Black Box Clustering and Parallel $\mathcal{H}$-LU Factorisation

Ronald Kriemann

Max Planck Institute for Mathematics
in the
Sciences Leipzig

Winterschool on $\mathcal{H}$-Matrices
2009
Overview

1 Motivation
2 Graph Partitioning
3 Admissibility
4 Nested Dissection
5 Parallelisation
Motivation
Consider

\[-\Delta u = 0 \quad \text{in } \Omega = [0, 1]^2\]

Using a uniform grid width stepwidth $h$

and standard piecewise linear finite elements with nodal points $x_i, i \in I$, one obtains the stiffness matrix $A$ as
Define the matrix graph \( G(A) = (V_A, E_A) \) of \( A \in \mathbb{R}^{I \times I} \) as

\[
E_A := I, \\
V_A := \{(i, j) \in I \times I : i \neq j \land a_{ij} \neq 0\},
\]

i.e. edges in the graph are defined by the sparsity pattern of the stiffness matrix.

**Remark**

*Non-zero entries \( a_{ij} \) only exist in \( A \) if \( i \) and \( j \) are neighboured.*

For the model problem the matrix graph looks as
Define distance $d_G(i, j)$ between nodes $i, j \in I$ as length of shortest path in $G(A)$. Then, for $i, j \in I$ we have:

$$\|x_i - x_j\|_2 \leq d_G(i, j)h,$$

i.e. distance in $\mathbb{R}^2$ is mapped to distance in $G(A)$.

$$\|x_i - x_j\|_2 = \sqrt{13}h, \quad d_G(i, j) = 5$$

$$\|x_i - x_k\|_2 = \sqrt{5}h, \quad d_G(i, k) = 3$$
Since nodes in $G(A)$ with small distance are geometrically neighboured, one can use graph distance to cluster indices.

Recursively partition sub graphs for cluster tree construction.
Graph Partitioning
Let $A \in \mathbb{R}^{I \times I}$ be a sparse matrix and $G = G(A) = (V_A, E_A)$ the corresponding matrix graph. Furthermore, let
\[
\text{diam}(G) := \max_{i,j \in V_A} d_G(i, j)
\]
\[
\text{diam}_G(V) := \max_{i,j \in V} d_G(i, j), \quad V \subseteq V_A
\]
denote the diameter of the graph and of a sub graph, respectively. For cluster tree construction, one needs a graph partitioning algorithm with the following properties:

- compact sub graphs (small diameter),
- small edge-cut (small number of edges connecting sub graphs).

**Remark**

*No edges between sub graphs corresponds to decoupled clusters and therefore to a block diagonal matrix.*
Algorithm:

1. determine two nodes $i, j \in V_A$ with (almost) maximal distance,
Algorithm:

1. determine two nodes $i, j \in V_A$ with (almost) maximal distance,
2. perform simultaneous BFS from $i$ and $j$ to construct sub clusters:
   - per step, add unvisited neighbours of nodes in sub clusters
Graph Partitioning

Partitioning via Breadth First Search

Algorithm:

1. determine two nodes $i, j \in V_A$ with (almost) maximal distance,
2. perform simultaneous BFS from $i$ and $j$ to construct sub clusters:
   • per step, add unvisited neighbours of nodes in sub clusters

![Graph Partitioning Diagram]

Black Box Clustering and Parallel $\mathcal{H}$-LU Factorisation
Algorithm:

1. determine two nodes $i, j \in V_A$ with (almost) maximal distance,
2. perform simultaneous BFS from $i$ and $j$ to construct sub clusters:
   - per step, add unvisited neighbours of nodes in sub clusters
Algorithm:

1. determine two nodes $i, j \in V_A$ with (almost) maximal distance,
2. perform simultaneous BFS from $i$ and $j$ to construct sub clusters:
   - per step, add unvisited neighbours of nodes in sub clusters
Algorithm:

1. Determine two nodes $i, j \in V_A$ with (almost) maximal distance,
2. Perform simultaneous BFS from $i$ and $j$ to construct sub clusters:
   - Per step, add unvisited neighbours of nodes in sub clusters.
Algorithm:

1. determine two nodes $i, j \in V_A$ with (almost) maximal distance,
2. perform simultaneous BFS from $i$ and $j$ to construct sub clusters:
   - per step, add unvisited neighbours of nodes in sub clusters
3. recurse in sub graphs
BFS based graph partitioning yields compact sub graphs, but not necessarily minimal edge-cut, but can be improved using “Fiduccia-Mattheyses-Algorithm” (see Literature).

#edge-cut: 8 → 6
In graph theory, the graph partitioning problem is defined as:

Given a graph \(G = (V, E)\) a partitioning \(P = \{V_1, V_2\}\), with \(V_1 \cap V_2 = \emptyset\) and \(V = V_1 \cup V_2\), of \(V\) is sought, such that

\[
\#V_1 \sim \#V_2 \quad \text{and} \quad \mathcal{I}_G(V_1, V_2) := \#\{(i, j) \in E : i \in V_1 \land j \in V_2\} = \min
\]

Unfortunately, the graph partitioning problem is NP-hard. But good approximation algorithm exist and are implemented in open source software libraries, e.g.:

- METIS, Scotch (multi-level graph partitioning),
- CHACO (multi-level and spectral graph partitioning).
General black box clustering algorithm:

```python
function blackbox_cluster( G = (V, E) )
    if #V ≤ n_min then
        return cluster t := V;
    else
        \{G_1, G_2\} = partition( G );
        t_1 := blackbox_cluster( G_1 );
        t_2 := blackbox_cluster( G_2 );
        return cluster t := V with S(t) := \{t_1, t_2\};
    end if
end
```

Here, `partition` implements the general graph partitioning algorithm, e.g. from METIS etc..
Graph Partitioning

General Graph Partitioning for Clustering

BFS ($\#I_G = 21$)

BFS+FM ($\#I_G = 11$)

METIS ($\#I_G = 12$)

Scotch ($\#I_G = 12$)
Admissibility
Standard admissibility is defined by

$$\min(\text{diam}(\Omega_t), \text{diam}(\Omega_s)) \leq \eta \text{dist}(\Omega_t, \Omega_s)$$

with support \(\Omega_i\) for each cluster \(i\) and, hence, uses unavailable geometrical data.

**Distance in Graphs**

For \(V_1, V_2 \subset V\), the distance between \(V_1\) and \(V_2\) is defined as

$$\text{dist}_G(V_1, V_2) := \min_{i \in V_1, j \in V_2} \text{dist}_G(i, j)$$

with

$$\text{dist}(i, j) := \text{length of shortest path between } i \text{ and } j \text{ in } G.$$
The simplest admissibility condition for a block cluster \((t, s)\) is defined by

\[
\text{adm}_{\text{weak}}(t, s) := \begin{cases} 
\text{true,} & \text{if } \text{dist}_G(t, s) > 1 \\
\text{false,} & \text{otherwise}
\end{cases},
\]

e.g. if no edge is connecting \(t\) and \(s\) in \(G\).

Weak admissibility is cheap to test and produces effective partitions for \(\mathcal{H}\)-arithmetics (see experiments).
The standard admissibility is defined by

\[
\text{adm}_{\text{std}}(t, s) := \begin{cases} 
\text{true}, & \min(\text{diam}_G(t), \text{diam}_G(s)) \leq \eta \text{dist}_G(t, s) \\
\text{false}, & \text{otherwise}
\end{cases}
\]

e.g. the equivalent of the geometrical admissibility.

Since diameter and distance between clusters in \( G \) costs \( \mathcal{O}(n^2) \), the admissibility is tested as:

- choose node \( i \in t \) and \( j \in t \) with \( \text{dist}_G(i, j) = \max \),
- \( \text{diam}_G(t) \leq 2 \text{dist}_G(i, j) =: \text{diam} \),
- construct surrounding \( t' \) around \( t \) in \( G \) via \( \frac{1}{\eta} \text{diam} \).
- if \( t' \cap s = \emptyset \), \( \text{adm}_{\text{std}}(t, s) = \text{true} \).
The standard admissibility is defined by

\[
\text{adm}_\text{std}(t, s) := \begin{cases} 
\text{true}, & \min(\text{diam}_G(t), \text{diam}_G(s)) \leq \eta \text{dist}_G(t, s) \\
\text{false}, & \text{otherwise}
\end{cases}
\]

e.g. the equivalent of the geometrical admissibility. Since diameter and distance between clusters in $G$ costs $\mathcal{O}(n^2)$, the admissibility is tested as:

- choose node $i \in t$ and $j \in t$ with $\text{dist}_G(i, j) = \max$, 
- $\text{diam}_G(t) \leq 2 \text{dist}_G(i, j) =: \tilde{\text{diam}}$, 
- construct surrounding $t'$ around $t$ in $G$ via $\frac{1}{\eta} \tilde{\text{diam}}$. 
- if $t' \cap s = \emptyset$, $\text{adm}_\text{std}(t, s) = \text{true}$.
The standard admissibility is defined by

\[
\text{adm}_{\text{std}}(t, s) := \begin{cases} 
\text{true}, & \min(\text{diam}_G(t), \text{diam}_G(s)) \leq \eta \text{ dist}_G(t, s) \\
\text{false}, & \text{otherwise}
\end{cases}
\]

e.g. the equivalent of the geometrical admissibility. Since diameter and distance between clusters in \( G \) costs \( \mathcal{O}(n^2) \), the admissibility is tested as:

- choose node \( i \in t \) and \( j \in t \) with \( \text{dist}_G(i, j) = \max \),
- \( \text{diam}_G(t) \leq 2 \text{ dist}_G(i, j) =: \tilde{\text{diam}} \),
- construct surrounding \( t' \) around \( t \) in \( G \) via \( \frac{1}{\eta} \tilde{\text{diam}} \).
- if \( t' \cap s = \emptyset \), \( \text{adm}_{\text{std}}(t, s) = \text{true} \).
The standard admissibility is defined by

\[ \text{adm}_{\text{std}}(t, s) := \begin{cases} 
\text{true,} & \min(\text{diam}_G(t), \text{diam}_G(s)) \leq \eta \text{dist}_G(t, s) \\
\text{false,} & \text{otherwise}
\end{cases} \]

e.g. the equivalent of the geometrical admissibility.

Since diameter and distance between clusters in \( G \) costs \( \mathcal{O}(n^2) \), the admissibility is tested as:

- choose node \( i \in t \) and \( j \in t \) with \( \text{dist}_G(i, j) = \max \),
- \( \text{diam}_G(t) \leq 2 \text{dist}_G(i, j) =: \overline{\text{diam}} \),
- construct surrounding \( t' \) around \( t \) in \( G \) via \( \frac{1}{\eta} \overline{\text{diam}} \).
- if \( t' \cap s = \emptyset \), \( \text{adm}_{\text{std}}(t, s) = \text{true} \).
The $\mathcal{H}$-LU factorisation of the Model Problem:

<table>
<thead>
<tr>
<th>$N$</th>
<th>Geometric</th>
<th></th>
<th></th>
<th></th>
<th>Black Box</th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Time (sec)</td>
<td>Mem (MB)</td>
<td>$\delta$</td>
<td>Time (sec)</td>
<td>Mem (MB)</td>
<td>$\delta$</td>
<td></td>
<td></td>
</tr>
<tr>
<td>253$^2$</td>
<td>3.8</td>
<td>76</td>
<td>$2_{10^{-4}}$</td>
<td>6.6</td>
<td>86</td>
<td>$1_{10^{-4}}$</td>
<td></td>
<td></td>
</tr>
<tr>
<td>358$^2$</td>
<td>10.0</td>
<td>169</td>
<td>$1_{10^{-4}}$</td>
<td>15.7</td>
<td>187</td>
<td>$6_{10^{-5}}$</td>
<td></td>
<td></td>
</tr>
<tr>
<td>511$^2$</td>
<td>24.1</td>
<td>374</td>
<td>$7_{10^{-5}}$</td>
<td>41.7</td>
<td>441</td>
<td>$3_{10^{-5}}$</td>
<td></td>
<td></td>
</tr>
<tr>
<td>729$^2$</td>
<td>61.1</td>
<td>840</td>
<td>$4_{10^{-5}}$</td>
<td>116.1</td>
<td>1020</td>
<td>$1_{10^{-5}}$</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1023$^2$</td>
<td>144.9</td>
<td>1780</td>
<td>$2_{10^{-5}}$</td>
<td>250.8</td>
<td>2110</td>
<td>$8_{10^{-6}}$</td>
<td></td>
<td></td>
</tr>
<tr>
<td>40$^3$</td>
<td>79.1</td>
<td>285</td>
<td>$1_{10^{-3}}$</td>
<td>106.5</td>
<td>292</td>
<td>$1_{10^{-3}}$</td>
<td></td>
<td></td>
</tr>
<tr>
<td>51$^3$</td>
<td>194.5</td>
<td>634</td>
<td>$1_{10^{-3}}$</td>
<td>326.1</td>
<td>763</td>
<td>$7_{10^{-4}}$</td>
<td></td>
<td></td>
</tr>
<tr>
<td>64$^3$</td>
<td>520.3</td>
<td>1400</td>
<td>$1_{10^{-3}}$</td>
<td>896.4</td>
<td>1760</td>
<td>$4_{10^{-4}}$</td>
<td></td>
<td></td>
</tr>
<tr>
<td>81$^3$</td>
<td>1440.0</td>
<td>3560</td>
<td>$5_{10^{-4}}$</td>
<td>2444.8</td>
<td>4330</td>
<td>$2_{10^{-4}}$</td>
<td></td>
<td></td>
</tr>
<tr>
<td>102$^3$</td>
<td>3875.5</td>
<td>8070</td>
<td>$4_{10^{-4}}$</td>
<td>6575.7</td>
<td>9940</td>
<td>$2_{10^{-4}}$</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Accuracy of $\mathcal{H}$-arithmetics defined by $\delta$ and chosen such that

$$\| I - (L_{\mathcal{H}}U_{\mathcal{H}})^{-1}A \|_2 \leq 10^{-4}$$
Nested Dissection
In nested dissection the two constructed sub graphs of a partition have to be separated via a vertex separator.

Matrix graph:

 Especially suited are graph partitioning algorithms yielding minimal edge-cut, therefore, maximizing the size of the zero off-diagonal matrix blocks.
Let $V_1, V_2 \subset V$, $V_1 \cap V_2 = \emptyset$ be a partition of $G = (V, E)$ and let $\mathcal{E} = \{(i, j) \in E : i \in V_1, j \in V_2\}$ be the edge-cut of $V_1, V_2$.

A vertex separator $\mathcal{V}$ for $V_1, V_2$ can be obtained by computing a vertex cover of $\mathcal{E}$, i.e. a set of nodes incident to all edges in $\mathcal{E}$.

Algorithm:

Loop until $\mathcal{E} \neq \emptyset$:
Let $V_1, V_2 \subset V, V_1 \cap V_2 = \emptyset$ be a partition of $G = (V, E)$ and let $\mathcal{E} = \{(i, j) \in E : i \in V_1, j \in V_2\}$ be the edge-cut of $V_1, V_2$.

A vertex separator $\mathcal{V}$ for $V_1, V_2$ can be obtained by computing a vertex cover of $\mathcal{E}$, i.e. a set of nodes incident to all edges in $\mathcal{E}$.

Algorithm:

Loop until $\mathcal{E} \neq \emptyset$:

- choose $(i, j) \in \mathcal{E}$;
Let $V_1, V_2 \subset V, V_1 \cap V_2 = \emptyset$ be a partition of $G = (V, E)$ and let $\mathcal{E} = \{(i, j) \in E : i \in V_1, j \in V_2\}$ be the edge-cut of $V_1, V_2$.

A vertex separator $\mathcal{V}$ for $V_1, V_2$ can be obtained by computing a vertex cover of $\mathcal{E}$, i.e. a set of nodes incident to all edges in $\mathcal{E}$.

Algorithm:

Loop until $\mathcal{E} \neq \emptyset$:

- choose $(i, j) \in \mathcal{E}$;
- choose $v \in \{i, j\}$ such that $v \in V'$ with $\#V' = \max V_i$;
Let $V_1, V_2 \subset V, V_1 \cap V_2 = \emptyset$ be a partition of $G = (V, E)$ and let $\mathcal{E} = \{(i, j) \in E : i \in V_1, j \in V_2\}$ be the edge-cut of $V_1, V_2$.

A vertex separator $\mathcal{V}$ for $V_1, V_2$ can be obtained by computing a vertex cover of $\mathcal{E}$, i.e. a set of nodes incident to all edges in $\mathcal{E}$.

Algorithm:

Loop until $\mathcal{E} \neq \emptyset$:

- choose $(i, j) \in \mathcal{E}$;
- choose $v \in \{i, j\}$ such that $v \in V'$ with $\#V' = \max V_i$;
- $\mathcal{V} := \mathcal{V} \cup \{v\};$ $V' := V' \setminus \{v\}$;
- $\mathcal{E} := \mathcal{E} \setminus \{(i, j') \in \mathcal{E}\}$;
Let $V_1, V_2 \subset V, V_1 \cap V_2 = \emptyset$ be a partition of $G = (V, E)$ and let $\mathcal{E} = \{(i, j) \in E : i \in V_1, j \in V_2\}$ be the edge-cut of $V_1, V_2$.

A vertex separator $\mathcal{V}$ for $V_1, V_2$ can be obtained by computing a vertex cover of $\mathcal{E}$, i.e. a set of nodes incident to all edges in $\mathcal{E}$.

Algorithm:

Loop until $\mathcal{E} \neq \emptyset$:

- choose $(i, j) \in \mathcal{E}$;
- choose $v \in \{i, j\}$ such that $v \in V'$ with $\#V' = \max V_i$;
- $\mathcal{V} := \mathcal{V} \cup \{v\}; V' := V' \setminus \{v\}$;
- $\mathcal{E} := \mathcal{E} \setminus \{(i, j') \in \mathcal{E}\}$;
Let \( V_1, V_2 \subset V, V_1 \cap V_2 = \emptyset \) be a partition of \( G = (V, E) \) and let \( \mathcal{E} = \{(i, j) \in E : i \in V_1, j \in V_2\} \) be the edge-cut of \( V_1, V_2 \).

A vertex separator \( \mathcal{V} \) for \( V_1, V_2 \) can be obtained by computing a vertex cover of \( \mathcal{E} \), i.e. a set of nodes incident to all edges in \( \mathcal{E} \).

Algorithm:

Loop until \( \mathcal{E} \neq \emptyset \):

- choose \((i, j) \in \mathcal{E}\);
- choose \(v \in \{i, j\}\) such that \(v \in V'\) with \(\#V' = \max V_i\);
- \(\mathcal{V} := \mathcal{V} \cup \{v\}\); \(V' := V' \setminus \{v\}\);
- \(\mathcal{E} := \mathcal{E} \setminus \{(i, j') \in \mathcal{E}\}\);
Let $V_1, V_2 \subset V, V_1 \cap V_2 = \emptyset$ be a partition of $G = (V, E)$ and let $\mathcal{E} = \{(i, j) \in E : i \in V_1, j \in V_2\}$ be the edge-cut of $V_1, V_2$.

A vertex separator $\mathcal{V}$ for $V_1, V_2$ can be obtained by computing a vertex cover of $\mathcal{E}$, i.e. a set of nodes incident to all edges in $\mathcal{E}$.

Algorithm:

Loop until $\mathcal{E} \neq \emptyset$:

- choose $(i, j) \in \mathcal{E}$;
- choose $v \in \{i, j\}$ such that $v \in V'$ with $\#V' = \max V_i$;
- $\mathcal{V} := \mathcal{V} \cup \{v\}; V' := V' \setminus \{v\}$;
- $\mathcal{E} := \mathcal{E} \setminus \{(i, j') \in \mathcal{E}\}$;
Let $V_1, V_2 \subset V, V_1 \cap V_2 = \emptyset$ be a partition of $G = (V, E)$ and let $\mathcal{E} = \{(i, j) \in E : i \in V_1, j \in V_2\}$ be the edge-cut of $V_1, V_2$.

A vertex separator $\mathcal{V}$ for $V_1, V_2$ can be obtained by computing a vertex cover of $\mathcal{E}$, i.e. a set of nodes incident to all edges in $\mathcal{E}$.

Algorithm:

Loop until $\mathcal{E} \neq \emptyset$:

• choose $(i, j) \in \mathcal{E}$;
• choose $v \in \{i, j\}$ such that $v \in V'$ with $\#V' = \max V_i$;
• $\mathcal{V} := \mathcal{V} \cup \{v\}$; $V' := V' \setminus \{v\}$;
• $\mathcal{E} := \mathcal{E} \setminus \{(i, j') \in \mathcal{E}\}$;
In contrast to classical nested dissection, $\mathcal{H}$-matrices also use a cluster tree for indices in the vertex separator. Hence, further subdivision is necessary.

Problem: restricting $G$ to nodes in $\mathcal{V}$ might remove important edges, e.g.

Therefore, graph partitioning for vertex separator is performed in sub graph induced by $V_1, V_2$ and $\mathcal{V}$. 
Modify BFS based algorithm for vertex separator:

For further subdivision, only consider visited nodes to reduce complexity.

Remark
Still open: efficient construction of minimal surrounding graph for subdivision of vertex separator.

Unfortunately, no graph partitioning packages, e.g. METIS, Scotch, etc., applicable to vertex separator partitioning.
Nested Dissection

Modify BFS based algorithm for vertex separator:

- choose start nodes for BFS in $\mathcal{V}$,
Modify BFS based algorithm for vertex separator:

- choose start nodes for BFS in $\mathcal{V}$,
- perform BFS step only for smaller node set to achieve balance,

![Graph with nodes colored differently.](image)
Modify BFS based algorithm for vertex separator:

- choose start nodes for BFS in $\mathcal{V}$,
- perform BFS step only for smaller node set to achieve balance,
Modify BFS based algorithm for vertex separator:

- choose start nodes for BFS in $\mathcal{V}$,
- perform BFS step only for smaller node set to achieve balance,
Modify BFS based algorithm for vertex separator:

- choose start nodes for BFS in \( \mathcal{V} \),
- perform BFS step only for smaller node set to achieve balance,
- stop BFS iteration when all nodes in \( \mathcal{V} \) have been visited.
Modify BFS based algorithm for vertex separator:

- choose start nodes for BFS in $\mathcal{V}$,
- perform BFS step only for smaller node set to achieve balance,
- stop BFS iteration when all nodes in $\mathcal{V}$ have been visited.

For further subdivision, only consider visited nodes to reduce complexity.

**Remark**

*Still open: efficient construction of minimal surrounding graph for subdivision of vertex separator.*

Unfortunately, no graph partitioning packages, e.g. METIS, Scotch, etc., applicable to vertex separator partitioning.
\( \mathcal{H} \)-LU factorisation of Model Problem using nested dissection:

<table>
<thead>
<tr>
<th>( N )</th>
<th>Geometric</th>
<th></th>
<th></th>
<th>Black Box</th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Time (sec)</td>
<td>Mem (MB)</td>
<td>( \delta )</td>
<td>Time (sec)</td>
<td>Mem (MB)</td>
<td>( \delta )</td>
</tr>
<tr>
<td>253( ^2 )</td>
<td>0.9</td>
<td>51</td>
<td>1(_{10^{-3}})</td>
<td>1.3</td>
<td>47</td>
<td>3(_{10^{-5}})</td>
</tr>
<tr>
<td>358( ^2 )</td>
<td>1.9</td>
<td>86</td>
<td>4(_{10^{-4}})</td>
<td>2.9</td>
<td>94</td>
<td>2(_{10^{-5}})</td>
</tr>
<tr>
<td>511( ^2 )</td>
<td>4.5</td>
<td>212</td>
<td>2(_{10^{-4}})</td>
<td>6.5</td>
<td>198</td>
<td>9(_{10^{-6}})</td>
</tr>
<tr>
<td>729( ^2 )</td>
<td>9.6</td>
<td>371</td>
<td>1(_{10^{-4}})</td>
<td>15.0</td>
<td>402</td>
<td>5(_{10^{-6}})</td>
</tr>
<tr>
<td>1023( ^2 )</td>
<td>20.2</td>
<td>878</td>
<td>6(_{10^{-5}})</td>
<td>31.6</td>
<td>819</td>
<td>2(_{10^{-6}})</td>
</tr>
<tr>
<td>40( ^3 )</td>
<td>12.6</td>
<td>99</td>
<td>1(_{10^{-2}})</td>
<td>32.7</td>
<td>135</td>
<td>3(_{10^{-4}})</td>
</tr>
<tr>
<td>51( ^3 )</td>
<td>46.9</td>
<td>300</td>
<td>3(_{10^{-3}})</td>
<td>97.6</td>
<td>323</td>
<td>2(_{10^{-4}})</td>
</tr>
<tr>
<td>64( ^3 )</td>
<td>117.4</td>
<td>592</td>
<td>2(_{10^{-3}})</td>
<td>289.1</td>
<td>719</td>
<td>1(_{10^{-4}})</td>
</tr>
<tr>
<td>81( ^3 )</td>
<td>269.8</td>
<td>1410</td>
<td>1(_{10^{-3}})</td>
<td>804.3</td>
<td>1570</td>
<td>8(_{10^{-5}})</td>
</tr>
<tr>
<td>102( ^3 )</td>
<td>752.3</td>
<td>3020</td>
<td>1(_{10^{-3}})</td>
<td>1907.3</td>
<td>3370</td>
<td>6(_{10^{-5}})</td>
</tr>
</tbody>
</table>

Again, \( \mathcal{H} \)-accuracy \( \delta \) chosen such that

\[
\| I - (L_{\mathcal{H}}U_{\mathcal{H}})^{-1}A \|_2 \leq 10^{-4}
\]
Comparison of algebraic $\mathcal{H}$-LU factorisation with direct solvers for

$$-\Delta u + \lambda u = f \quad \text{in } \Omega = [0, 1]^2$$
Comparison of algebraic $\mathcal{H}$-LU factorisation with direct solvers for

$$-\Delta u + \lambda u = f \quad \text{in } \Omega = [0, 1]^3$$

![Graph comparing time for setup in seconds vs. number of unknowns for different solvers.](image-url)
Graph $G$ is partitioned into $p$ sub graphs decoupled by single vertex separator:
Graph $G$ is partitioned into $p$ sub graphs decoupled by single vertex separator:

Parallel $\mathcal{H}$-LU factorisation on processor $i$:

1. factorise $A_{ii} = L_{ii}U_{ii}$, (seq. LU Fac.)
2. solve $A_{ip} = L_{ii}U_{ip}$ and $A_{pi} = L_{pi}U_{ii}$, (seq. Algo.)
3. compute and exchange $L_{pi}U_{ip}$, (log $p$ steps)
4. update $A_{pp} = A_{pp} - \sum_i L_{pi}U_{ip}$, (seq. Matrix Mult.)
5. factorise $A_{pp} = L_{pp}L_{pp}$, (seq. LU Fac.)
For the complexity of the parallel $\mathcal{H}$-LU factorisation in the model problem, we assume

- equal load of order $n/p$ per sub graph,
- sizes $n_V$ of vertex separator is of optimal order $p^{1/d}n^{(d−1)/d}$

Then one obtains:

$$O\left( \frac{n \log^2 n}{p} + p^{1/d}n^{(d−1)/d} \log^2 n \log p \right)$$

The speedup is limited by size of vertex separator, which increases with $p$. 

![Graph showing speedup vs. number of processors](image)
Graph $G$ is hierarchically partitioned with local vertex separators:

1. choose $i \in \{0, 1\}$ such that $A_{ii}$ is on local processor;
2. factorise $A_{ii} = L_{ii} U_{ii}$, (Recursion)
3. solve $A_{i2} = L_{ii} U_{i2}$ and $A_{2i} = L_{2i} U_{ii}$, (parallel Matrix Mult.)
4. compute and exchange $L_{2i} U_{i2}$,
5. update $A_{22} = A_{22} - \sum_i L_{2i} U_{i2}$, (seq. Matrix Mult.)
6. factorise $A_{22} = L_{22} L_{22}$, (seq. LU Fac.)
Parallelisation | Nested Dissection

Graph $G$ is hierarchically partitioned with local vertex separators:

Parallel $\mathcal{H}$-LU factorisation is based on algorithm for direct domain decomposition with $p = 2$:

1. choose $i \in \{0, 1\}$ such that $A_{ii}$ is on local processor;
2. factorise $A_{ii} = L_{ii}U_{ii}$, \hfill (Recursion)
3. solve $A_{i2} = L_{ii}U_{i2}$ and $A_{2i} = L_{2i}U_{ii}$, \hfill (parallel Matrix Mult.)
4. compute and exchange $L_{2i}U_{i2}$,
5. update $A_{22} = A_{22} - \sum_i L_{2i}U_{i2}$, \hfill (seq. Matrix Mult.)
6. factorise $A_{22} = L_{22}L_{22}$, \hfill (seq. LU Fac.)
Data distribution on to $\mathcal{P} := \{1, \ldots, p\}$ processors follows hierarchical decomposition during nested dissection:

- on level 0, all processors handle the matrix,

\[ \{1, \ldots, 4\} \]
Data distribution on to $\mathcal{P} := \{1, \ldots, p\}$ processors follows hierarchical decomposition during nested dissection:

- on level 0, all processors handle the matrix,
- on level 1, $\mathcal{P}$ is split into two halves according to graph bisection,
Data distribution on to $\mathcal{P} := \{1, \ldots, p\}$ processors follows hierarchical decomposition during nested dissection:

- on level 0, all processors handle the matrix,
- on level 1, $\mathcal{P}$ is split into two halves according to graph bisection,
- recursively divide the processor set.
Data distribution on to $\mathcal{P} := \{1, \ldots, p\}$ processors follows hierarchical decomposition during nested dissection:

- on level 0, all processors handle the matrix,
- on level 1, $\mathcal{P}$ is split into two halves according to graph bisection,
- recursively divide the processor set.

For processor $i$:
- only handle those matrices with processor set $\mathcal{P}$, if $i \in \mathcal{P}$,
- exchange data only with other processors in $\mathcal{P}$.
For the complexity of the parallel $\mathcal{H}$-LU factorisation in the model problem, we again assume

- equal load of order $n/p$ per sub graph,
- minimal order w.r.t. dimension $d$ of local vertex separator

Then one obtains:

$$O\left( \frac{n \log^2 n}{p} + n^{(d-1)/d} \log^2 n \log p \right)$$

The speedup is now limited by size $O\left( n^{(d-1)/d} \right)$ of first vertex separator and much less dependent on $p$. 

![Graph showing speedup vs. number of processors for different values of n.]
L. Grasedyck, R. Kriemann and S. Le Borne,
*Domain Decomposition Based $\mathcal{H}$-LU Preconditioning*,
to appear in “Numerische Mathematik”.

L. Grasedyck, R. Kriemann and S. Le Borne,
*Parallel Black Box $\mathcal{H}$-LU Preconditioning for Elliptic Boundary Value Problems*,

G. Karypis and V. Kumar
*A fast and high quality multilevel scheme for partitioning irregular graphs*,

C.M. Fiduccia and R.M. Mattheyses,
*A linear-time heuristic for improving network partitions*,