain symmetry
- if a_{ii} is small, swaps rows i and j where both a_{ij} and a_{ji} are large.
- attempts to pick i and j with similar degree.
- may be “unsuccessful” (still too many zeros on diagonal, or symmetry significantly deteriorated).
- if unsuccessful, unsymmetric strategy used instead
- otherwise, uses symmetric strategy on $\mathbf{P}_{2by2}\mathbf{A}$.
- much better for CFD fluid/structure interaction matrices

Preordering and analysis phase

2-by-2 strategy



Experimental results

- Dell Latitude C840, 2 GHz Pentium 4M, 1 GB memory
- Goto's DGEMM peak performance: 3.3 Gflops
- UMFPACK v4.1 (peak: 1.7 Gflops)
- LU (peak: 0.2 Gflops)
- SuperLU (peak: 0.7 Gflops)
- MA38 (peak: 1.2 Gflops)
- MA41u (unsymmetric multifrontal method, local symmetrization. peak: 2.0 Gflops)
- Note: SuperLU and MA41u have parallel versions; UMFPACK does not

Test matrices

● unsymmetric:

Mallya: chemical process engineering

AT&T: harmonic circuit simulation

Graham, Vavasis: irregular finite-element

PSMIGR: population migration

Simon: 2D airfoil with unsymmetric turbulence

Saad: Navier-Stokes, finite-element

● 2-by-2:

Goodwin: fluid mechanics

Averous: plate-fin heat exchanger

Norris: human heart

Bova: Charleston harbor

PSMIGR: population migration

● symmetric:

Norris: human torso, stomach

Simon: structure problems, fluid-flow, turbulence

Bai: airfoil

Zhao: electromagnetics

Wang, Ronis: crystals, semiconductors

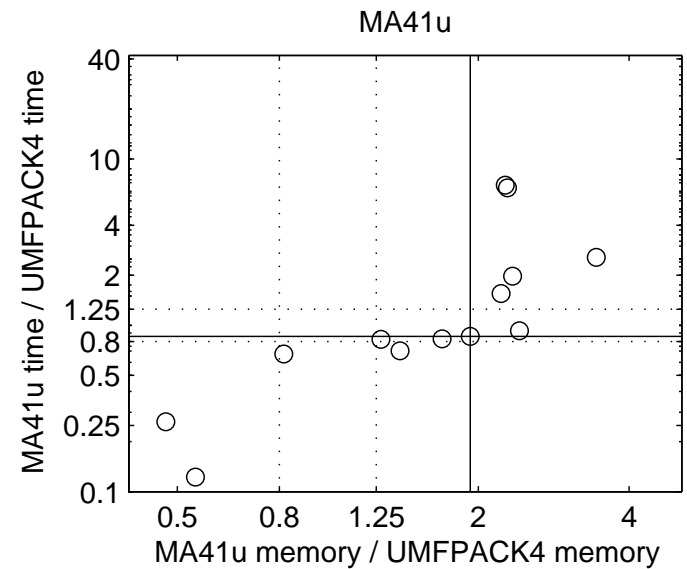
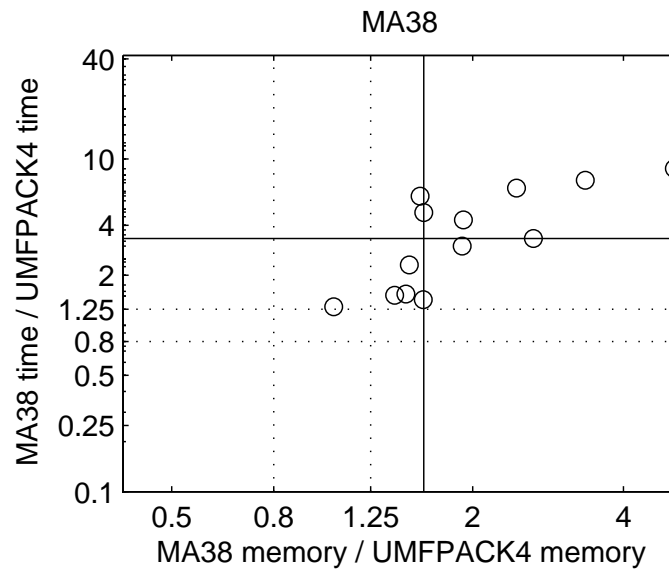
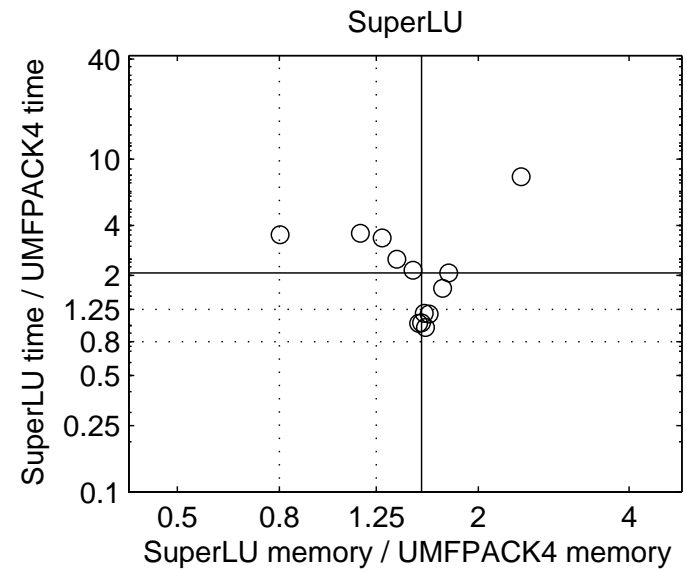
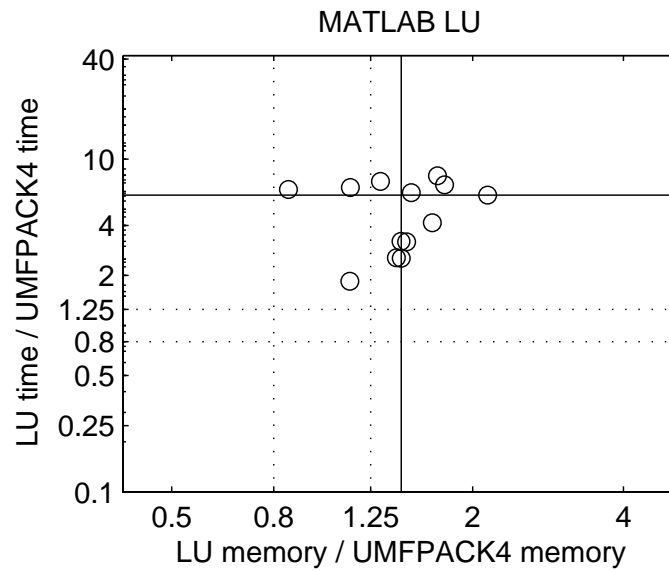
van Heukelum: DNA electrophoresis

Results for unsymmetric set

Median ratio of (LU, SuperLU, MA38, MA41u) / UMFPACK results (time, flop count, nz in LU, and memory), and median bytes/nz.

	LU	SuperLU	UMF4	MA38	MA41u
time:	6.09	2.07	1	3.33	0.86
flop:	1.37	1.45	1	2.40	1.38
nz LU:	1.19	1.23	1	1.36	1.16
mem:	1.44	1.54	1	1.60	1.93
mem/nz:	12.43	12.66	10.46	12.76	19.87

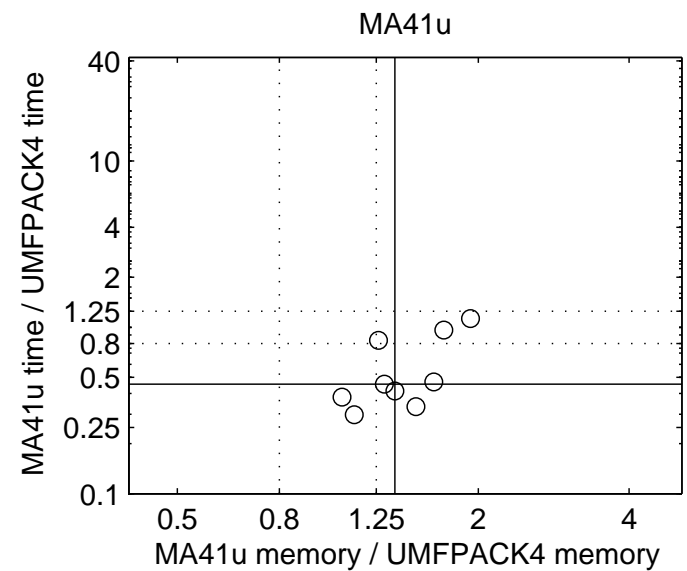
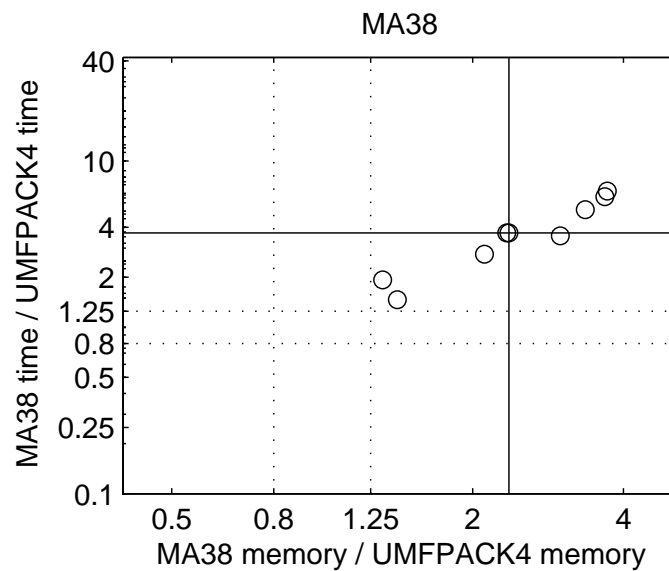
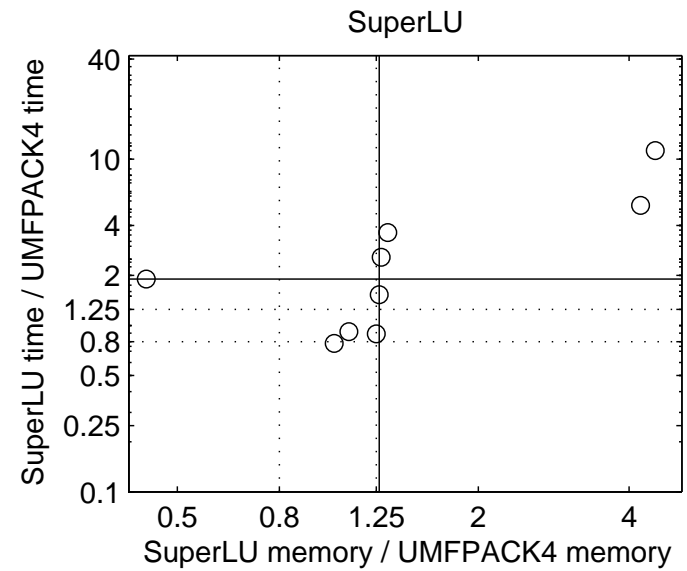
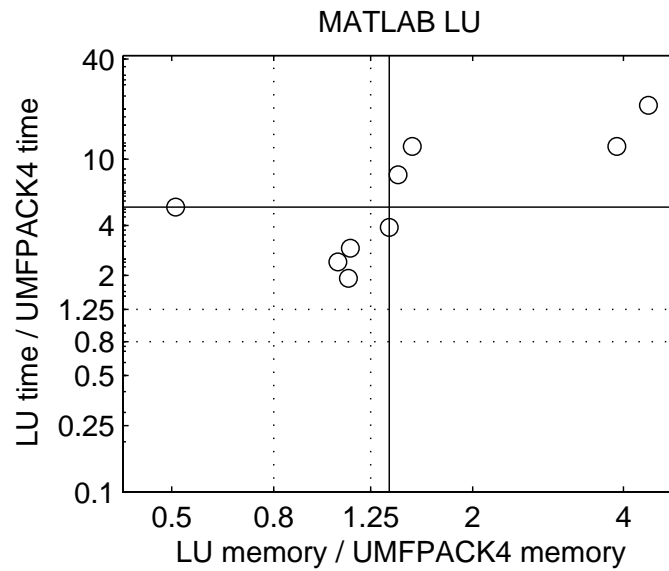
Results for unsymmetric set



Results for 2-by-2 set

	LU	SuperLU	UMF4	MA38	MA41u
time:	5.15	1.90	1	3.70	0.45
flop:	1.08	1.14	1	5.52	1.00
nz LU:	1.04	1.07	1	2.35	1.01
mem:	1.36	1.27	1	2.36	1.36
mem/nz:	12.09	11.05	11.55	12.36	15.62

Results for 2-by-2 set

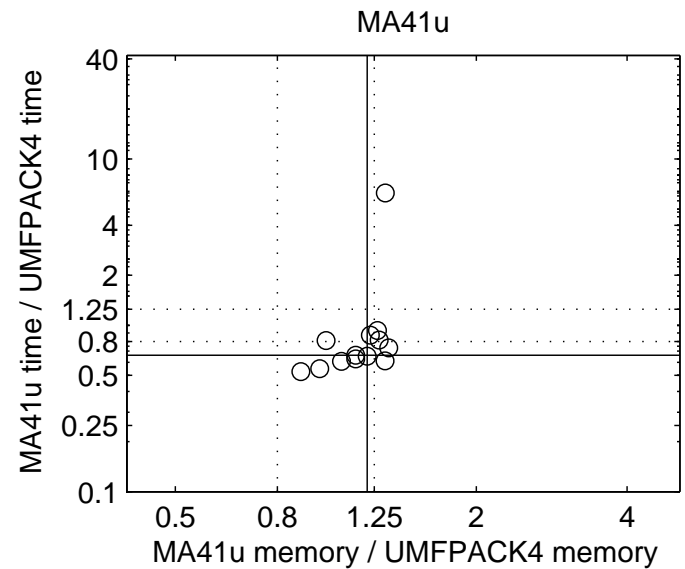
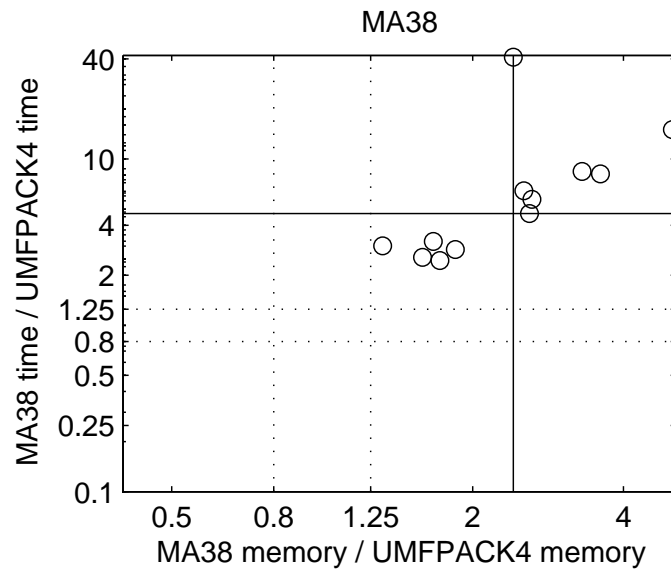
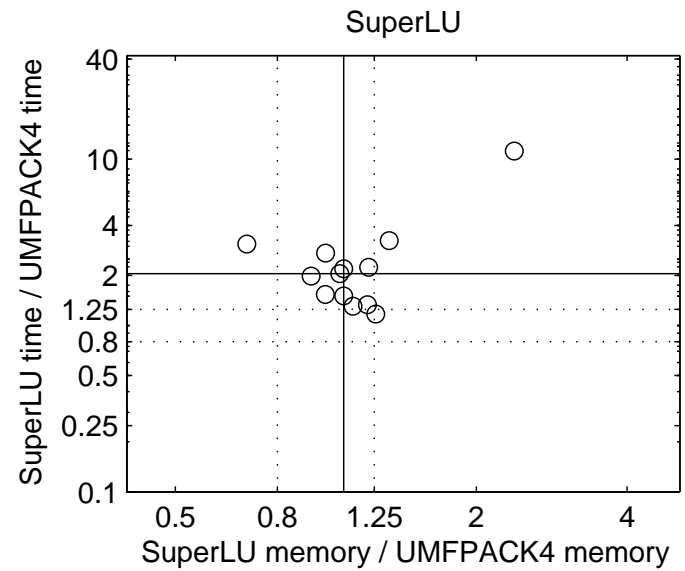
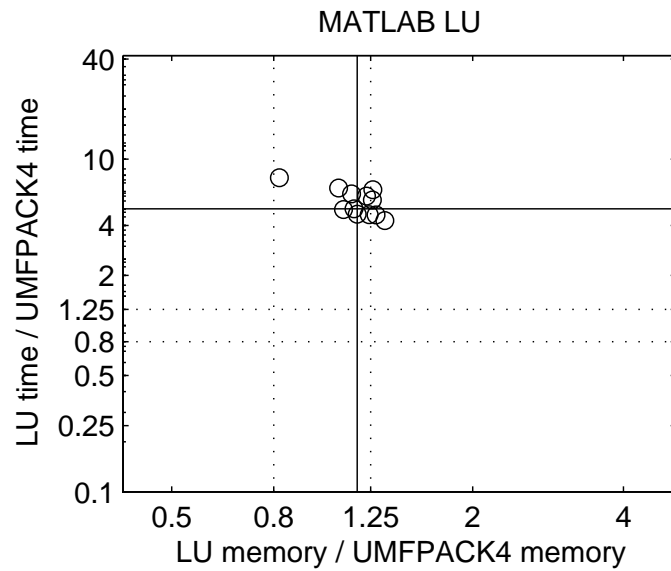


Results for symmetric set

	LU	SuperLU	UMF4	MA38	MA41u
time:	5.02	2.05	1	4.71	0.66
flop:	1.00	1.00	1	3.78	1.00
nz LU:	1.00	1.00	1	2.17	1.01
mem:	1.17	1.09	1	2.41	1.21
mem/nz:	12.13	10.46	10.19	11.88	11.86

TAUCS (multifrontal Cholesky): half the work, “20% to 50% faster than UMFPACK v4.1.”

Results for symmetric set



Summary

- column pre-ordering + refinement during numerical factorization = better results for unsymmetric matrices
- practical meta-algorithm (automatic strategy selection):
 - unsymmetric: $A^T A$ -based, “pessimistic” upper bound + refinement
 - symmetric: diagonal pivoting, “optimistic” ordering
 - 2-by-2: cheap matching + symmetric strategy
 - worst case automatic selection: run time 30% higher, memory 20% higher than best strategy
- competitive performance (not parallel yet, however)
- v4.0 in MATLAB 6.5 (v4.1 faster)
- MATHEMATICA, NASTRAN, FEMLAB, ARPACK, TRILINOS, ...