Numerically-aware Nested Dissection Ordering

Jonathan Hogg, Jennifer Scott and Sue Thorne

STFC Rutherford Appleton Laboratory

1 July 2016
Sparse Days at CERFACS 2016
Toulouse, France

Introduction

Solve: $$Ax = b$$

A is:

- Sparse
- Large
- Symmetric
- **Indefinite** - may need pivoting

Sparse factorization phases

Traditionally:

1. Find fill-reducing ordering
2. Find diagonal scaling
3. Perform numerical factorization

Sparse factorization phases

Traditionally:

1. Find fill-reducing ordering
2. Find diagonal scaling
3. Perform numerical factorization

For difficult problems:

1. Find diagonal scaling using Hungarian algorithm (MC64)
2. Use matching from Hungarian algorithm to compress matrix
3. Find fill-reducing ordering on compressed matrix
4. Uncompress ordering to one on original matrix
5. Perform numerical factorization

Sparse factorization phases

Traditionally:

1. Find fill-reducing ordering
2. Find diagonal scaling
3. Perform numerical factorization

For difficult problems:

1. Find diagonal scaling using Hungarian algorithm (MC64)
2. Use matching from Hungarian algorithm to compress matrix
3. Find fill-reducing ordering on compressed matrix
4. Uncompress ordering to one on original matrix
5. Perform numerical factorization

Aim: replace steps 2-4 with something more intelligent.

	1	2	3	4	5	6
1	0.0	0.9			1.0	0.2
2	0.9	0.0			0.1	1.0
3			1.0	0.2	0.4	0.9
4			0.2	1.0	0.2	0.6
5	1.0	0.1	0.4	0.2	0.5	0.2
6	0.2	1.0	0.9	0.6	0.2	0.8

- Simple nested dissection structure

Science & Technology Facilities Council

$$
\begin{array}{c|cccccc|}
 & 1 & 2 & 3 & 4 & 5 & 6 \\
\hline
1 & 0.0 & 0.9 & & & \circled{1.0} & 0.2 \\
2 & 0.9 & 0.0 & & & 0.1 & \circled{1.0} \\
3 & & & \circled{1.0} & 0.2 & 0.4 & 0.9 \\
4 & & & 0.2 & \circled{1.0} & 0.2 & 0.6 \\
5 & \circled{1.0} & 0.1 & 0.4 & 0.2 & 0.5 & 0.2 \\
6 & 0.2 & \circled{1.0} & 0.9 & 0.6 & 0.2 & 0.8
\end{array}
$$

- Simple nested dissection structure
- Hungarian scaled

	1	2	3	4	5	6
1	0.0	0.9			1.0	0.2
2	0.9	0.0			0.1	1.0
3			1.0	0.2	0.4	0.9
4			0.2	1.0	0.2	0.6
5	1.0	0.1	0.4	0.2	0.5	0.2
6	0.2	1.0	0.9	0.6	0.2	0.8

- Simple nested dissection structure
- Hungarian scaled

But what if we force compression...

Science & Technology Facilities Council

	1	2	3	4	5	6
1	0.0	0.9			1.0	0.2
2	0.9	0.0			0.1	1.0
3			1.0	0.2	0.4	0.9
4			0.2	1.0	0.2	0.6
5	1.0	0.1	0.4	0.2	0.5	0.2
6	0.2	1.0	0.9	0.6	0.2	0.8

- Simple nested dissection structure
- Hungarian scaled

But what if we force compression...

	1	5	2	6	3	4
1	0.0	1.0	0.9	0.2		
5	1.0	0.5	0.1	0.2	0.4	0.2
2	0.9	0.1	0.0	1.0		
6	0.2	0.2	1.0	0.8	0.9	0.6
3		0.4		0.9	1.0	0.2
4		0.2		0.6	0.2	1.0

- Forced pivots
- More numerically stable/Less pivoting

	1	2	3	4	5	6
1	0.0	0.9			1.0	0.2
2	0.9	0.0			0.1	1.0
3			1.0	0.2	0.4	0.9
4			0.2	1.0	0.2	0.6
5	1.0	0.1	0.4	0.2	0.5	0.2
6	0.2	1.0	0.9	0.6	0.2	0.8

- Simple nested dissection structure
- Hungarian scaled

But what if we force compression...

	1	5	2	6	3	4
1	0.0	1.0	0.9	0.2		
5	1.0	0.5	0.1	0.2	0.4	0.2
2	0.9	0.1	0.0	1.0	**f**	**f**
6	0.2	0.2	1.0	0.8	0.9	0.6
3		0.4	**f**	0.9	1.0	0.2
4		0.2	**f**	0.6	0.2	1.0

- Forced pivots
- More numerically stable/Less pivoting
- More fill!

	1	2	3	4	5	6
1	0.0	0.9			1.0	0.2
2	0.9	0.0			0.1	1.0
3			1.0	0.2	0.4	0.9
4			0.2	1.0	0.2	0.6
5	1.0	0.1	0.4	0.2	0.5	0.2
6	0.2	1.0	0.9	0.6	0.2	0.8

- Simple nested dissection structure
- Hungarian scaled

But wasn't that a "good enough" pivot?

But what if we force compression...

	1	5	2	6	3	4
1	0.0	1.0	0.9	0.2		
5	1.0	0.5	0.1	0.2	0.4	0.2
2	0.9	0.1	0.0	1.0		
6	0.2	0.2	1.0	0.8	0.9	0.6
3		0.4		0.9	1.0	0.2
4		0.2		0.6	0.2	1.0

- Forced pivots
- More numerically stable/Less pivoting
- More fill!

Let's think about this...

Assumptions

- Only want to work with entries of A.
- "Large" entries stay large.
- Only concerned with "small" diagonals.

Let's think about this...

Assumptions

- Only want to work with entries of A.
- "Large" entries stay large.
- Only concerned with "small" diagonals.

Define diagonal entry a_{ii} to be:

large if $\max_{k \neq i} |a_{ik}| < u_{ord}^{-1} |a_{ii}|$,

small otherwise.

Let's think about this...

Assumptions

- Only want to work with entries of A.
- "Large" entries stay large.
- Only concerned with "small" diagonals.

Define diagonal entry a_{ii} to be:

large if $\max_{k \neq i} |a_{ik}| < u_{ord}^{-1} |a_{ii}|$,

small otherwise.

Define off-diagonal entry a_{ij} to be:

large if $$\left| \begin{pmatrix} a_{ii} & a_{ij} \\ a_{ji} & a_{jj} \end{pmatrix} \right| \begin{pmatrix} \max_{k \neq i,j} |a_{ik}| \\ \max_{k \neq i,j} |a_{jk}| \end{pmatrix} \leq u_{ord}^{-1} \begin{pmatrix} 1 \\ 1 \end{pmatrix},$$

small otherwise.

Let's think about this...

Assumptions

- Only want to work with entries of A.
- "Large" entries stay large.
- Only concerned with "small" diagonals.

Define diagonal entry a_{ii} to be:

large if $\max_{k \neq i} |a_{ik}| < u_{ord}^{-1} |a_{ii}|$,

small otherwise.

Define off-diagonal entry a_{ij} to be:

large if $\left| \begin{pmatrix} a_{ii} & a_{ij} \\ a_{ji} & a_{jj} \end{pmatrix} \right| \begin{pmatrix} \max_{k \neq i,j} |a_{ik}| \\ \max_{k \neq i,j} |a_{jk}| \end{pmatrix} \leq u_{ord}^{-1} \begin{pmatrix} 1 \\ 1 \end{pmatrix}$,

small otherwise.

Note: can ignore large a_{ij} if both a_{ii} and a_{jj} are large.

Back to our example

	1	2	3	4	5	6
1	0.0	0.9			1.0	0.2
2	0.9	0.0			0.1	1.0
3			1.0	0.2	0.4	0.9
4			0.2	1.0	0.2	0.6
5	1.0	0.1	0.4	0.2	0.5	0.2
6	0.2	1.0	0.9	0.6	0.2	0.8

Back to our example

	1	2	3	4	5	6
1	0.0	0.9			1.0	0.2
2	0.9	0.0			0.1	1.0
3			1.0	0.2	0.4	0.9
4			0.2	1.0	0.2	0.6
5	1.0	0.1	0.4	0.2	0.5	0.2
6	0.2	1.0	0.9	0.6	0.2	0.8

Problem reduces to:

- For every red diagonal
- Ensure it can be paired with a green off-diagonal
- Without messing up fill-reducing order

WARNING!

"Ensure there is a large entry in column" is insufficient.

$$\begin{pmatrix} \color{red}{\mathbf{x}} & x & \color{green}{\mathbf{x}} \\ x & \color{red}{\mathbf{x}} & \color{green}{\mathbf{x}} \\ \color{green}{\mathbf{x}} & \color{green}{\mathbf{x}} & x \end{pmatrix}$$

WARNING!

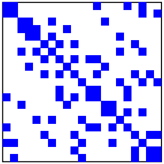
"Ensure there is a large entry in column" is insufficient.

$$\begin{pmatrix} \mathbf{x} & x & \mathbf{x} \\ x & \mathbf{x} & \mathbf{x} \\ \mathbf{x} & \mathbf{x} & x \end{pmatrix}$$

Big entries in row 3 can only be paired with a_{11} **or** a_{22}, not both!

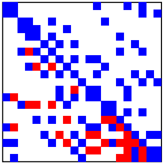
Need a matching!

Nested Dissection Primer



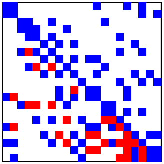
1. Original matrix

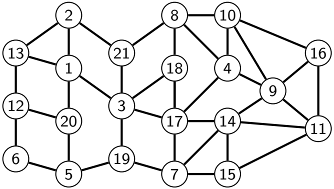
Nested Dissection Primer



1. Original matrix (with fill)

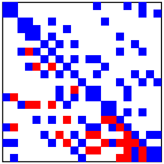
Nested Dissection Primer

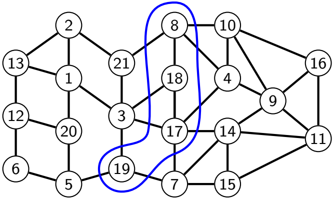




1. Original matrix (with fill)
2. Represent as graph

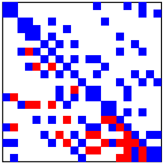
Nested Dissection Primer

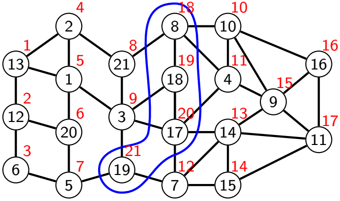




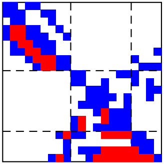
1. Original matrix (with fill)
2. Represent as graph
3. Find separator

Nested Dissection Primer



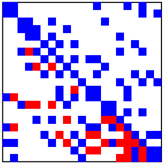


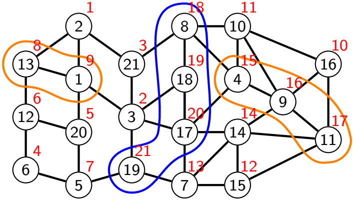
1. Original matrix (with fill)
2. Represent as graph
3. Find separator
4. Reorder to end

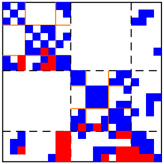


Science & Technology Facilities Council

Nested Dissection Primer







1. Original matrix (with fill)
2. Represent as graph
3. Find separator
4. Reorder to end
5. Recurse

Compatible matchings

Matching $\mathcal{M}$

- Set of pairs (i, j) such that i and j occur in a most one pair.
- Pairs represent 2×2 pivots, i.e. a_{ij} must be **large**.
- Need not have full cardinality.

Science & Technology Facilities Council

Compatible matchings

Matching $\mathcal{M}$

- Set of pairs (i, j) such that i and j occur in a most one pair.
- Pairs represent 2×2 pivots, i.e. a_{ij} must be **large**.
- Need not have full cardinality.

Partition $(\mathcal{B}, \mathcal{W}, \mathcal{S})$

- Disjoint sets such that $i \in \mathcal{B}, j \in \mathcal{W} \Rightarrow a_{ij} = 0$.
- $\mathcal{S}$ is the separator.

Compatible matchings

Matching $\mathcal{M}$

- Set of pairs (i, j) such that i and j occur in a most one pair.
- Pairs represent 2×2 pivots, i.e. a_{ij} must be **large**.
- Need not have full cardinality.

Partition $(\mathcal{B}, \mathcal{W}, \mathcal{S})$

- Disjoint sets such that $i \in \mathcal{B}, j \in \mathcal{W} \Rightarrow a_{ij} = 0$.
- $\mathcal{S}$ is the separator.

Definition: $\mathcal{M}$ compatible with $(\mathcal{B}, \mathcal{W}, \mathcal{S})$ if

- $(i, j) \in \mathcal{M} \Rightarrow i$ and j in same subset.
- $i \notin \mathcal{S} \Rightarrow a_{ii}$ is **large** or some j such that $a_{ij} \in \mathcal{M}$.

Observations

Definition: $\mathcal{M}$ compatible with $(\mathcal{B}, \mathcal{W}, \mathcal{S})$ if

- $(i, j) \in \mathcal{M} \Rightarrow i$ and j in same subset.
- $i \notin \mathcal{S} \Rightarrow a_{ii}$ is **large** or some j such that $a_{ij} \in \mathcal{M}$.

Observations

- Don't care about small diagonals in $\mathcal{S}$. Either:
  1. They have a partner; or
  2. Must be updated by a **large** entry before elimination.

Observations

Definition: $\mathcal{M}$ compatible with $(\mathcal{B}, \mathcal{W}, \mathcal{S})$ if

- $(i, j) \in \mathcal{M} \Rightarrow i$ and j in same subset.
- $i \notin \mathcal{S} \Rightarrow a_{ii}$ is **large** or some j such that $a_{ij} \in \mathcal{M}$.

Observations

- Don't care about small diagonals in $\mathcal{S}$. Either:
  1. They have a partner; or
  2. Must be updated by a **large** entry before elimination.
- Otherwise, all small diagonals have a matched entry in the same partition.
- As we only match on **large** entries, we have a 2×2 pivot for each small diagonal.

Observations

Definition: $\mathcal{M}$ compatible with $(\mathcal{B}, \mathcal{W}, \mathcal{S})$ if

- $(i, j) \in \mathcal{M} \Rightarrow i$ and j in same subset.
- $i \notin \mathcal{S} \Rightarrow a_{ii}$ is **large** or some j such that $a_{ij} \in \mathcal{M}$.

Observations

- Don't care about small diagonals in $\mathcal{S}$. Either:
  1. They have a partner; or
  2. Must be updated by a **large** entry before elimination.
- Otherwise, all small diagonals have a matched entry in the same partition.
- As we only match on **large** entries, we have a 2×2 pivot for each small diagonal.

$\Rightarrow$ **Compatibility is exactly what we want**

Outline Algorithm

2. Find a partition $(\mathcal{B}, \mathcal{W}, \mathcal{S})$.

4. Order $\mathcal{S}$ to end.
5. Restrict matrix to $\mathcal{B}$ and recurse.
6. Restrict matrix to $\mathcal{W}$ and recurse.

Science & Technology Facilities Council

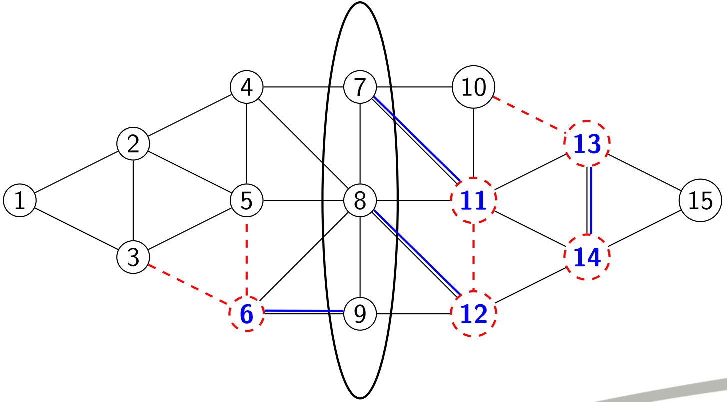
Outline Algorithm

1. Be supplied a matching $\mathcal{M}$.
2. Find a partition $(\mathcal{B}, \mathcal{W}, \mathcal{S})$.
3. Adjust $\mathcal{M}$ and $(\mathcal{B}, \mathcal{W}, \mathcal{S})$ to be compatible.
4. Order $\mathcal{S}$ to end.
5. Restrict matrix and matching to $\mathcal{B}$ and recurse.
6. Restrict matrix and matching to $\mathcal{W}$ and recurse.

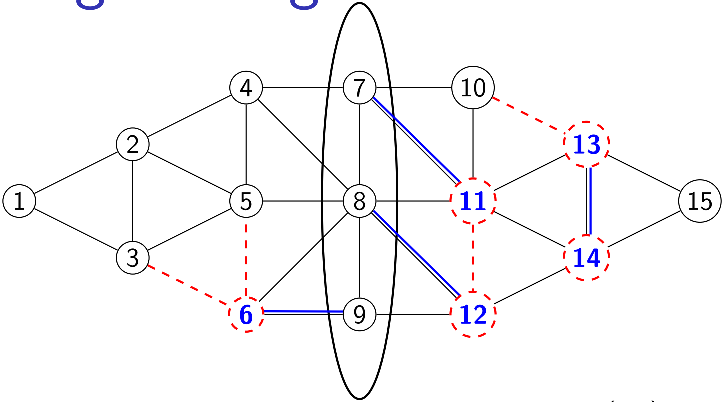
Moving to compatibility

Given incompatible $\mathcal{M}$ and $(\mathcal{B}, \mathcal{W}, \mathcal{S})$:

- Assume $\mathcal{M}$ is compatible on entire matrix.
- There must exist some edge $(i, j) \in \mathcal{M}$ cut by $\mathcal{S}$.



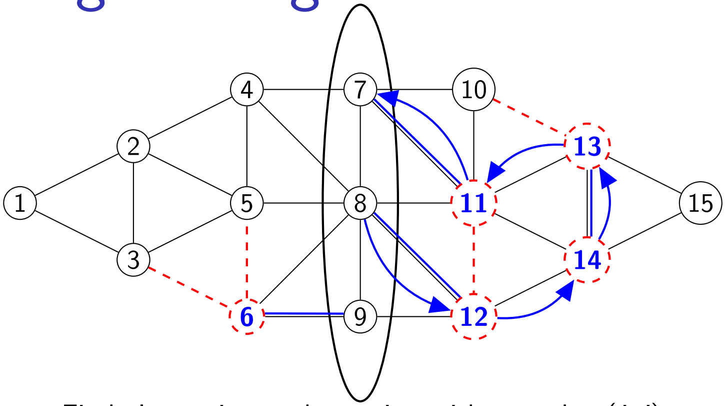
Fixing cut edges



- Find alternating path starting with cut edge (i, j).
- Finish in $\mathcal{S}$ with an edge in $\mathcal{M}$; or
- Finish outside $\mathcal{S}$ with edge not in $\mathcal{M}$.
- WITHOUT intermediate vertex in $\mathcal{S}$.
- Otherwise move j into $\mathcal{S}$.

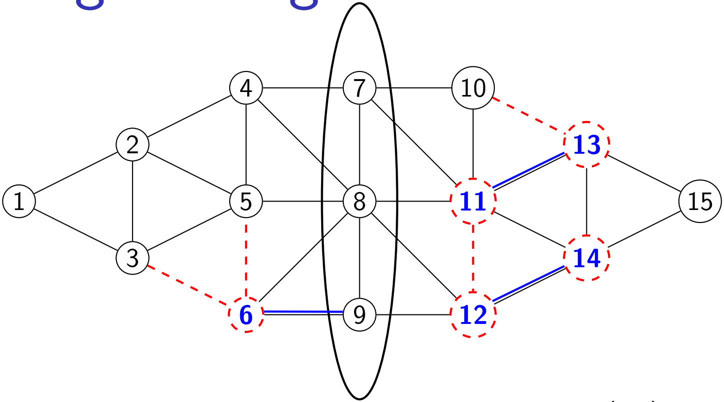
Science & Technology Facilities Council

Fixing cut edges



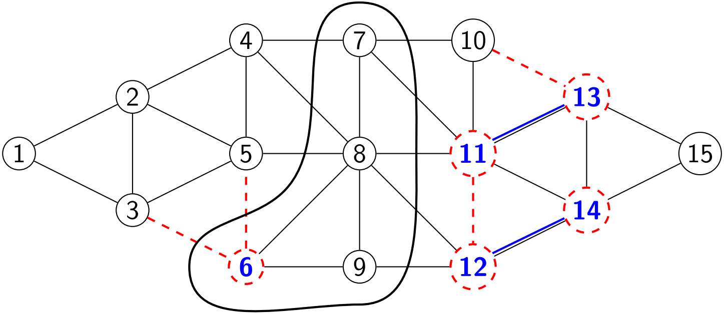
- Find alternating path starting with cut edge (i, j).
- Finish in $\mathcal{S}$ with an edge in $\mathcal{M}$; or
- Finish outside $\mathcal{S}$ with edge not in $\mathcal{M}$.
- WITHOUT intermediate vertex in $\mathcal{S}$.
- Otherwise move j into $\mathcal{S}$.

Fixing cut edges



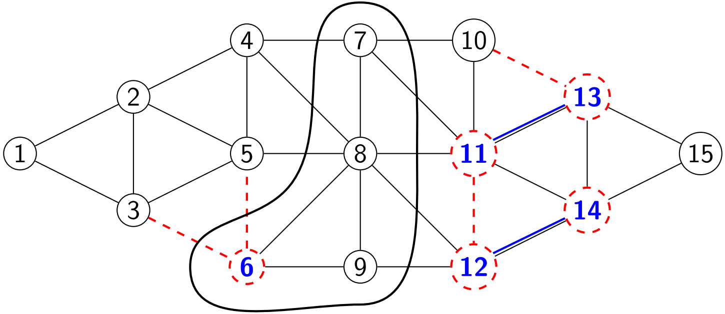
- Find alternating path starting with cut edge (i, j).
- Finish in $\mathcal{S}$ with an edge in $\mathcal{M}$; or
- Finish outside $\mathcal{S}$ with edge not in $\mathcal{M}$.
- WITHOUT intermediate vertex in $\mathcal{S}$.
- Otherwise move j into $\mathcal{S}$.

Fixing cut edges



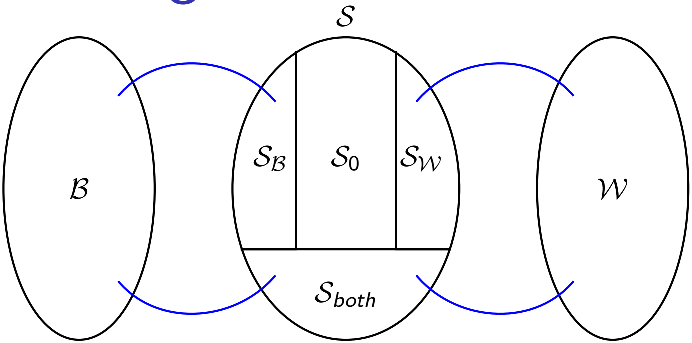
- Find alternating path starting with cut edge (i, j).
- Finish in $\mathcal{S}$ with an edge in $\mathcal{M}$; or
- Finish outside $\mathcal{S}$ with edge not in $\mathcal{M}$.
- WITHOUT intermediate vertex in $\mathcal{S}$.
- Otherwise move j into $\mathcal{S}$.

Fixing cut edges



But now we don't need $9 \in \mathcal{S}$?

Trimming



- Try pulling nodes out of $\mathcal{S}_\mathcal{B}$ into $\mathcal{B}$.
- If they have a small diagonal, find a partner from $\mathcal{S}_\mathcal{B} \cup \mathcal{S}_0$.
- Same for $\mathcal{S}_\mathcal{W}$.
- Update sets and repeat until nothing moves.
- Can allow alternating paths in $\mathcal{B}$ and $\mathcal{W}$ too: complicated

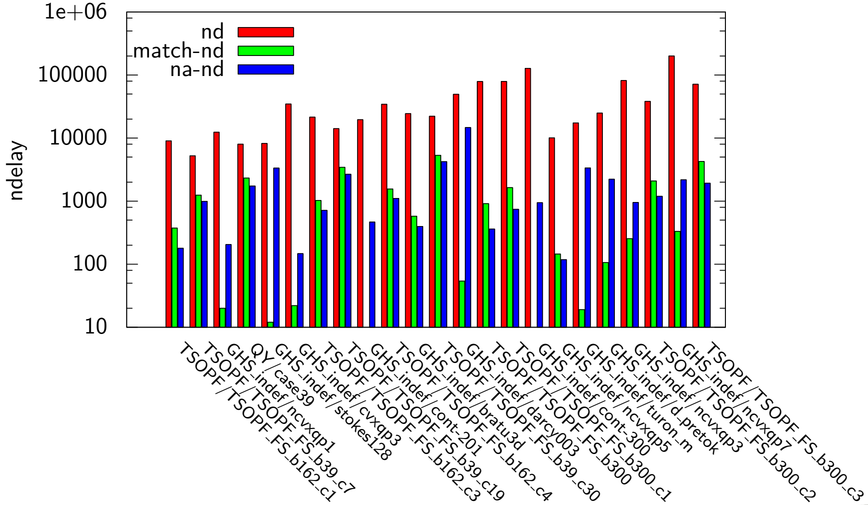
Numerical results

- 23 very difficult problems from UFL (just scaling doesn't work)
- Ordering with in-house Nested Dissection code
- Pre-scaled by symmetrized Hungarian scaling (MC64)
- Factorized with `HSL_MA97`
- Consider:

  ndelay — number of delayed pivots (i.e. how far from predicted order)

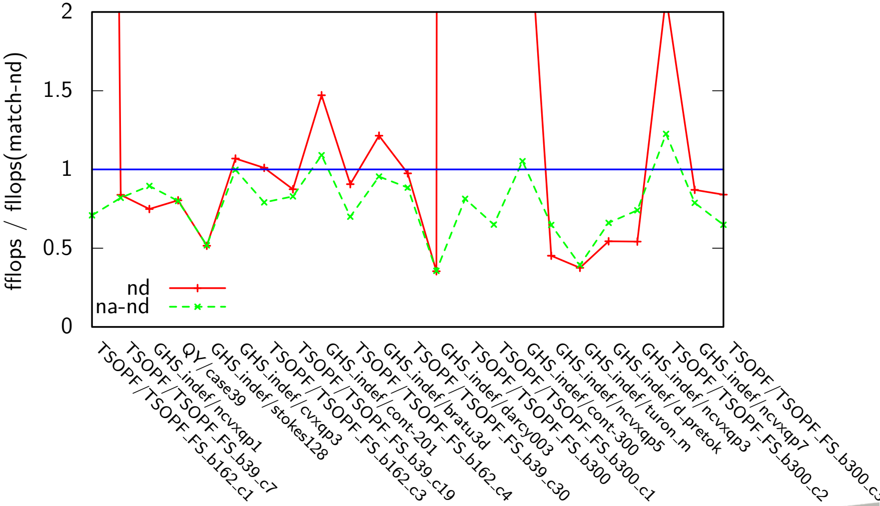
  fflop — number of floating point operation to calculate L

ndelay



Science & Technology Facilities Council

fflops



fflops / fflops(match-nd)
2
1.5
1
0.5
0
nd
na-nd
TSOPF/TSOPF_FS_b162_c1
TSOPF/TSOPF_FS_b39_c7
GHS_indef/ncvxqp1
QY/case39
GHS_indef/stokes128
GHS_indef/cvxqp3
TSOPF/TSOPF_FS_b162_c3
TSOPF/TSOPF_FS_b39_c19
GHS_indef/cont-201
TSOPF/TSOPF_FS_b162_c4
GHS_indef/bratu3d
TSOPF/TSOPF_FS_b39_c30
GHS_indef/darcy003
TSOPF/TSOPF_FS_b300
TSOPF/TSOPF_FS_b300_c1
GHS_indef/cont-300
GHS_indef/ncvxqp5
GHS_indef/turon_m
GHS_indef/d_pretok
GHS_indef/ncvxqp3
TSOPF/TSOPF_FS_b300_c2
GHS_indef/ncvxqp7

For AMD

Modifying Approximate Minimum Degree

- Approach by Duff and Pralet in 2005
- Relaxed version of “compress then order” approach
- Rather than glue pivots together, constrain them
- Only allow small pivots to be ordered after update by **big** off-diagonal.

Reasonable results obtained.
Unclear if used by any solvers in practice.

Summary

- Post processing to a dissection algorithm
- But part of a *nested* dissection algorithm
- Could be put into any ND code with a little work
- Effective at reducing delays without significantly increasing size/work for factors
- More details in technical report: RAL-P-2016-004

Thanks for listening!

Questions?

http://www.numerical.rl.ac.uk/spral

Funded by ESPRC grant EP/M025179/1

Say what now

An appendix?