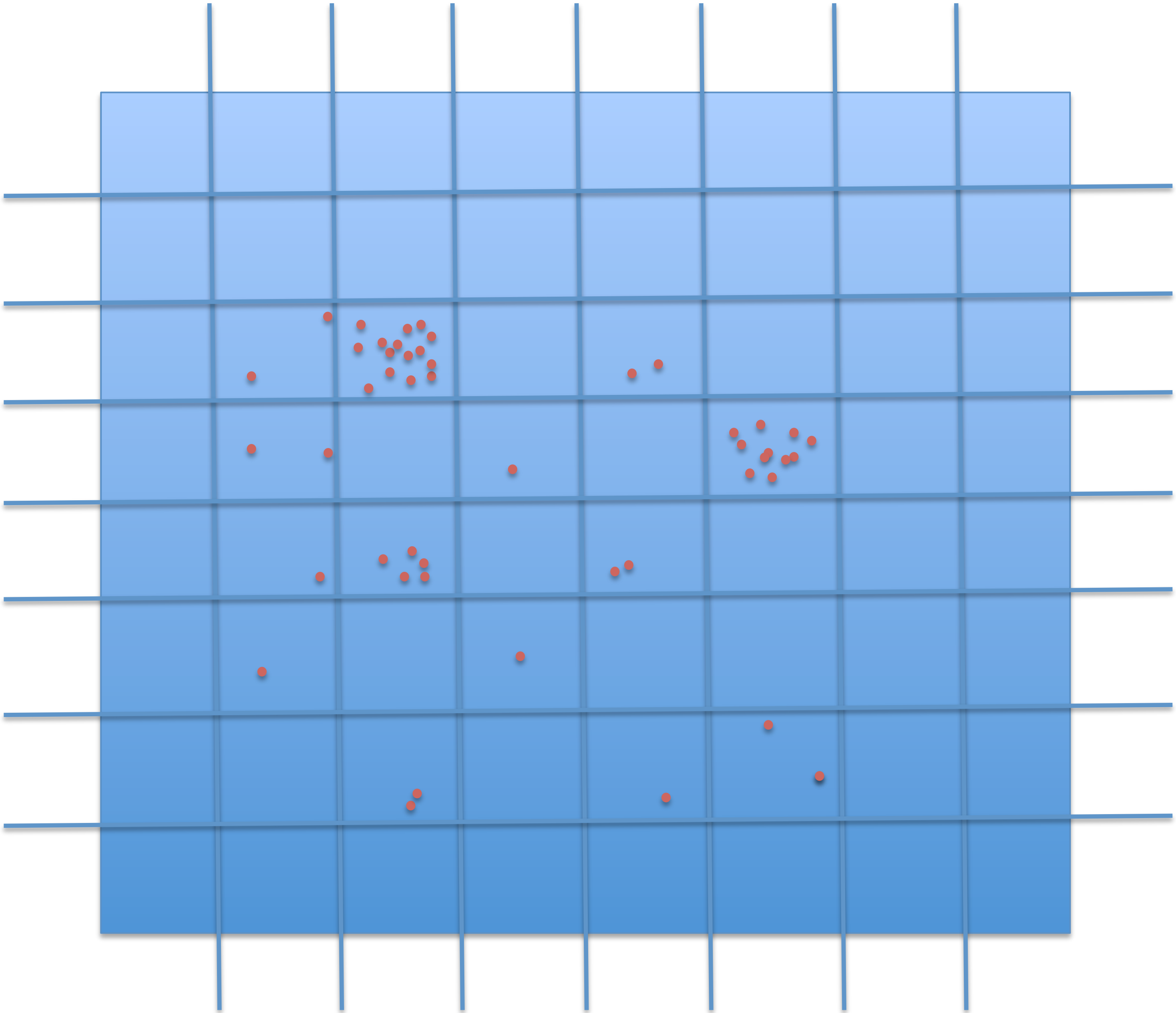
Parallel Sparse Matrix Computations

Parallel Sparse BLAS 2 Matrix Multiplication

Like dense matrix multiplications, sparse matrix time vector multiplication can be blocked:

```
DOALL II = 1, M1
    DOALL JJ = 1, M2
        DO I = II, II + N/M1 - 1
            DO J = JJ, JJ + N/M2 - 1
                C(I) = C(I) + A(I,J) * B(J)
            ENDDO
        ENDDO
    ENDDO
ENDDO
```

However, this can lead to uneven load balance!!!!!!!



This can (partly) be prevented by only row slicing/partitioning:

```
DOALL II = 1, M1
    DO I = II, II + N/M1 – 1
        DO J = 1, N
            C(I) = C(I) + A(I,J) * B(J)
        ENDDO
    ENDDO
ENDDO
```

Mostly the number of NNZ per row/column is rather constant.

Each processor needs a full copy of the B vector!!

Column slicing/partitioning:

```
DOALL JJ = 1, M1
    DO J = JJ, JJ + N/M1 – 1
        DO I = 1, N
            C(I) = C(I) + A(I,J) * B(J)
        ENDDO
    ENDDO
ENDDO
```

Each processor just has a part of the B vector.

But every processor needs a full copy of the C vector plus the processors need to communicate their changes to C!!

Solution:

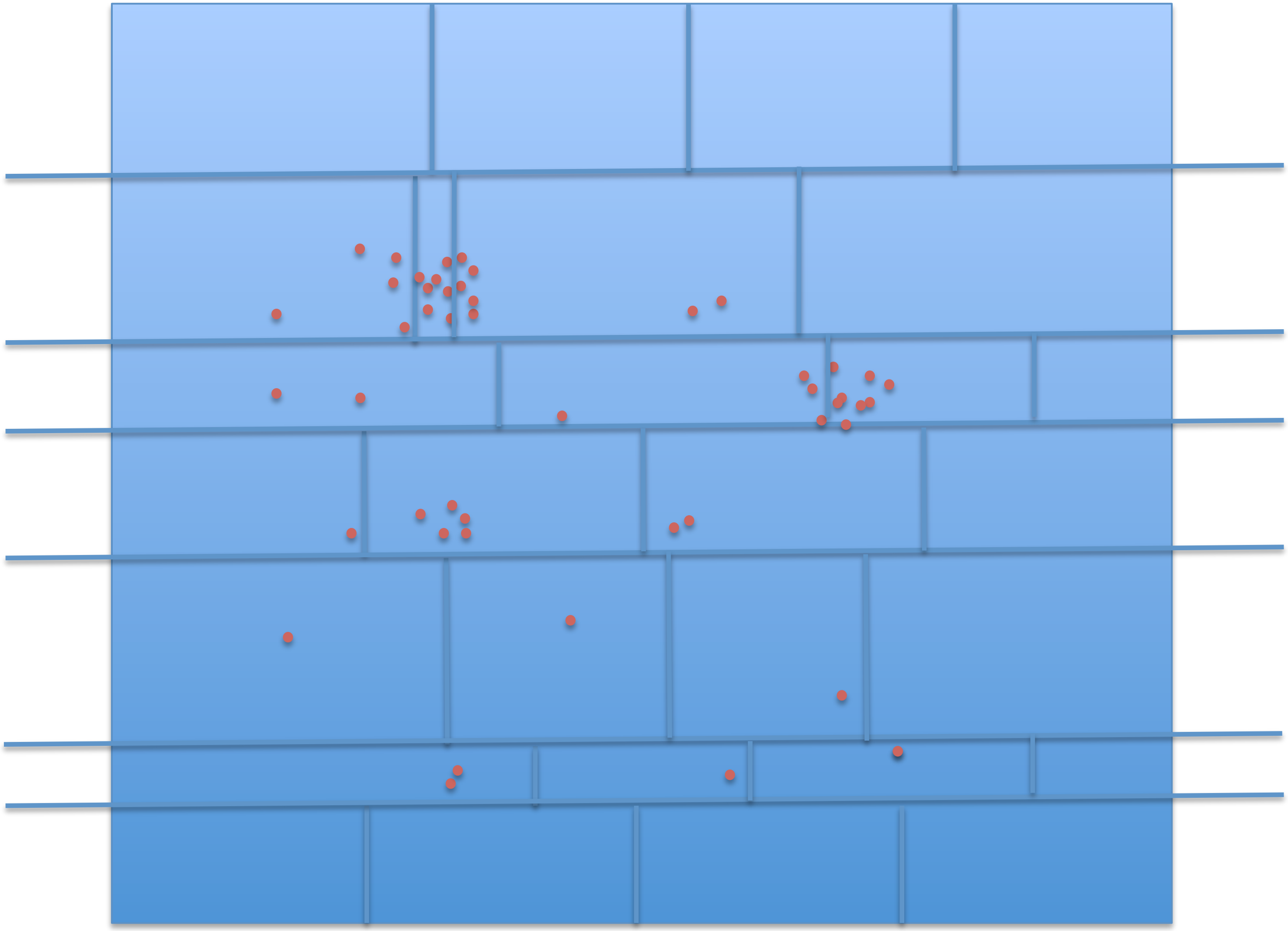
Let NNZ be the number of non-zero elements of the sparse matrix. Assume we want to compute in parallel on PxQ processors.

➔ Divide the rows into P partitions: $R_1 R_2 \ldots R_{P-1} R_P$ such that for all k: NNZ $(R_k) \approx$ NNZ/P, then partition every row partition R_k into Q partitions: $C^k_1 C^k_2 \ldots C^k_{Q-1} C^k_Q$ columns, such that for every m: NNZ $(C^k_m) \approx$ NNZ (R_k) / Q.

By doing so, we have for all k, m:

$$\text{NNZ}\ (C^k_m) \approx \text{NNZ} / PQ$$

In a picture:

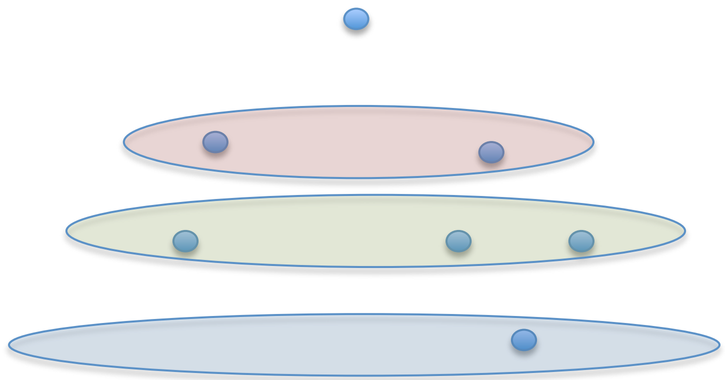


Parallel Sparse (Upper) Triangular Solver

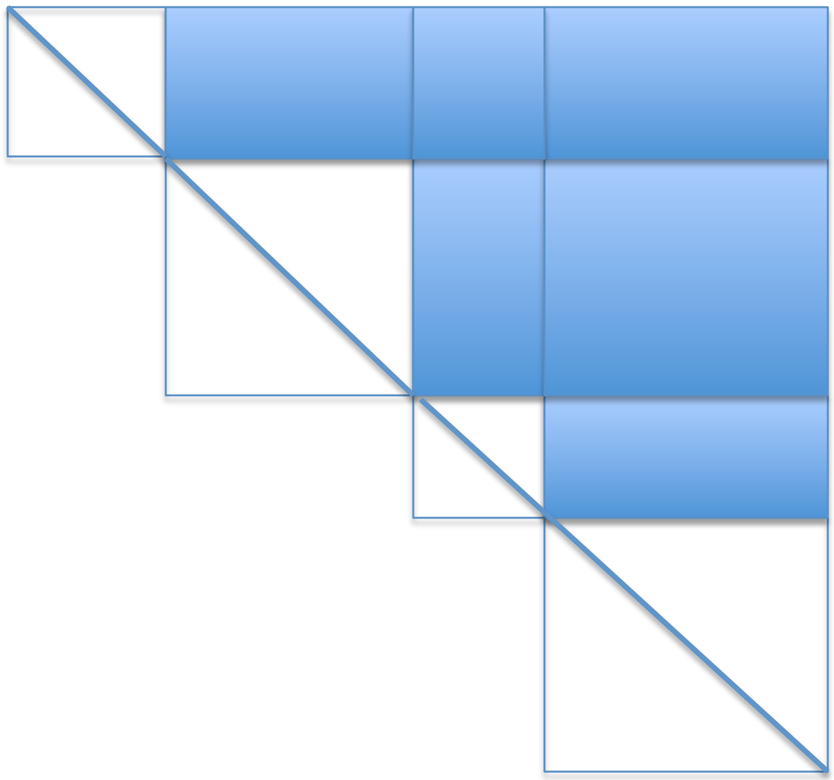
$$Ux = c$$

Levelization:

Take a DFS spanning tree of the associated symmetric graph of $U+U^T$, and group all nodes at the same level of the tree together



- ➔ the nodes within each group are not connected, i.e. will not have an edge in common
- ➔ some nnz's might be introduced in the lower triangle, which will be corrected by simple permutations
- ➔ in other words each group will form a diagonal, diagonal block
- ➔ in fact the associated digraph of a triangular matrix can be seen as a "partially ordered" set (poset) and a diagonal block as an incomparable subset of elements



So, not only do we have easily invertible U_{kk} blocks, this operation can be executed in parallel or as a vector operation.

In fact not only do we have parallelism on a block level but also on column/row level

$$\begin{bmatrix} U_1 & & \tilde{U}_1 & \\ & U_2 & & \tilde{U}_2 \\ & & U_3 & \tilde{U}_3 \\ \emptyset & & & U_4 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} = \begin{bmatrix} c_1 \\ c_2 \\ c_3 \\ c_4 \end{bmatrix}$$

1. Solve $U_4 x_4 = c_4$
2. $c_3 = c_3 - \tilde{U}_3 \cdot x_4$
3. Solve $U_3 x_3 = c_3$
4. $c_2 = c_2 - \tilde{U}_2 \cdot \begin{bmatrix} x_3 \\ x_4 \end{bmatrix}$
5. Solve $U_2 x_2 = c_2$
6. $c_1 = c_1 - \tilde{U}_1 \cdot \begin{bmatrix} x_2 \\ x_3 \\ x_4 \end{bmatrix}$
7. Solve $U_1 x_1 = c_1$

Orderings to Special Form

An ordering of a sparse matrix A to a sparse matrix B is called asymmetric if

$$B = PAQ^T,$$

with P and Q permutation matrices

If $P = Q$, then the ordering is symmetric.

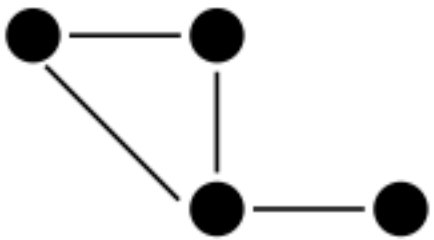
Note that the minimum degree ordering is a symmetric ordering. Also the levelization ordering is symmetric. Partial Pivoting is asymmetric!!

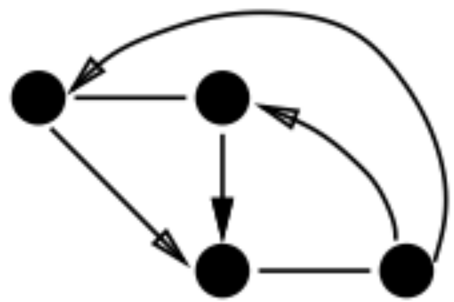
➔ With a symmetric ordering the associated digraphs of A and B are isomorphic.

➔ Properties like diagonal dominant and eigenvalues do not changes with symmetric orderings

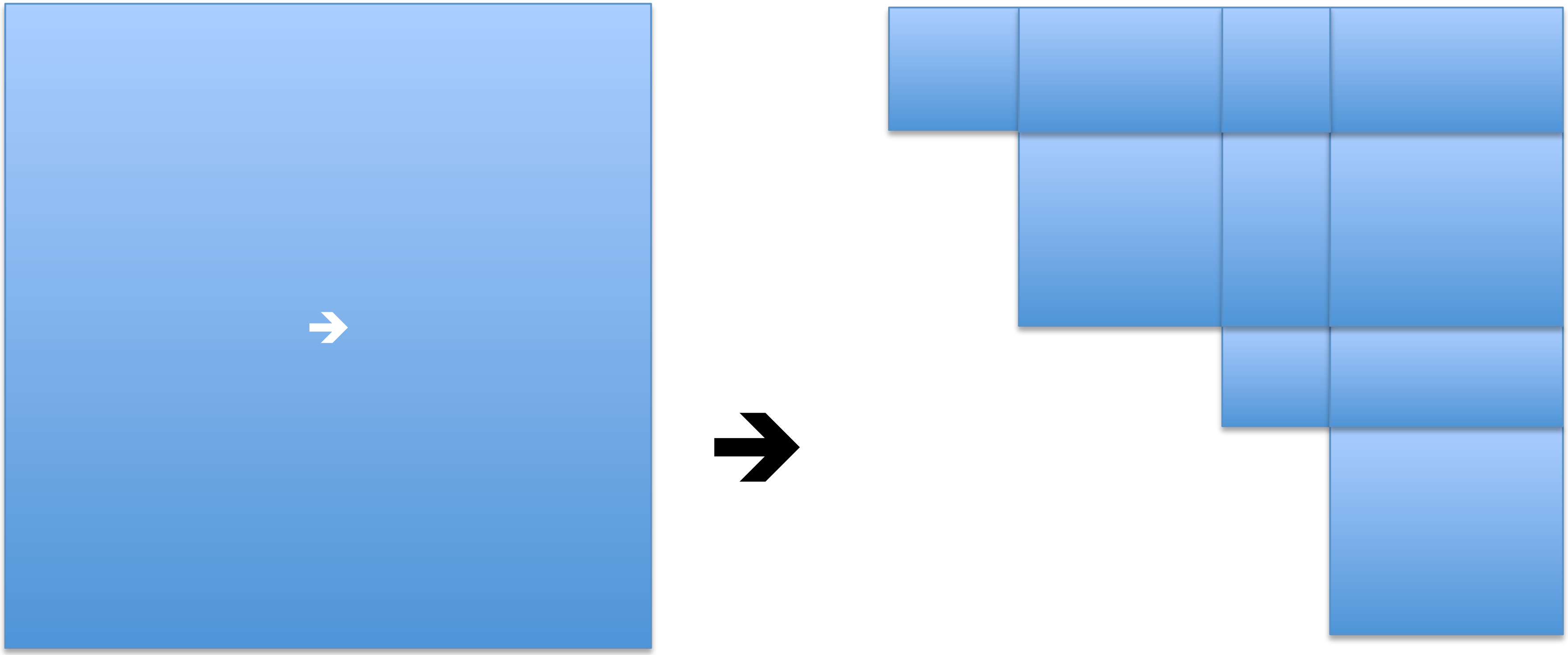
Example

$$
\begin{array}{lccccc}
 & x & x & & x & \\
 & x & x & & x & \\
\text{row} & & & x & x & \uparrow \\
\text{interchange} & x & x & x & x & \downarrow
\end{array}
\qquad\qquad
\begin{array}{cccc}
x & x & & x \\
x & x & & x \\
x & x & x & x \\
 & & x & x
\end{array}
$$



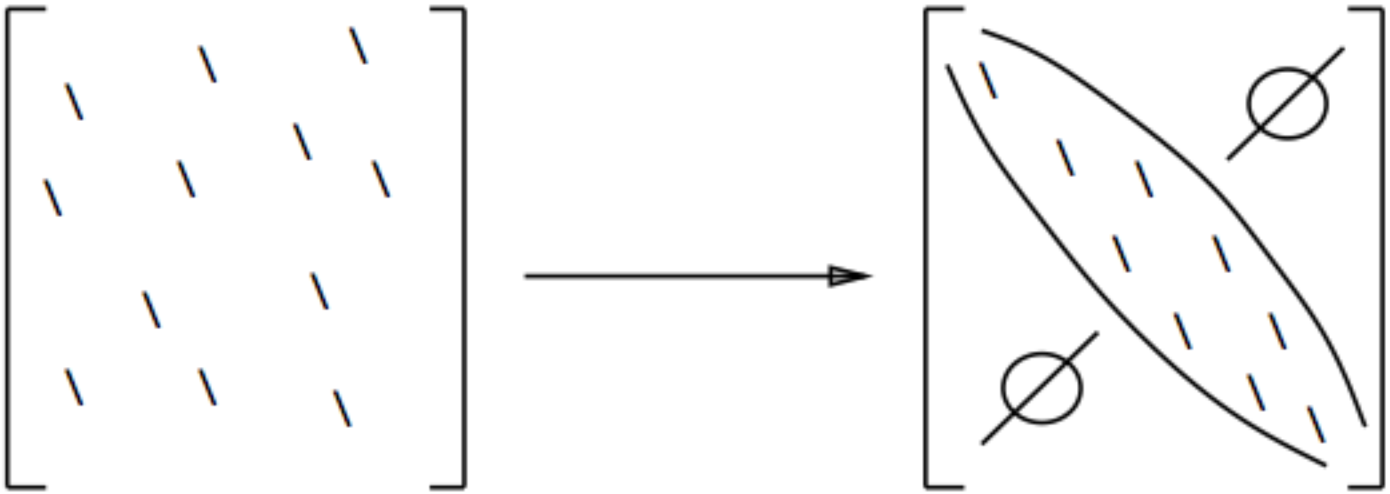


Block Triangular Form for Parallel LU



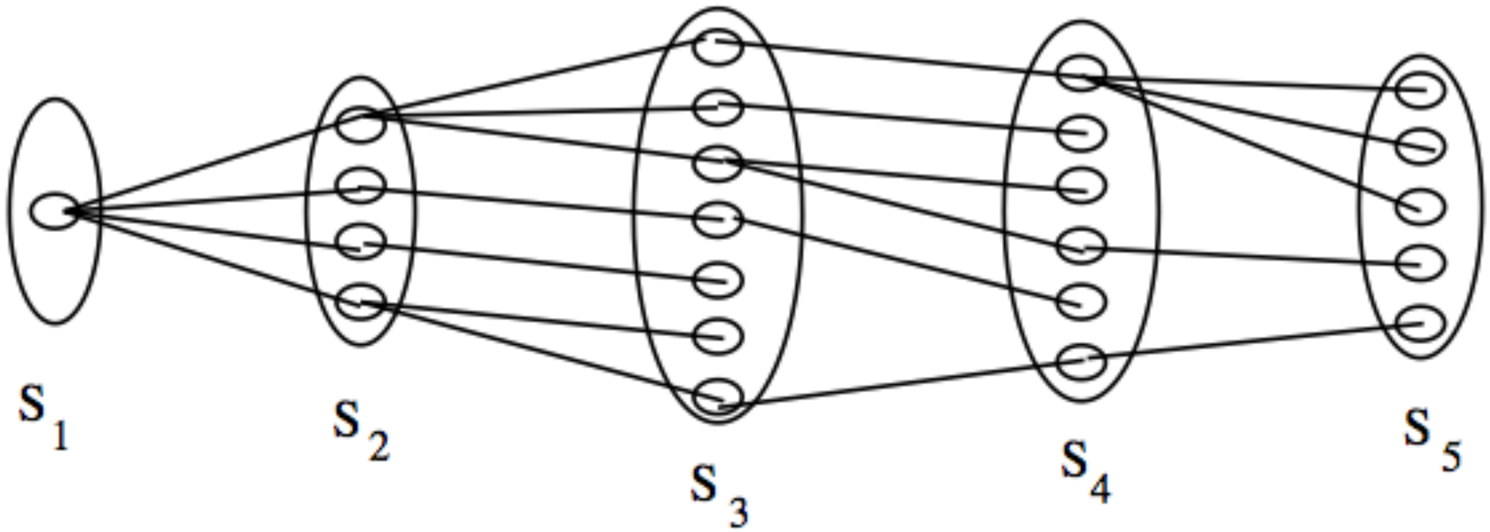
- Based on finding strongly connected components O(n+m)
- Symmetric ordering
- Unique decomposition
- Every diagonal block can be factured in parallel

Banded Structure



- Better Storage Opportunities (Diagonal Storage)
- Minimization of fill-in in LU factorization
- Better exploitation of spatial locality (stride 1 accesses)
- In some cases convergence of iterative methods are enhanced if nnz's are located near the diagonal

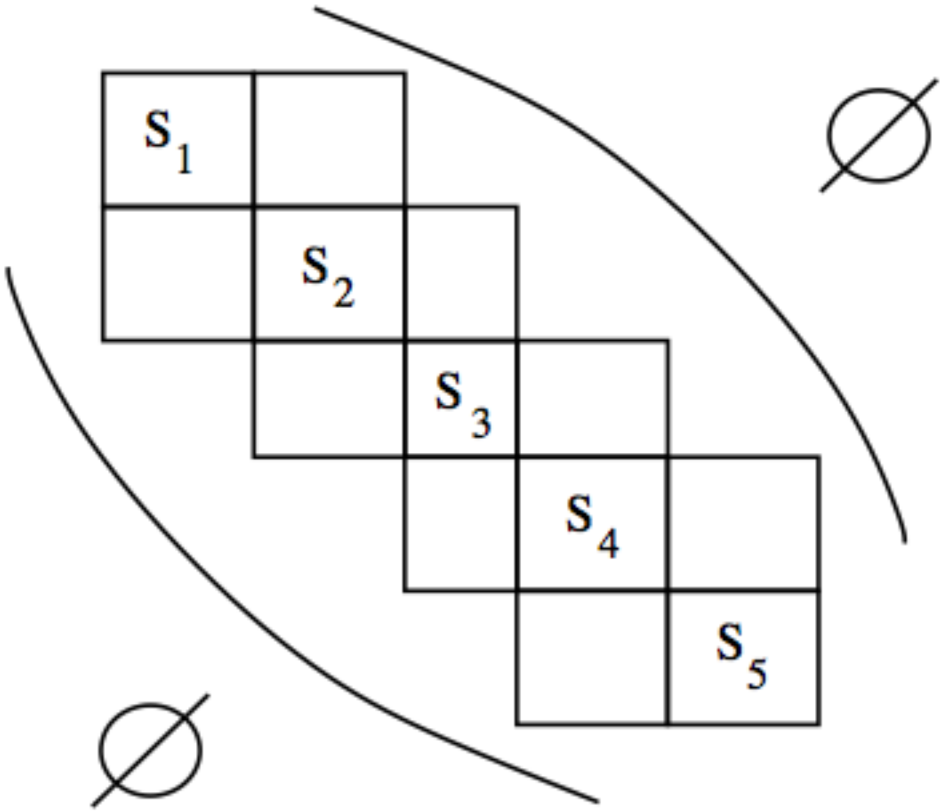
Banded structure through Cuthill-McKee



- Start with an arbitrary node α. Let $S_1 = \{ \alpha \}$.
- Let S_i = { nodes, which are not contained in any S_j with $j < i$ }. The nodes in S_i are ordered such that first nodes are the nodes which are neighbors of the first node in S_{i-1}, the following nodes are neighbors of the second node in S_{i-1}, etc.

(Basically a BFS tree is constructed of $A + A^T$)

This results in:



BTW As a side effect: Reversing Cuthill-McKee leads in many cases to minimization of fill-in

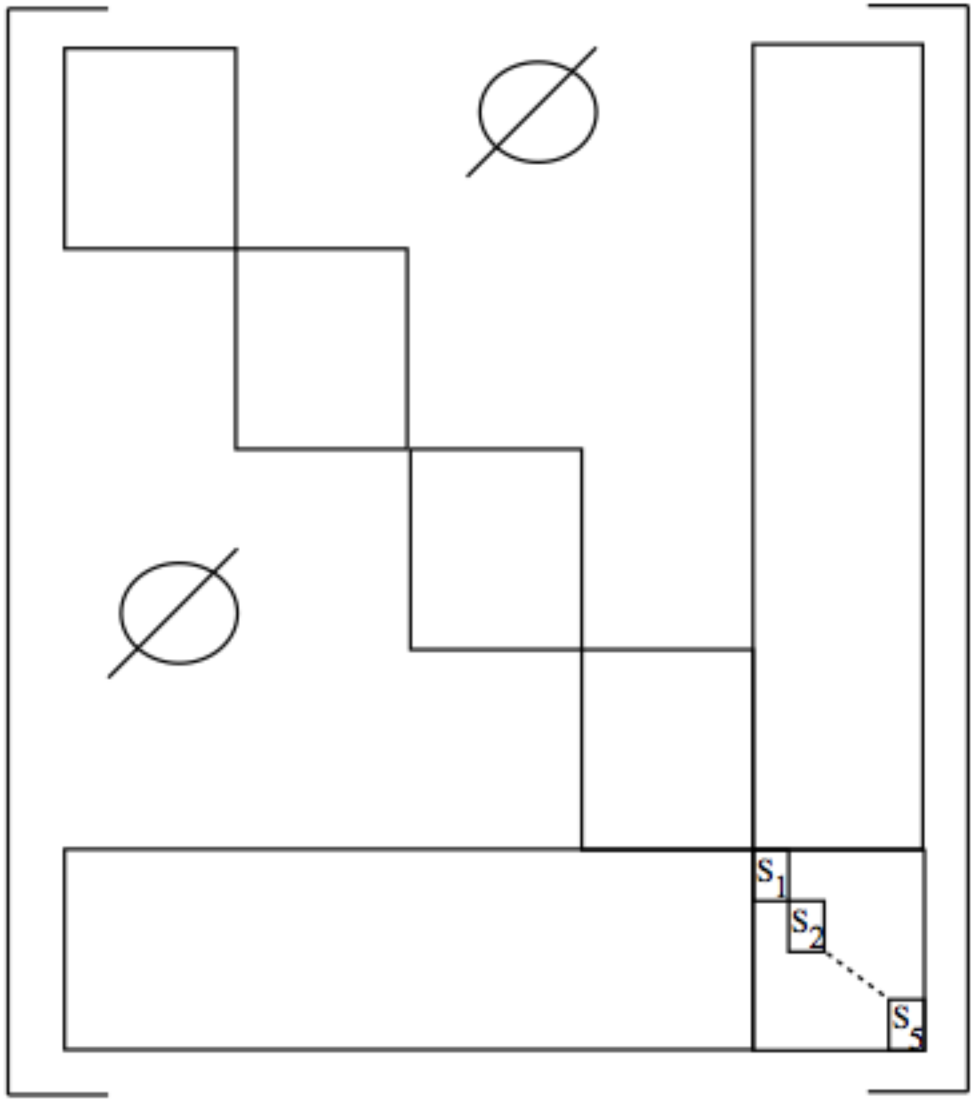
Banded Structure through One-Way/ Nested Dissection

One way dissection is based on Cuthill-KcKee:

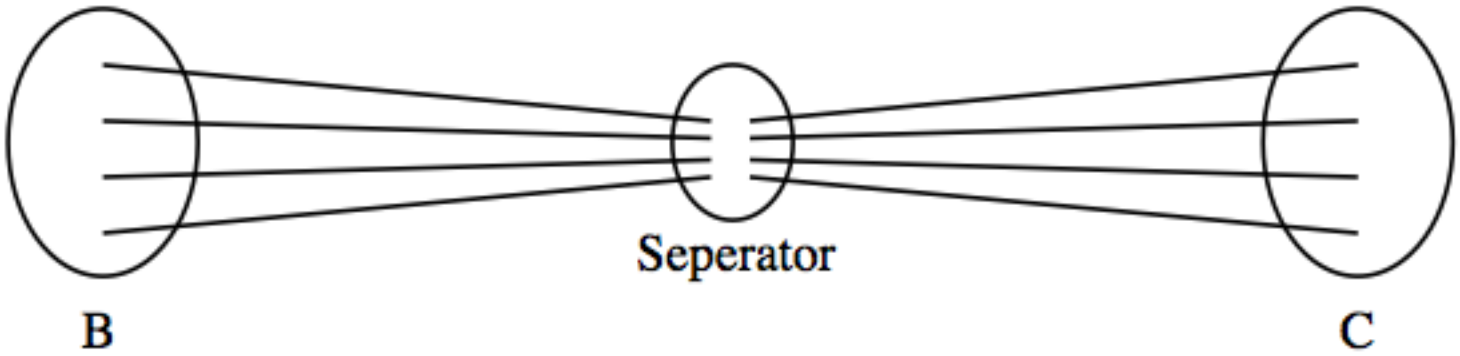
- Let $S_1\, S_2 \ldots\, S_k$ be the levelization sets obtained by Cuthill-McKee on the associated graph of a (symmetric) matrix A
- Compute
$$m = \left\lfloor (\, \Sigma_{i=1,2..,k} S_i \,) \,/\, k \right\rfloor,$$
the average number of elements per set.
- Let
$$\delta = \surd\, (\, (\, 3m + 13 \,) \,/\, 2 \,)$$
- Take all the nodes from sets S_j with $j = \lfloor i\delta + 0.5 \rfloor,\ i = 1,2,\ldots$
- Number these nodes last

The choice of δ is based on experiments run on regular grid matrices.

This results in the following matrix



The same result can be obtained by nested dissection

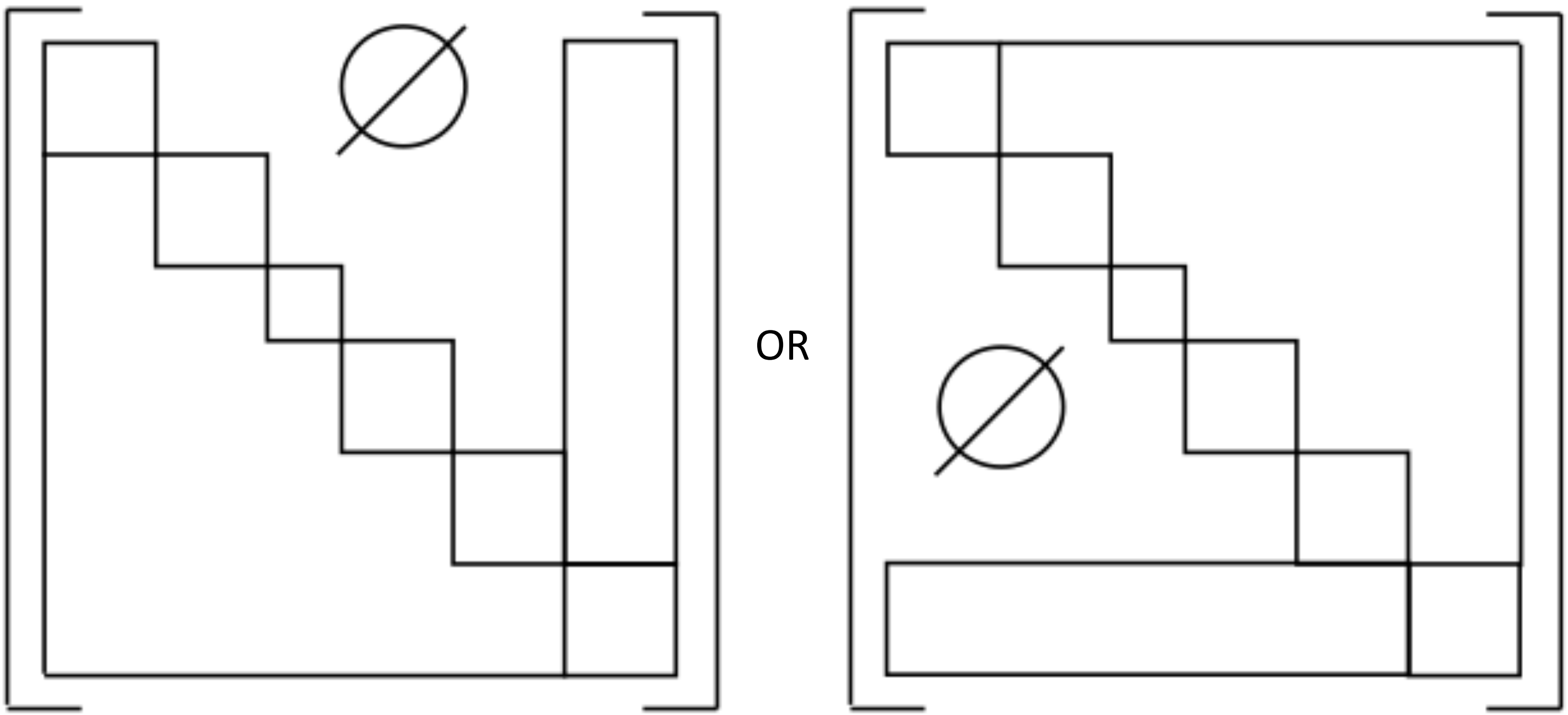


And recursively computing separator sets for B and C, and so on, and so on....

Number the nodes of these separator sets last

➔ As a result we have a more general method, not only suited for grid matrices.

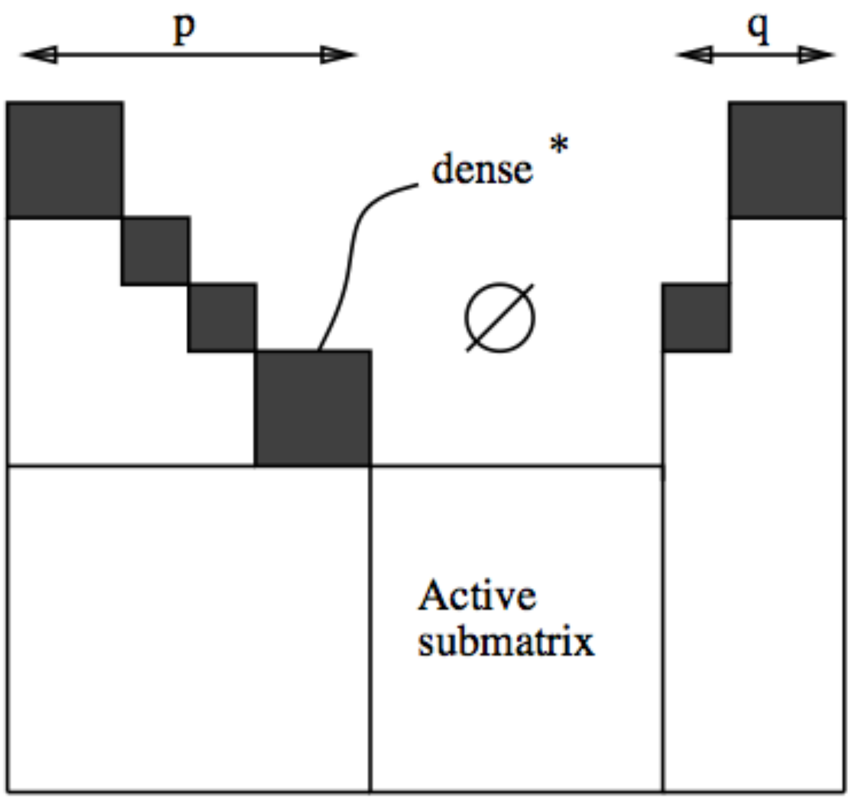
Tearing Techniques



A large grain decomposition for computing LU factorization in parallel

The desired form is bordered upper block triangular form

Hellerman-Rarick



- Unsymmetric Ordering
- Diagonal elements are assigned pivots
- The q columns are called spikes and will form the border

The algorithm

1. $p=0$, $q=0$ and the whole matrix is active
2. m is the minimum NNZ entries in any row of the active (sub)matrix. Choose m columns, by choosing first the column with most NNZ's in rows with NNZ-count of m, then the column is chosen with most NNZ's in rows with NNZ-count of m-1, and so on.
3. If the last column has s rows with a singleton NNZ then these rows are permuted to the beginning of the active (sub) matrix and these rows are assigned pivot rows
4. The last s columns chosen are also permuted to the front of the active (sub) matrix and these columns are assigned pivot columns
5. The remaining m-s columns are permuted to the border
6. $p = p + s$ and $q = q + m - s$
7. If $p + q = n$ then stop else goto 2

Example

$$
\begin{array}{c c}
 & \begin{array}{cccccc} 1 & 2 & 3 & 4 & 5 & 6 \end{array} \\
\begin{array}{c} 1 \\ 2 \\ 3 \\ 4 \\ 5 \\ 6 \end{array} &
\left[\begin{array}{cccccc}
X & X & X &   & X &   \\
X &   &   & X &   & X \\
X & X & X & X &   &   \\
X &   &   & X &   & X \\
  & X & X & X & X & X \\
X & X & X &   & X &
\end{array}\right]
\end{array}
\qquad m = 3
$$

Column 1 is chosen first: it has most entries in rows of count 3.
Then column 4 is chosen, because it has most entries in rows with new count 2.
Then column 6 is chosen because it has singletons in rows 2 and 4 .
➔ Rows 2 and 4 are permuted to the front

Example 2

$$
\begin{array}{c}
\begin{array}{cccccc} 1 & 2 & 3 & 4 & 5 & 6 \end{array} \\
\begin{array}{c} 4 \\ 2 \\ 3 \\ 1 \\ 5 \\ 6 \end{array}
\left[\begin{array}{cccccc}
X & & & X & & X \\
X & & & X & & X \\
X & X & X & X & & \\
X & X & X & & X & \\
 & X & X & X & X & X \\
X & X & X & & X &
\end{array}\right]
\end{array}
$$

Now columns 6 and 4 are permuted to the front and column 1 is permuted to the back

As a results we have:

DENSE

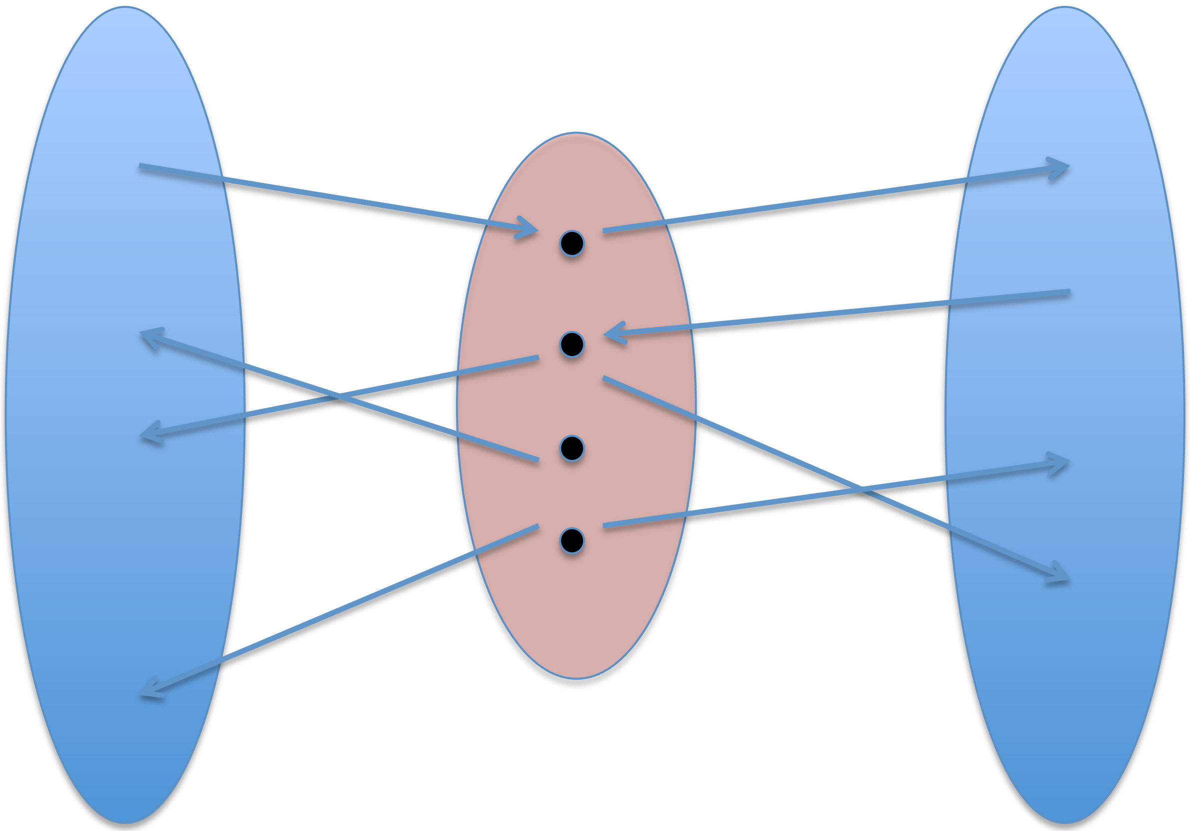
$$
\begin{array}{c}
\begin{array}{cccccc} 6 & 4 & 2 & 5 & 3 & 1 \end{array} \\
\begin{array}{c} 4 \\ 2 \\ 3 \\ 1 \\ 5 \\ 6 \end{array}
\left[\begin{array}{cc|ccc|c}
X & X & & & & X \\
X & X & & & & X \\
\hline
 & X & X & & X & X \\
 & & X & X & X & X \\
X & X & X & X & X & \\
 & & X & X & X & X
\end{array}\right]
\end{array}
$$

DENSE

active

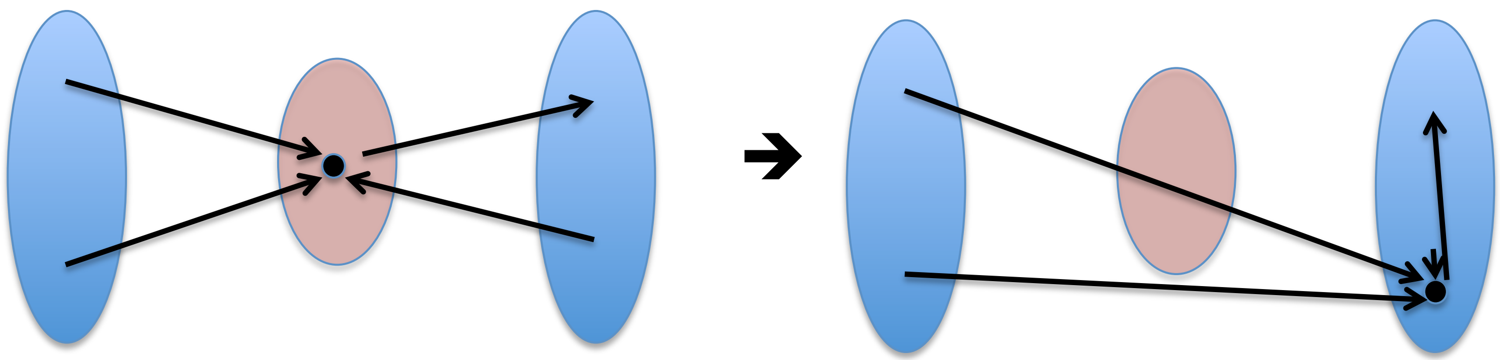
Tearing based on nested dissection

Remark: Separator sets were constructed on $A + A^T$

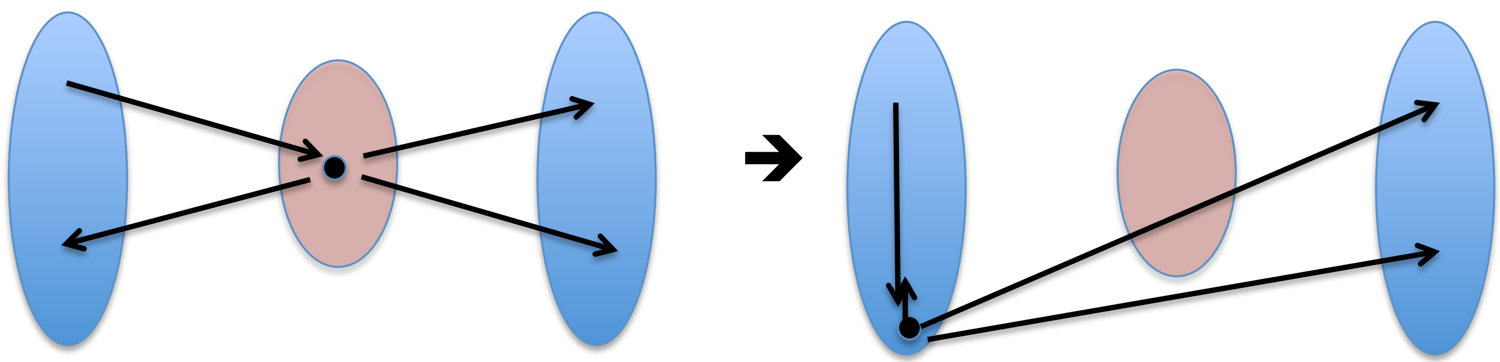


Edges from the separators can go both directions to B and C

For nodes u in S with only incoming edges from B, move u to C



For nodes v in S with only outgoing edges to C, move v to B



➔ As a result the size of the separator sets (border) is reduced, while there are NNZ introduced in the upper triangular part

A Hybrid Reordering H*

- H0: Through an asymmetric ordering $A' = PAQ^T$ permute "large values to the diagonal", i.e. for each k find the largest a_{mn} such that $|a_{mn}| >= |a_{ij}|$, for all $a_{ij} \varepsilon A_{kk}$. Permute row k and row m, permute column k and column n.
- H1: Find strongly connected components using Tarjan's algorithm, and permute the matrix with a symmetric ordering into block upper triangular form: $A'' = VA'V^T$
- H2: Use tearing based on nested dissection on each diagonal block, and number all nodes of the separator sets last. As a results the (block upper triangular) matrix is transformed into a bordered block upper triangular matrix: $A''' = WA''W^T$
- So $A''' = WVPAQ^TV^TW^T$ and the L and U factors can be computed in parallel using the diagonal elements as pivots