REORDERINGS FOR FILL-REDUCTION

- **Permutations and reorderings - graph interpretations**
- **Band-reduction orderings: Cuthill-Mc Kee, Reverse Cuthill Mc Kee**
- **Profile/envelope methods. Profile reduction.**
- **Multicoloring and independent sets [for iterative methods]**
- **Minimal degree ordering**
- **Nested Dissection**

Reorderings and graphs

➤ Let $\pi = \{i_1, \cdots, i_n\}$ a permutation

➤ $A_{\pi,*} = \{a_{\pi(i),j}\}_{i,j=1,\ldots,n}$ = matrix A with its i-th row replaced by row number $\pi(i)$.

➤ $A_{*,\pi}$ = matrix A with its j-th column replaced by column $\pi(j)$.

➤ Define $P_\pi = I_{\pi,*}$ = "Permutation matrix" – Then:

(1) Each row (column) of P_π consists of zeros and exactly one "1"
(2) $A_{\pi,*} = P_\pi A$
(3) $P_\pi P_\pi^T = I$
(4) $A_{*,\pi} = A P_\pi^T$

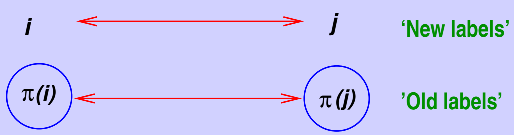
Consider now:

$$A' = A_{\pi,\pi} = P_\pi A P_\pi^T$$

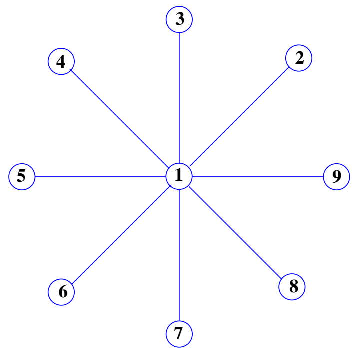
➤ Element in position (i,j) in matrix A' is exactly element in position $(\pi(i), \pi(j))$ in A. ($a'_{ij} = a_{\pi(i),\pi(j)}$)

$$(i,j) \in E_{A'} \iff (\pi(i), \pi(j)) \in E_A$$

General picture :



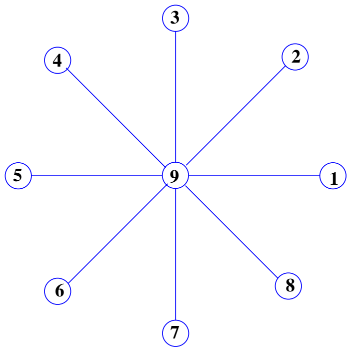
Example: A 9 × 9 'arrow' matrix and its adjacency graph.



$$\begin{bmatrix} * & * & * & * & * & * & * & * & * \\ * & * & & & & & & & \\ * & & * & & & & & & \\ * & & & * & & & & & \\ * & & & & * & & & & \\ * & & & & & * & & & \\ * & & & & & & * & & \\ * & & & & & & & * & \\ * & & & & & & & & * \end{bmatrix}$$

✍1 Fill-in?

➤ Graph and matrix after swapping nodes 1 and 9:



$$\begin{bmatrix} * & & & & & & & & * \\ & * & & & & & & & * \\ & & * & & & & & & * \\ & & & * & & & & & * \\ & & & & * & & & & * \\ & & & & & * & & & * \\ & & & & & & * & & * \\ & & & & & & & * & * \\ * & * & * & * & * & * & * & * & * \end{bmatrix}$$

✍₂ Fill-in?

The Cuthill-McKee and its reverse orderings

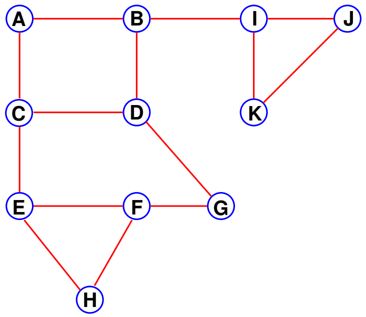
➤ A class of reordering techniques which proceed by levels in the graph.

➤ Related to Breadth First Search (BFS) traversal in graph theory.

➤ Idea of BFS is to visit the nodes by 'levels'. Level 0 = level of starting node.

➤ Start with a node, visit its neighbors, then the (unmarked) neighbors of its neighbors, etc...

Example:



h	Queue
A	A* B, C
B	A, B* C, D, I
C	A, B, C* D, I, E
D	A, B, C , D* I, E, G
I	A, B, C , D , I* E, G, J, K
E	A, B, C , D , I, E* G, J, K, F, H
G	...
E	A, B, C , D , I, E, G, J, K, F, H*

➤ Final traversal order: | A, | B, C, | D, I, E, | G, J, K, F, H |

Level 0 — Level 1 — Level 2 — Level 3

➤ Levels represent distances from the root

➤ Algorithm can be implemented by crossing levels 1,2, ...

➤ More common: Queue implementation

Algorithm $\boldsymbol{BFS(G, v)}$ – Queue implementation

- Initialize: $\boldsymbol{Queue := \{v\}}$; Mark $\boldsymbol{v}$; $\boldsymbol{ptr = 1}$;
- While $\boldsymbol{ptr < length(Queue)}$ do
  - $\boldsymbol{head = Queue(ptr)}$;
  - ForEach Unmarked $\boldsymbol{w} \in$ $\boldsymbol{Adj(head)}$:
    - $*$ Mark $\boldsymbol{w}$;
    - $*$ Add $\boldsymbol{w}$ to Queue: Queue = {Queue, w};
  - $\boldsymbol{ptr++}$;

```
   function [p] = bfs(A,init )
%% BFS traversal. queue implementation
%%--------------------  enqueue first node
 p=[init];
 n = size(A,1);
 mask = zeros(n,1);
 mask(init) = 1;
%%-------------------- main loop
for h=1:n
%%-------------------- scan nodes in adj(p(h))
    [ii, jj, rr] = find(A(:,p(h)));
    for v=ii'
        if (mask(v)==0)
          p = [p, v] ;
          mask(v) = 1;
        end
    end
end
```


A few properties of Breadth-First-Search

➤ If G is a connected undirected graph then each vertex will be visited once; each edge will be inspected at least once

➤ Therefore, for a connected undirected graph,

The cost of BFS is $O(|V| + |E|)$

➤ Distance = level number; ➤ For each node v we have:

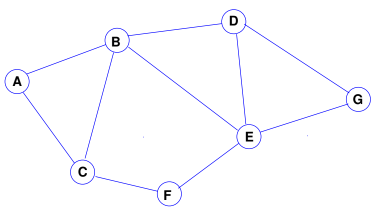
$$min\_dist(s,v) = level\_number(v) = depth_T(v)$$

➤ Several reordering algorithms are based on variants of Breadth-First-Search

Cuthill McKee ordering

Same as BFS except: Adj($head$) always sorted by increasing degree

Example:



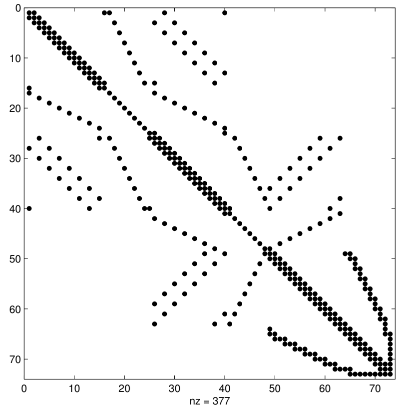
A	C(3) B(4)
A, C	B, F(2)
A, C, B	F, D(3), E(4)
A, C, B, F	D, E
A, C, B, F, D	E, G(2)
A, C, B, F, D, E	G
A, C, B, F, D, E, G	

Rule: when adding nodes to the queue list them in ↑ deg.

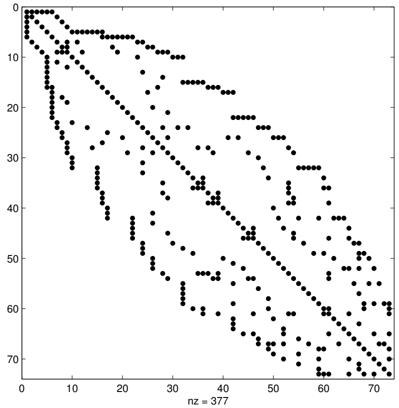
Reverse Cuthill McKee ordering

➤ The Cuthill - Mc Kee ordering has a tendency to create small arrow matrices (going the wrong way):

Origimal matrix

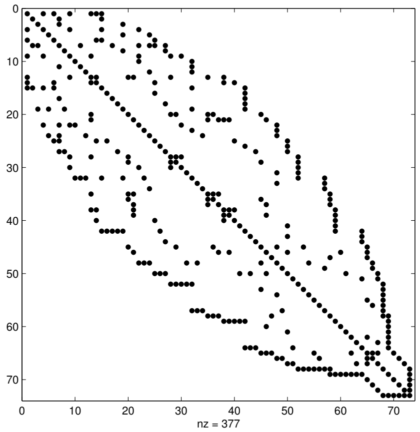


CM ordering



➤ Idea: Take the reverse ordering

RCM ordering



➤ Reverse Cuthill M Kee ordering (RCM).

Envelope/Profile methods

Many terms used for the same methods: Profile, Envelope, Skyline, ...

➤ Generalizes band methods

➤ Consider only the symmetric (in fact SPD) case

➤ Define bandwith of row i. ("i-th bandwidth of A):

$$\beta_i(A) = \max_{j \leq i; a_{ij} \neq 0} |i - j|$$

Definition: Envelope of A is the set of all pairs (i, j) such that $0 < i - j \leq \beta_i(A)$. The quantity $|Env(A)|$ is called profile of A.

Main result The envelope is preserved by GE (no-pivoting)

Theorem: Let $A = LL^T$ the Cholesky factorization of A. Then

$$Env(A) = Env(L + L^T)$$

➤ An envelope / profile/ Skyline method is a method which treats any entry a_{ij}, with $(i, j) \in Env(A)$ as nonzero.

Matlab test: do the following

1. Generate `A = Lap2D(64,64)`
2. Compute `R = chol(A)`
3. show `nnz(R)`
4. Compute RCM permutation (symrcm)
5. Compute `B = A(p,p)`
6. `spy(B)`
7. compute `R1 = chol(B)`
8. Show `nnz(R)`
9. `spy(R1)`

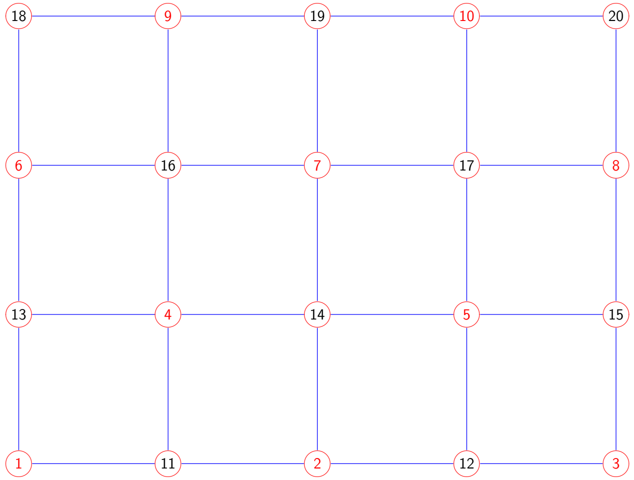
Orderings for parallelism: Multicoloring

➤ General technique that can be exploited in many different ways to introduce parallelism – generally of order N.

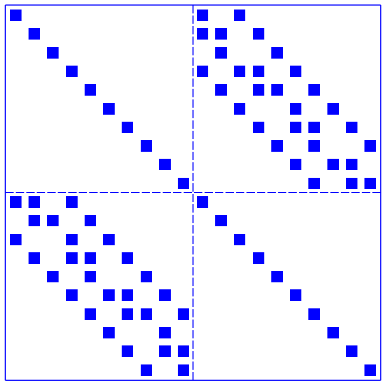
➤ Constitutes one of the most successful techniques for introducing vector computations for iterative methods..

➤ Want: assign colors so that no two adjacent nodes have the same color.

Simple example: Red-Black ordering.



Corresponding matrix



➤ Observe: L-U solves with lower and upper parts of A will require only diagonal scalings + matrix-vector products with matrices of size $N/2$.

How to generalize Red-Black ordering?

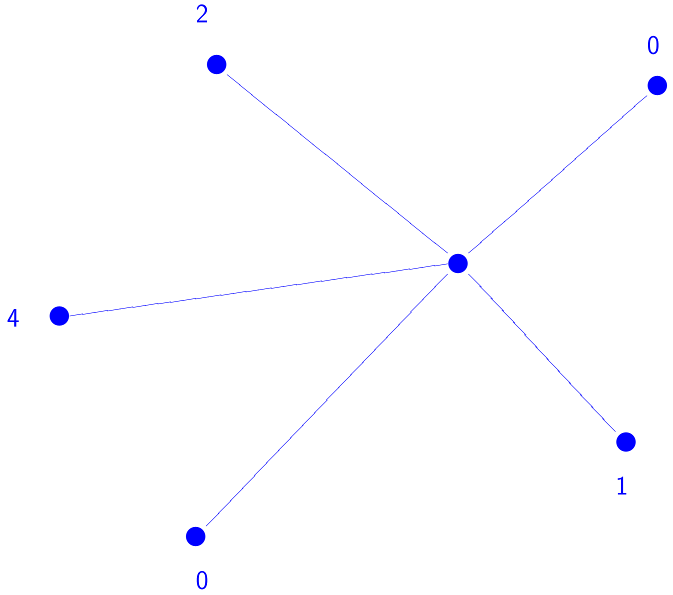
Answer: **Multicoloring** & **independent sets**

A greedy multicoloring technique:

- Initially assign color number zero (uncolored) to every node.
- Choose an order in which to traverse the nodes.
- Scan all nodes in the chosen order and at every node i do

$$Color(i) = \min\{k \neq 0 | k \neq Color(j), \forall\ j \in \text{ Adj (i)}\}$$

Adj(i) = set of nearest neighbors of $i = \{k \mid a_{ik} \neq 0\}$.



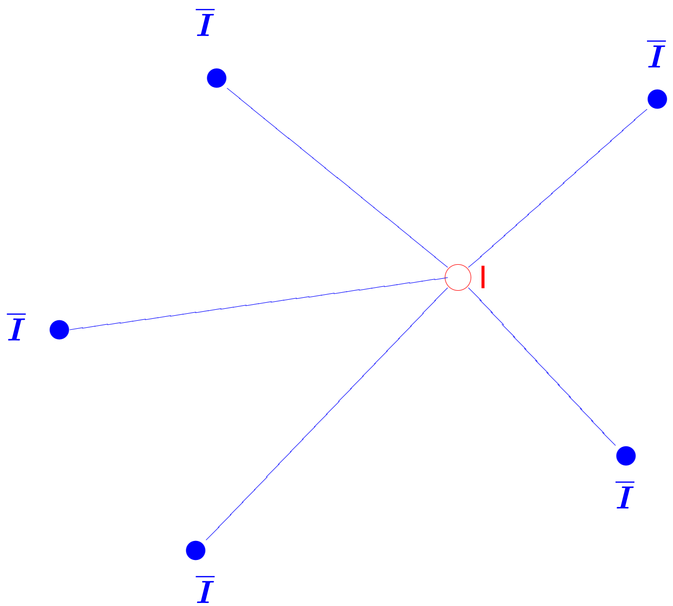
Independent Sets

An independent set (IS) is a set of nodes that are not coupled by an equation. The set is maximal if all other nodes in the graph are coupled to a node of IS. If the unknowns of the IS are labeled first, then the matrix will have the form:

$$\begin{bmatrix} B & F \\ E & C \end{bmatrix}$$

in which B is a diagonal matrix, and E, F, and C are sparse.

Greedy algorithm: Scan all nodes in a certain order and at every node i do: if i is not colored color it Red and color all its neighbors Black. Independent set: set of red nodes. Complexity: $O(|E| + |V|)$.



✍3 Show that the size of the independent set I is such that

$$|I| \geq \frac{n}{1 + d_I}$$

where d_I is the maximum degree of each vertex in I (not counting self cycle).

✍4 According to the above inequality what is a good (heuristic) order in which to traverse the vertices in the greedy algorithm?

✍5 Are there situations when the greedy alorithm for independent sets yield the same sets as the multicoloring algorithm?

Orderings used in direct solution methods

➤ Two broad types of orderings used:

- Minimal degree ordering + many variations
- Nested dissection ordering + many variations

➤ Minimal degree ordering is easiest to describe:

At each step of GE, select next node to eliminate, as the node v of smallest degree. After eliminating node v, update degrees and repeat.

Minimal Degree Ordering

At any step i of Gaussian elimination define for any candidate pivot row j

$$\boxed{Cost(j) = (nz_c(j) - 1)(nz_r(j) - 1)}$$

where $nz_c(j)$ = number of nonzero elements in column j of 'active' matrix, $nz_r(j)$ = number of nonzero elements in row j of 'active' matrix.

➤ Heuristic: fill-in at step j is $\leq cost(j)$

➤ Strategy: select pivot with minimal cost.

➤ Local, greedy algorithm

➤ Good results in practice.

Many improvements made over the years

- Alan George and Joseph W-H Liu, The evolution of the minimum degree ordering algorithm, SIAM Review, vol 31 (1989), pp. 1-19.

Min. Deg. Algorithm	Storage (words)	Order. time
Final min. degree	1,181 K	43.90
Above w/o multiple elimn.	1,375 K	57.38
Above w/o elimn. absorption	1,375 K	56.00
Above w/o incompl. deg. update	1,375 K	83.26
Above w/o indistiguishible nodes	1,308 K	183.26
Above w/o mass-elimination	1,308 K	2289.44

➤ Results for a 180×180 9-point mesh problem

➤ Since this article, many important developments took place.

➤ In particular the idea of "Approximate Min. Degree" and and "Approximate Min. Fill", see

- E. Rothberg and S. C. Eisenstat, Node selection strategies for bottom-up sparse matrix ordering, SIMAX, vol. 19 (1998), pp. 682-695.

- Patrick R. Amestoy, Timothy A. Davis, and Iain S. Duff. An Approximate Minimum Degree Ordering Algorithm. SIAM Journal on Matrix Analysis and Applications, 17 (1996), pp. 886-905.

Practical Minimal degree algorithms

First Idea: Use quotient graphs

* Avoids elimination graphs which are not economical

* Elimination creates cliques

* Represent each clique by a node termed an *element* (recall FEM methods)

* No need to create fill-edges and elimination graph

* Still expensive: updating the degrees

Second idea: Multiple Minimum degree

* Many nodes will have the same degree. Idea: eliminate many of them simultaneously –

* Specifically eliminate independent sets of nodes with same degree.

Third idea: Approximate Minimum degree

* Degree updates are expensive –

* Goal: To save time.

* Approach: only compute an approximation (upper bound) to degrees.

* Details are complex and can be found in Tim Davis' book

✍6 Explore *symamd* and *amd* in matlab

Nested Dissection Reordering (Alan George)

➤ Computer science 'Divide-and-Conquer' strategy.

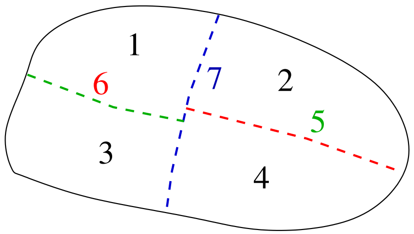
➤ Best illustration: PDE finite difference grid.

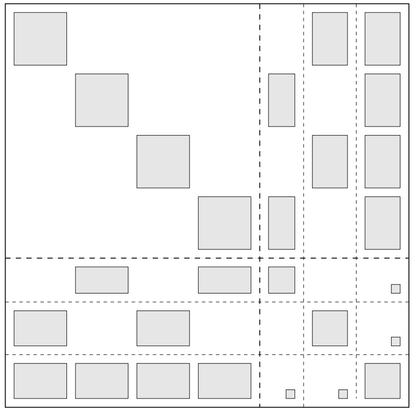
➤ Easily described by using recursivity and by exploiting 'separators': 'separate' the graph in three parts, two of which have no coupling between them. The 3rd set ('the separator') has couplings with vertices from both of the first 2 sets.

➤ Key idea: dissect the graph; take the subgraphs and dissect them recursively.

➤ Nodes of separators always labeled last after those of the parents

Nested dissection ordering: illustration



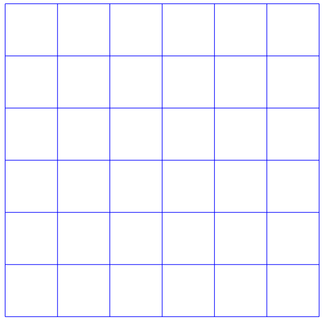


➤ For regular $n \times n$ meshes, can show: fill-in is of order $n^2 \log n$ and computational cost of factorization is $O(n^3)$

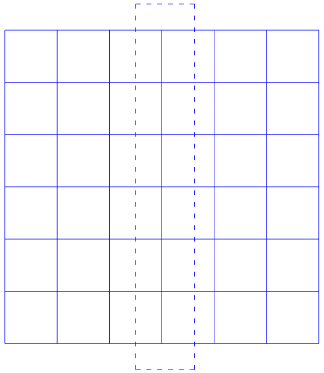
✍7 How does this compare with a standard band solver?

Nested dissection for a small mesh

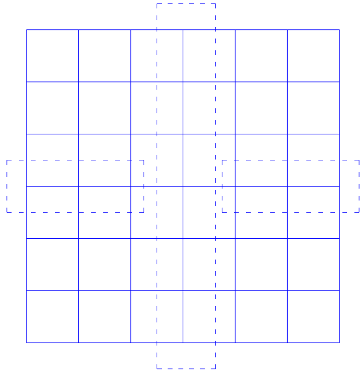
Original Grid



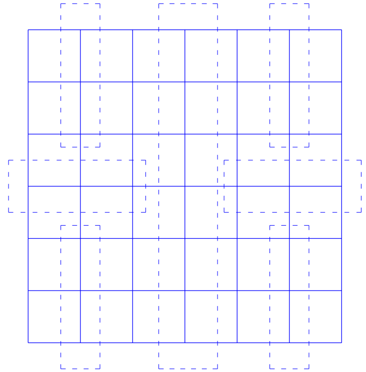
First dissection



Second Dissection



Third Dissection



Nested dissection: cost for a regular mesh

- In 2-D consider an $n \times n$ problem, $N = n^2$
- In 3-D consider an $n \times n \times n$ problem, $N = n^3$

	2-D	3-D
space (fill)	$O(N \log N)$	$O(N^{4/3})$
time (flops)	$O(N^{3/2})$	$O(N^2)$

- Significant difference in complexity between 2-D and 3-D

Nested dissection and separators

- Nested dissection methods depend on finding a good graph separator: $V = T_1 \cup U T_2 \cup S$ such that the removal of S leaves T_1 and T_2 disconnected.

- Want: S small and T_1 and T_2 of about the same size.

- Simplest version of the graph partitioning problem.

A theoretical result:

If G is a planar graph with N vertices, then there is a separator S of size $\leq \sqrt{N}$ such that $|T_1| \leq 2N/3$ and $|T_2| \leq 2N/3$.

In other words "Planar graphs have $O(\sqrt{N})$ separators"

- Many techniques for finding separators: Spectral, iterative swapping (K-L), multilevel (Metis), BFS, ...