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

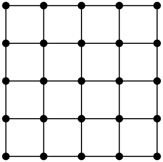
Overview

Motivation

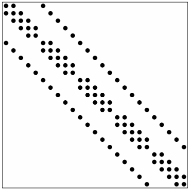
Consider

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

Using a uniform grid width stepwidth h



and standard piecewiese linear finite elements with nodal points $x_i, i \in I$, one obtains the stiffness matrix A as



Motivation | Matrixgraph

Define the matrix graph $G(A) = (V_A, E_A)$ of $A \in \mathbb{R}^{I \times I}$ as

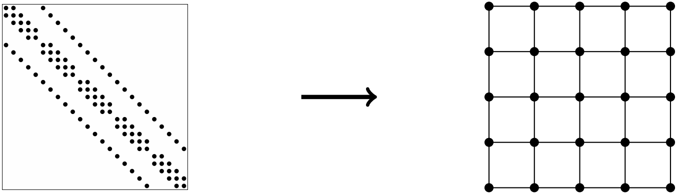
$$E_A := I,$$
$$V_A := \{(i,j) \in I \times I \ : \ i \neq j \wedge a_{ij} \neq 0\},$$

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.

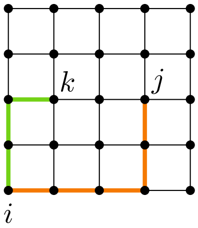
For the model problem the matrix graph looks as



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:

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

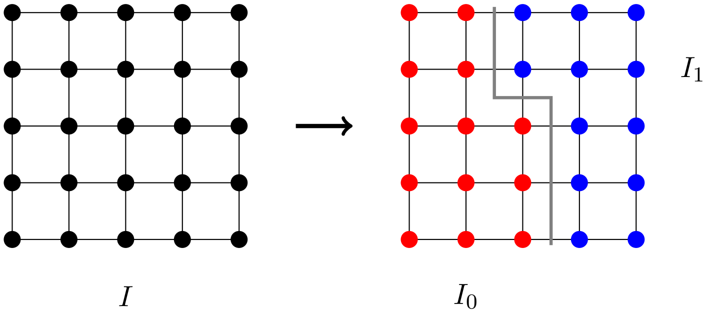
i.e. distance in $\mathbb{R}^2$ is mapped to distance in $G(A)$.



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

$$\|x_i - x_k\|_2 = \sqrt{5}h, \qquad d_G(i,k) = 3$$

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

$$\operatorname{diam}(G) := \max_{i,j \in V_A} d_G(i,j)$$

$$\operatorname{diam}_G(V) := \max_{i,j \in V} d_G(i,j), \quad V \subseteq V_A$$

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:

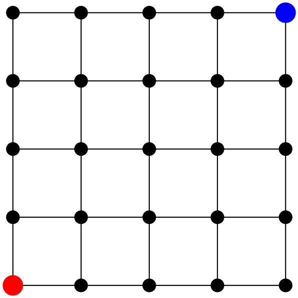
- compact sub graphs (small diameter),
- small edge-cut (small number of edges connecting sub graphs).

Remark

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

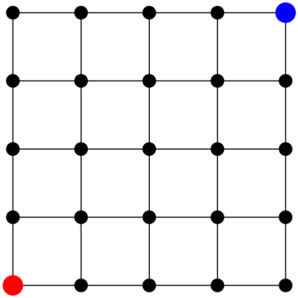
Algorithm:

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



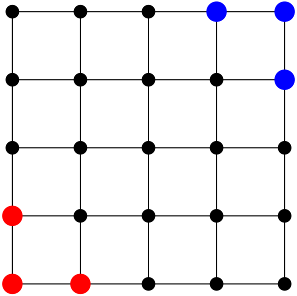
Algorithm:

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



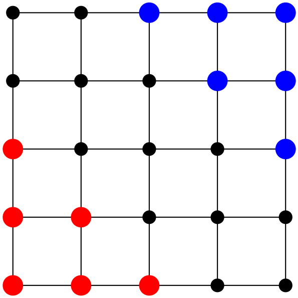
Algorithm:

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



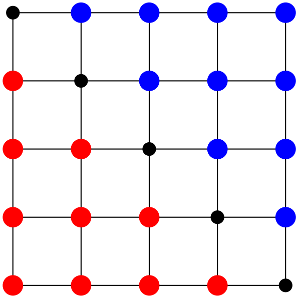
Algorithm:

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



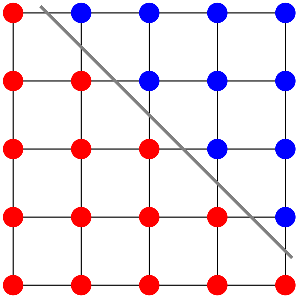
Algorithm:

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



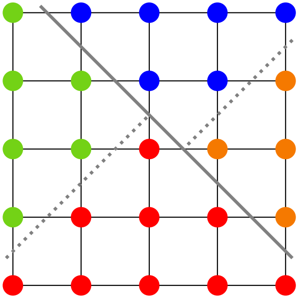
Algorithm:

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

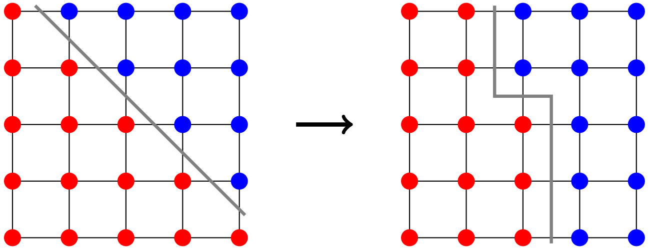


Algorithm:

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 neccessarily 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*

$$\#V_1 \sim \#V_2 \quad \text{and}$$

$$\mathcal{I}_G(V_1, V_2) := \#\{(i,j) \in E \ : \ i \in V_1 \wedge j \in V_2\} = \min$$

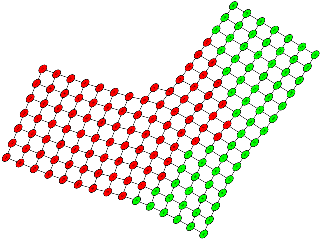
Unfortunately, the graph partitioning problem is NP-hard. But good approximation algorithm exist and are implemented in open source software libraries, e.g.:

- METIS, Scotch (multi-level graph partitioning),
- CHACO (multi-level and spectral graph partitioning).

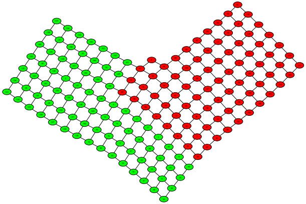
General black box clustering algorithm:

function blackbox_cluster($G = (V, E)$)
  if $\#V \leq n_{\min}$ **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 $\mathcal{S}(t) := \{t_1, t_2\}$;
  end if
end

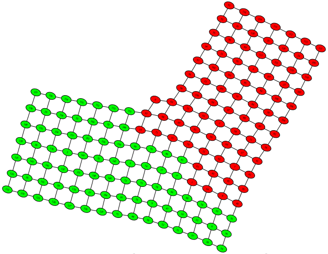
Here, `partition` implements the general graph partitioning algorithm, e.g. from METIS etc..



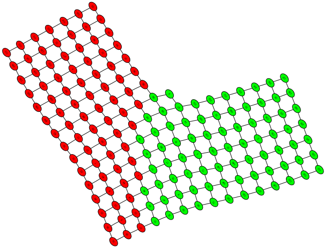
BFS ($\#\mathcal{I}_G = 21$)



BFS+FM ($\#\mathcal{I}_G = 11$)



METIS ($\#\mathcal{I}_G = 12$)



Scotch ($\#\mathcal{I}_G = 12$)

Admissibility

Standard admissibility is defined by

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

with support Ω_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

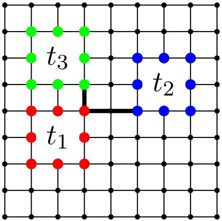
$$\operatorname{dist}_G(V_1, V_2) := \min_{i \in V_1, j \in V_2} \operatorname{dist}_G(i, j) \qquad \text{with}$$

$$\operatorname{dist}(i, j) := \text{length of shortest path between } i \text{ and } j \text{ in } G.$$

The simplest admissibility condition for a block cluster (t, s) is defined by

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

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



$$\mathrm{adm}_{\mathsf{weak}}(t_1, t_2) = \text{true}$$
$$\mathrm{adm}_{\mathsf{weak}}(t_1, t_3) = \text{false}$$

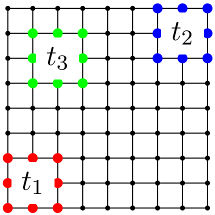
Weak admissibility is cheap to test and produces effective partitions for $\mathcal{H}$-arithmetics (see experiments).

The standard admissibility is defined by

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

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

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

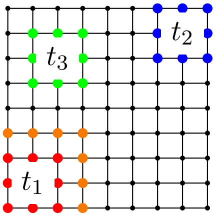


The standard admissibility is defined by

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

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

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

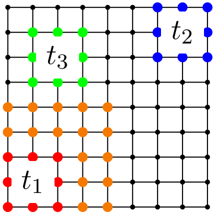


The standard admissibility is defined by

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

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

- choose node $i \in t$ and $j \in t$ with $\mathrm{dist}_G(i,j) = \max$,
- $\mathrm{diam}_G(t) \leq 2\,\mathrm{dist}_G(i,j) =: \widetilde{\mathrm{diam}}$,
- construct surrounding t' around t in G via $\frac{1}{\eta}\widetilde{\mathrm{diam}}$.
- if $t' \cap s = \emptyset, \mathrm{adm}_{\mathsf{std}}(t,s) = $ true.

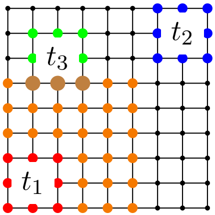


The standard admissibility is defined by

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

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

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



$\mathcal{H}$-LU factorisation of Model Problem:

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

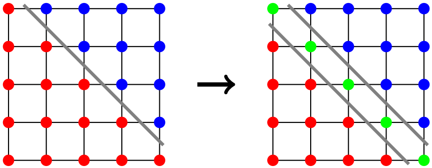
Accuracy of $\mathcal{H}$-arithmetics defined by δ and chosen such that

$$\|I - (L_{\mathcal{H}} U_{\mathcal{H}})^{-1} A\|_2 \leq 10^{-4}$$

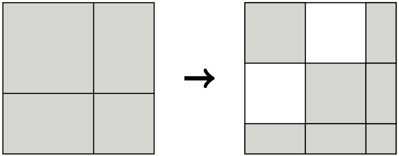
Nested Dissection

In nested dissection the two constructed sub graphs of a partition have to be separated via a vertex separator.

Matrix graph:



Matrix:



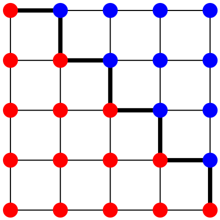
Especially suited are graph partitioning algorithms yielding minimal edge-cut, therefore, maximizing the size of the zero off-diagonal matrix blocks.

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



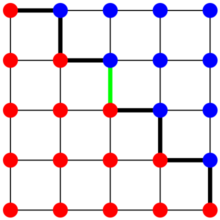
Let $V_1, V_2 \subset V, V_1 \cap V_2 = \emptyset$ be a partition of $G = (V, E)$ and let $\mathcal{E} = \{(i,j) \in E \ : \ i \in V_1, j \in V_2\}$ be the edge-cut of V_1, V_2.

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 $(i,j) \in \mathcal{E}$;



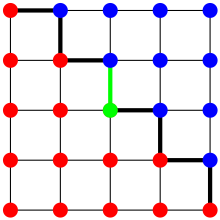
Let $V_1, V_2 \subset V, V_1 \cap V_2 = \emptyset$ be a partition of $G = (V, E)$ and let $\mathcal{E} = \{(i,j) \in E \ : \ i \in V_1, j \in V_2\}$ be the edge-cut of V_1, V_2.

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 $\#V' = \max V_i$;



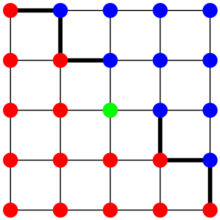
Let $V_1, V_2 \subset V, V_1 \cap V_2 = \emptyset$ be a partition of $G = (V, E)$ and let $\mathcal{E} = \{(i,j) \in E \ : \ i \in V_1, j \in V_2\}$ be the edge-cut of V_1, V_2.

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 $\#V' = \max V_i$;
- $\mathcal{V} := \mathcal{V} \cup \{v\}$; $V' := V' \setminus \{v\}$;
- $\mathcal{E} := \mathcal{E} \setminus \{(i,j') \in \mathcal{E}\}$;



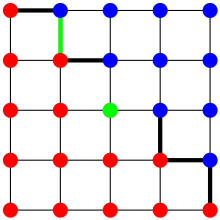
Let $V_1, V_2 \subset V, V_1 \cap V_2 = \emptyset$ be a partition of $G = (V, E)$ and let $\mathcal{E} = \{(i,j) \in E \;:\; i \in V_1, j \in V_2\}$ be the edge-cut of V_1, V_2.

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 $\#V' = \max V_i$;
- $\mathcal{V} := \mathcal{V} \cup \{v\}$; $V' := V' \setminus \{v\}$;
- $\mathcal{E} := \mathcal{E} \setminus \{(i,j') \in \mathcal{E}\}$;



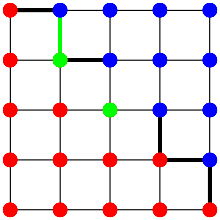
Let $V_1, V_2 \subset V, V_1 \cap V_2 = \emptyset$ be a partition of $G = (V, E)$ and let $\mathcal{E} = \{(i,j) \in E \ : \ i \in V_1, j \in V_2\}$ be the edge-cut of V_1, V_2.

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 $\#V' = \max V_i$;
- $\mathcal{V} := \mathcal{V} \cup \{v\}$; $V' := V' \setminus \{v\}$;
- $\mathcal{E} := \mathcal{E} \setminus \{(i,j') \in \mathcal{E}\}$;



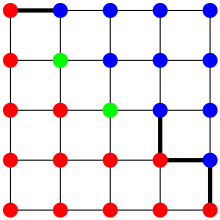
Let $V_1, V_2 \subset V, V_1 \cap V_2 = \emptyset$ be a partition of $G = (V, E)$ and let $\mathcal{E} = \{(i, j) \in E \ : \ i \in V_1, j \in V_2\}$ be the edge-cut of V_1, V_2.

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 $\#V' = \max V_i$;
- $\mathcal{V} := \mathcal{V} \cup \{v\}$; $V' := V' \setminus \{v\}$;
- $\mathcal{E} := \mathcal{E} \setminus \{(i, j') \in \mathcal{E}\}$;



Nested Dissection

Constructing the Vertex Separator

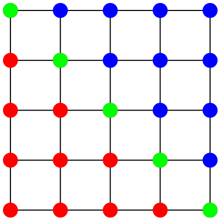
Let $V_1, V_2 \subset V, V_1 \cap V_2 = \emptyset$ be a partition of $G = (V, E)$ and let $\mathcal{E} = \{(i,j) \in E \ : \ i \in V_1, j \in V_2\}$ be the edge-cut of V_1, V_2.

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 $\mathcal{E}$.

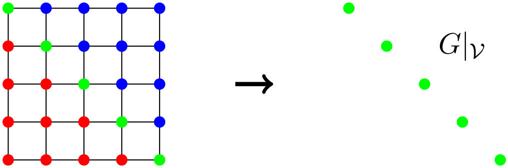
Algorithm:

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

- choose $(i,j) \in \mathcal{E}$;
- choose $v \in \{i,j\}$ such that $v \in V'$ with $\#V' = \max V_i$;
- $\mathcal{V} := \mathcal{V} \cup \{v\}$; $V' := V' \setminus \{v\}$;
- $\mathcal{E} := \mathcal{E} \setminus \{(i,j') \in \mathcal{E}\}$;

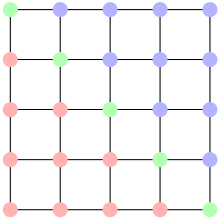


In contrast to classical nested dissection, $\mathcal{H}$-matrices also use a cluster tree for indices in the vertex seperator. Hence, further subdivision is necessary.
Problem: restricting G to nodes in $\mathcal{V}$ might remove important edges, e.g.



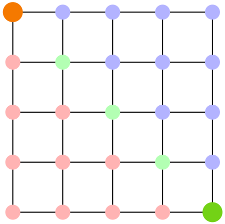
Therefore, graph partitioning for vertex separator is performed in sub graph induced by V_1, V_2 and $\mathcal{V}$.

Modify BFS based algorithm for vertex separator:



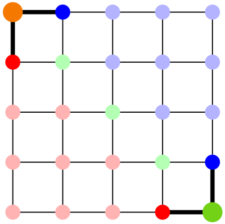
Modify BFS based algorithm for vertex separator:

- choose start nodes for BFS in $\mathcal{V}$,



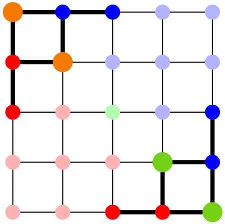
Modify BFS based algorithm for vertex separator:

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



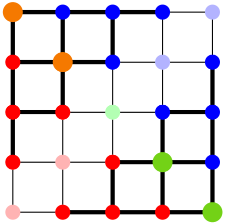
Modify BFS based algorithm for vertex separator:

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



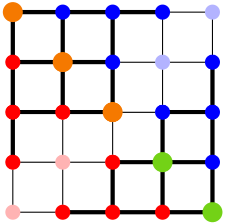
Modify BFS based algorithm for vertex separator:

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



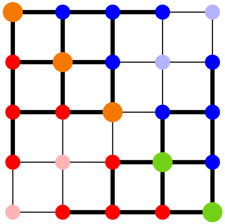
Modify BFS based algorithm for vertex separator:

- 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:

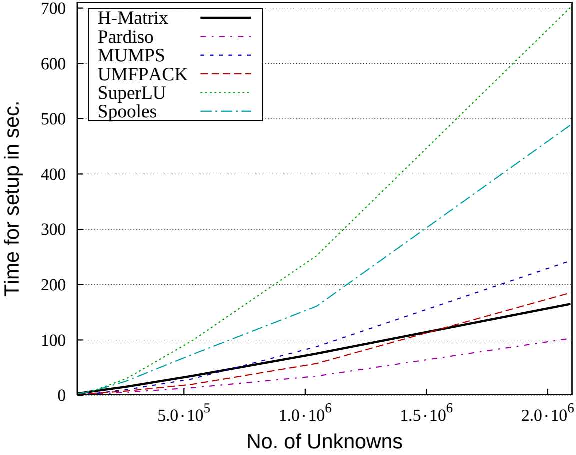
N	Geometric			Black Box		
	Time (sec)	Mem (MB)	δ	Time (sec)	Mem (MB)	δ
253^2	0.9	51	1_{10}-3	1.3	47	3_{10}-5
358^2	1.9	86	4_{10}-4	2.9	94	2_{10}-5
511^2	4.5	212	2_{10}-4	6.5	198	9_{10}-6
729^2	9.6	371	1_{10}-4	15.0	402	5_{10}-6
1023^2	20.2	878	6_{10}-5	31.6	819	2_{10}-6
40^3	12.6	99	1_{10}-2	32.7	135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

Again, $\mathcal{H}$-accuracy δ chosen such that

$$\|I - (L_{\mathcal{H}} U_{\mathcal{H}})^{-1} A\|_2 \leq 10^{-4}$$

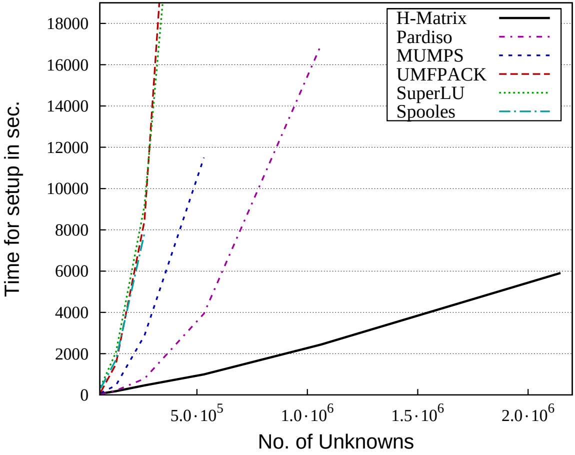
Comparison of algebraic $\mathcal{H}$-LU factorisation with direct solvers for

$$-\Delta u + \lambda u = f \qquad \text{in } \Omega = [0,1]^2$$



Comparison of algebraic $\mathcal{H}$-LU factorisation with direct solvers for

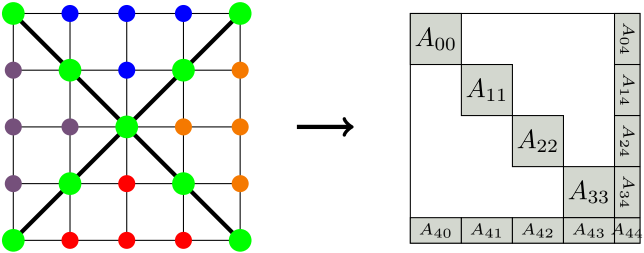
$$-\Delta u + \lambda u = f \qquad \text{in } \Omega = [0,1]^3$$



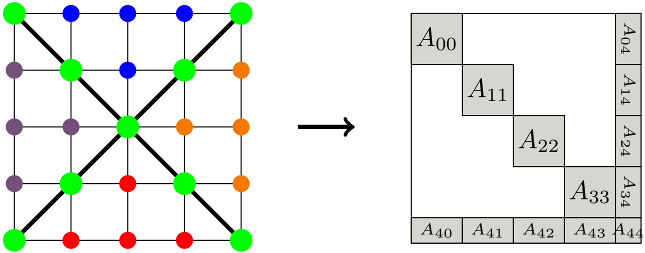
Parallelisation

Parallelisation | Direct Domain Decomposition

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.)

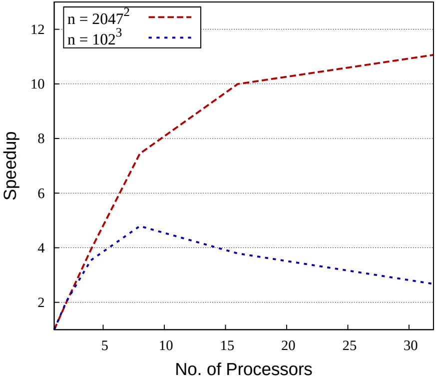
For the complexity of the parallel $\mathcal{H}$-LU factorisation in the model problem, we assume

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

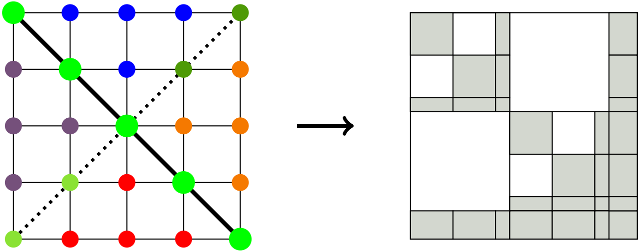
Then one obtains:

$$\mathcal{O}\Big(\frac{n \log^2 n}{p} \quad + \quad p^{1/d} n^{(d-1)/d} \log^2 n \log p \Big)$$

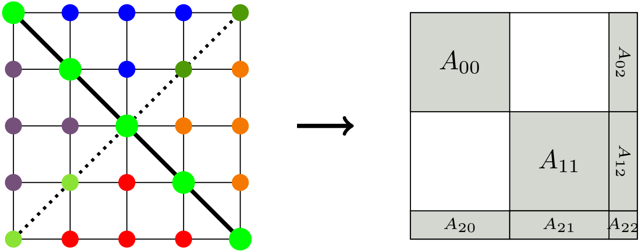
The speedup is limited by size of vertex separator, which increases with p.



Graph G is hierarchically partitioned with local vertex separators:



Graph G is hierarchically partitioned with local vertex separators:

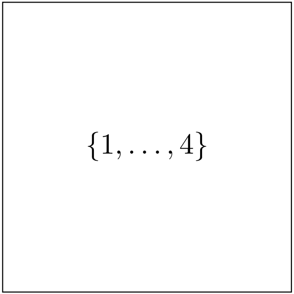


Parallel $\mathcal{H}$-LU factorisation is based on algorithm for direct domain decomposition with $p = 2$:

1. choose $i \in \{0, 1\}$ such that A_{ii} is on local processor;
2. factorise $A_{ii} = L_{ii}U_{ii}$, (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.)

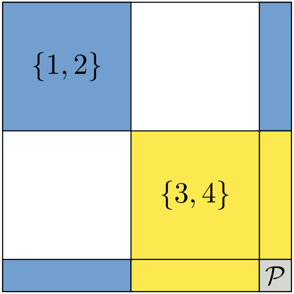
Parallelisation | Nested Dissection

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



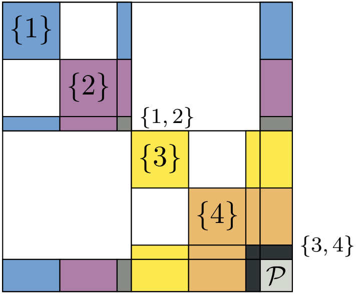
- on level 0, all processors handle the matrix,

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



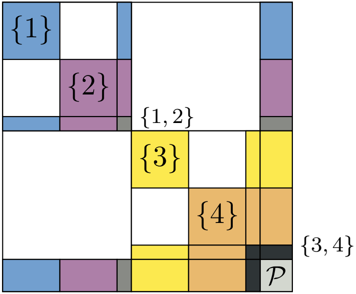
- on level 0, all processors handle the matrix,
- on level 1, $\mathcal{P}$ is split into two halves according to graph bisection,

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 $\mathcal{P}$.

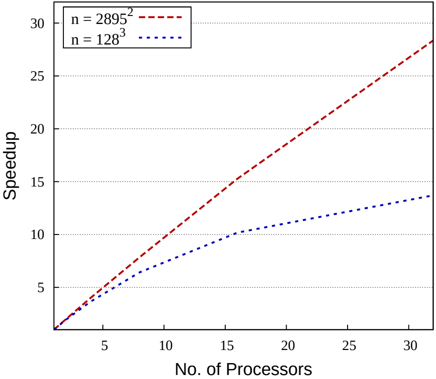
For the complexity of the parallel $\mathcal{H}$-LU factorisation in the model problem, we again assume

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

Then one obtains:

$$\mathcal{O}\Big(\frac{n \log^2 n}{p} \quad + \quad n^{(d-1)/d} \log^2 n \log p \Big)$$

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



Literature

- 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,
  "Computing and Visualization in Science", 11(4-6), pp. 273–291, 2008.
- G. Karypis and V. Kumar
  A fast and high quality multilevel scheme for partitioning irregular graphs,
  "SIAM Journal on Scientific Computing", 20(1), pp. 359–392, 1999.
- C.M. Fiduccia and R.M. Mattheyses,
  A linear-time heuristic for improving network partitions,
  In "Proceedings of the 19th Design Automation Conference", pp. 175–181, IEEE, 1982.