Parallel Black Box $\mathcal{H}$-LU Preconditioning

Ronald Kriemann
MPI MIS Leipzig

Workshop "Complex Systems"
METU

2009-05-14/15

Overview

1. $\mathcal{H}$-Matrices
2. Algebraic Clustering
3. Algebraic Admissibility
4. Nested Dissection
5. Numerical Experiments

$\mathcal{H}$-Matrices

Uniformly elliptic 2nd order PDE

$$\operatorname{div} \alpha(x) \, \nabla u(x) = f(x), \quad x \in \Omega$$

with Dirichlet/Neumann boundary conditions.

Galerkin discretisation

$$Ax = b, \qquad A_{ij} = \langle \nabla\varphi_i, \alpha\nabla\varphi_j \rangle_{L^2(\Omega)}$$

with basis functions

$$\varphi_i : \Omega \to \mathbb{R}, \qquad i \in I = \{1, \dots, N\}$$

Goal: Solve the system fast and robust using LU factorisation of A as preconditioner.

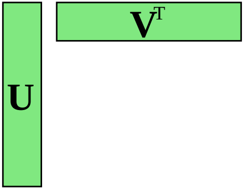
Problem: LU factors are usually prohibitively dense.

Solution: Compute approximate LU factorisation using $\mathcal{H}$-matrices with (almost) linear complexity.

A matrix $M \in \mathbb{R}^{n\times m}$ of rank $\leq k$ can be represented as

$$M = UV^T, \quad U \in \mathbb{R}^{n\times k}, V \in \mathbb{R}^{m\times k}$$

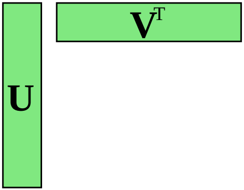
- R(k)-matrix format

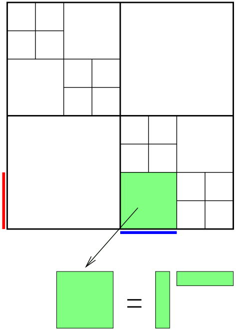


A matrix $M \in \mathbb{R}^{n \times m}$ of rank $\leq k$ can be represented as

$$M = UV^T, \quad U \in \mathbb{R}^{n \times k}, V \in \mathbb{R}^{m \times k}$$

► R(k)-matrix format





For a block-wise low-rank matrix $M \in \mathbb{R}^{n \times m}$

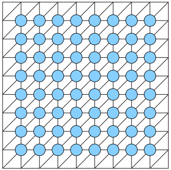
- each block is R(k)-matrix
- for small blocks: fullmatrix format

► $\mathcal{H}$-matrix format with hierarchically block organisation.

Needed: reordering (clustering) of index sets to allow low-rank representation.

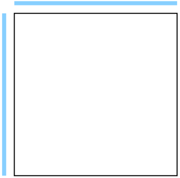
Domain

Construct cluster tree using geometrical data:



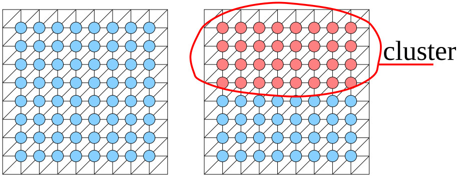
Matrix

Construct block cluster tree



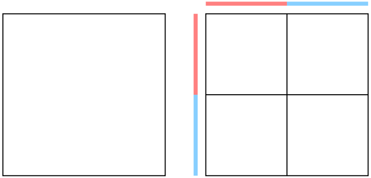
Domain

Construct cluster tree using geometrical data:



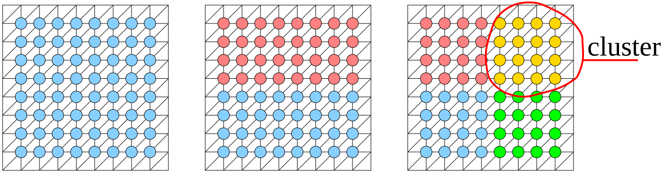
Matrix

Construct block cluster tree



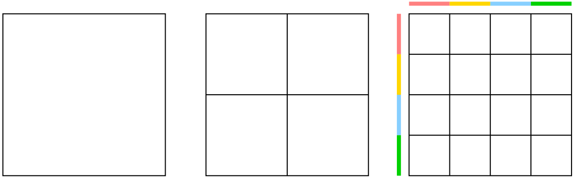
Domain

Construct cluster tree using geometrical data:



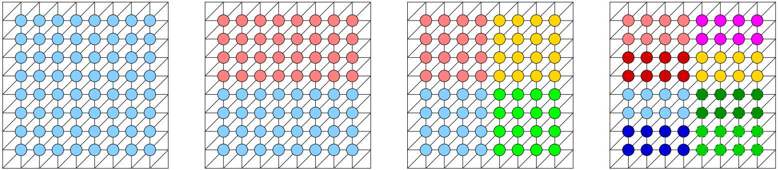
Matrix

Construct block cluster tree



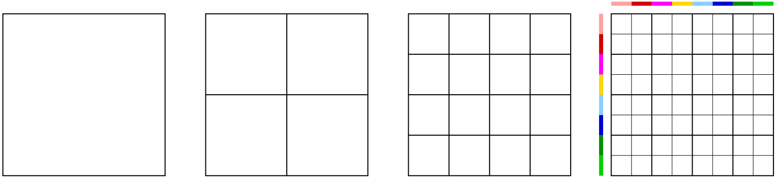
Domain

Construct cluster tree using geometrical data:



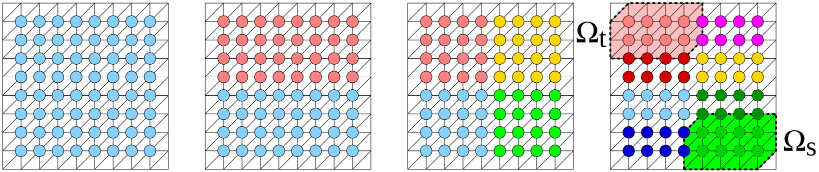
Matrix

Construct block cluster tree



Domain

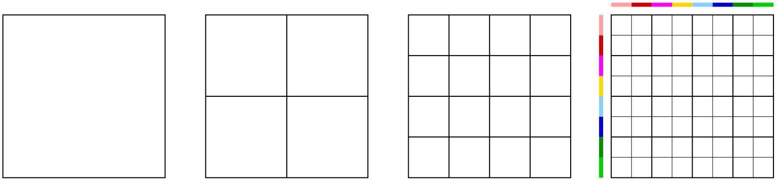
Construct cluster tree using geometrical data:



Matrix

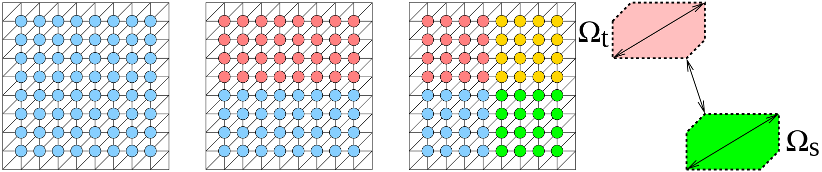
Construct block cluster tree with admissibility condition

$$\min(\mathrm{diam}(t), \mathrm{diam}(s)) \leq \eta \, \mathrm{dist}(t, s), \quad \eta > 0$$



Domain

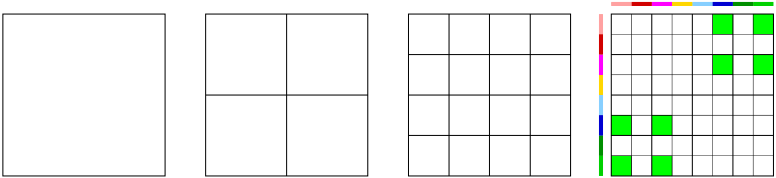
Construct cluster tree using geometrical data:



Matrix

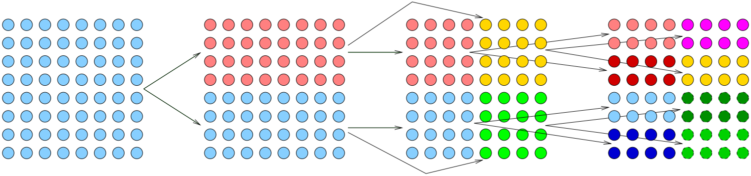
Construct block cluster tree with admissibility condition

$$\min(\mathrm{diam}(t), \mathrm{diam}(s)) \leq \eta \, \mathrm{dist}(t, s), \quad \eta > 0$$



Domain

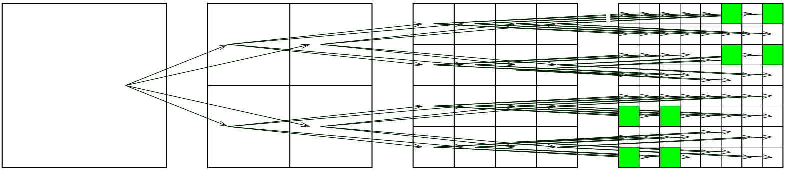
Construct cluster tree using geometrical data:

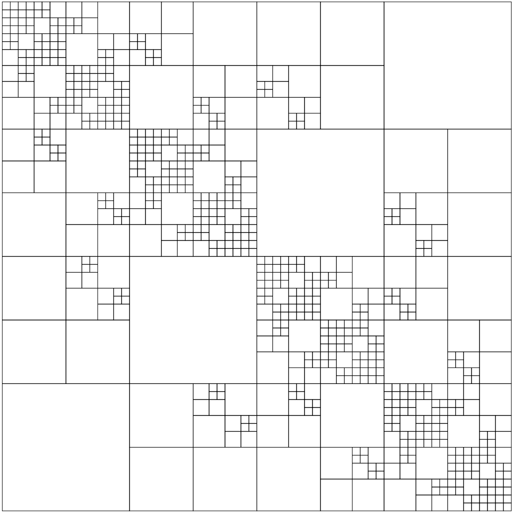


Matrix

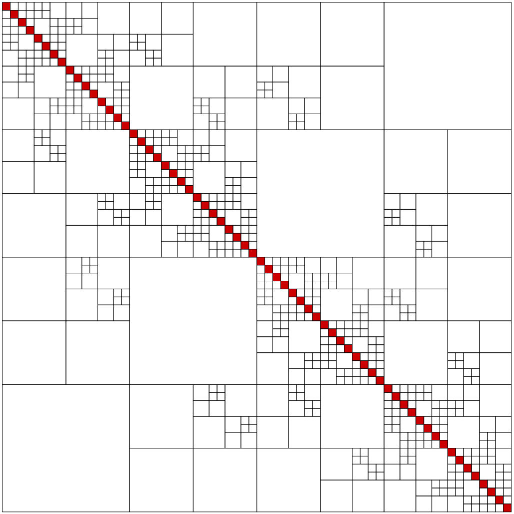
Construct block cluster tree with admissibility condition

$$\min(\operatorname{diam}(t), \operatorname{diam}(s)) \leq \eta \operatorname{dist}(t, s), \quad \eta > 0$$

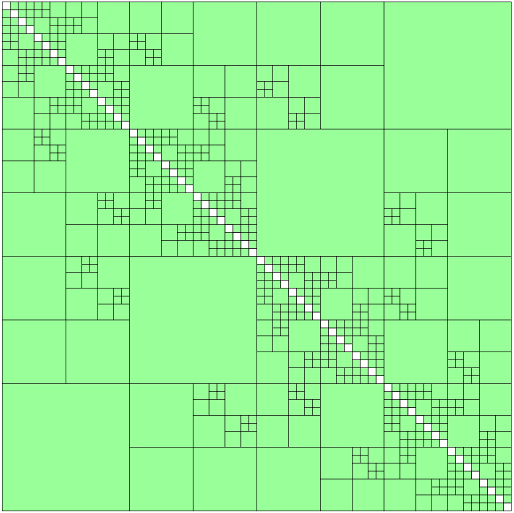




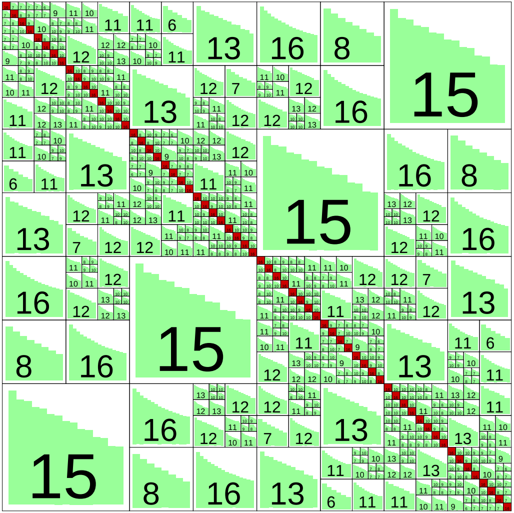
- $\mathcal{O}(n)$ blocks



- $\mathcal{O}(n)$ blocks
- Small red blocks: full matrices



- $\mathcal{O}(n)$ blocks
- Small red blocks: full matrices
- All other blocks: $R(k)$-matrices



- block-wise: exponential decay of singular values

Due to hierarchical block structure, standard recursive block algorithms can be used, e.g. for multiplication:

$$\begin{pmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{pmatrix} \cdot \begin{pmatrix} B_{11} & B_{12} \\ B_{21} & B_{22} \end{pmatrix} = \begin{pmatrix} A_{11} \cdot B_{11} + A_{12} \cdot B_{21} & A_{11} \cdot B_{12} + A_{12} \cdot B_{22} \\ A_{21} \cdot B_{11} + A_{22} \cdot B_{21} & A_{21} \cdot B_{12} + A_{22} \cdot B_{22} \end{pmatrix}$$

Due to hierarchical block structure, standard recursive block algorithms can be used, e.g. for multiplication:

$$\begin{pmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{pmatrix} \cdot \begin{pmatrix} B_{11} & B_{12} \\ B_{21} & B_{22} \end{pmatrix} = \begin{pmatrix} A_{11} \cdot B_{11} + A_{12} \cdot B_{21} & A_{11} \cdot B_{12} + A_{12} \cdot B_{22} \\ A_{21} \cdot B_{11} + A_{22} \cdot B_{21} & A_{21} \cdot B_{12} + A_{22} \cdot B_{22} \end{pmatrix}$$

But, addition of low-rank matrices increases the rank and finally produces full-rank matrices.

To limit complexity, a truncated addition is performed using SVD:

$$A_1 B_1^T + A_2 B_2^T =: CD^T \quad \rightarrow \quad USV^T \quad \rightarrow \quad C'D'^T$$

with (predefined) $\mathrm{rank}(C'D'^T) < \mathrm{rank}(CD^T)$.

Due to hierarchical block structure, standard recursive block algorithms can be used, e.g. for multiplication:

$$\begin{pmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{pmatrix} \cdot \begin{pmatrix} B_{11} & B_{12} \\ B_{21} & B_{22} \end{pmatrix} = \begin{pmatrix} A_{11} \cdot B_{11} + A_{12} \cdot B_{21} & A_{11} \cdot B_{12} + A_{12} \cdot B_{22} \\ A_{21} \cdot B_{11} + A_{22} \cdot B_{21} & A_{21} \cdot B_{12} + A_{22} \cdot B_{22} \end{pmatrix}$$

But, addition of low-rank matrices increases the rank and finally produces full-rank matrices.

To limit complexity, a truncated addition is performed using SVD:

$$A_1 B_1^T + A_2 B_2^T =: C D^T \quad \to \quad U S V^T \quad \to \quad C' D'^T$$

with (predefined) $\operatorname{rank}(C' D'^T) < \operatorname{rank}(C D^T)$.

Complexity

truncation	$\mathcal{O}(n)$	multiplication	$\mathcal{O}\left(n \log^2 n\right)$
storage	$\mathcal{O}(n \log n)$	inversion	$\mathcal{O}\left(n \log^2 n\right)$
matrix $\times$ vector	$\mathcal{O}(n \log n)$	triangular solve	$\mathcal{O}\left(n \log^2 n\right)$
addition	$\mathcal{O}(n \log n)$	LU decomposition	$\mathcal{O}\left(n \log^2 n\right)$

To solve $Ax = b$ using $\mathcal{H}$-LU factorisation:

1. construct cluster tree using geometrical data,
2. construct block cluster tree using admissibility condition (based on geometrical data),
3. build $\mathcal{H}$-matrix representation of A,
4. perform $\mathcal{H}$-LU factorisation (with approximation due to truncated addition),
5. solve $Ax = b$ preconditioned with $\mathcal{H}$-LU approximated A^{-1}.

To solve $Ax = b$ using $\mathcal{H}$-LU factorisation:

1. construct cluster tree using geometrical data,
2. construct block cluster tree using admissibility condition (based on geometrical data),
3. build $\mathcal{H}$-matrix representation of A,
4. perform $\mathcal{H}$-LU factorisation (with approximation due to truncated addition),
5. solve $Ax = b$ preconditioned with $\mathcal{H}$-LU approximated A^{-1}.

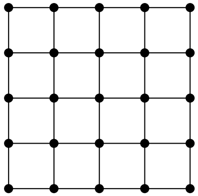
But what to do if no geometry information is available?

Algebraic Clustering

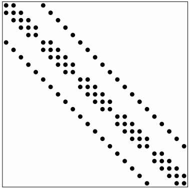
Consider

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

Using a uniform grid with step width h and standard piecewise linear finite elements with nodal points $x_i, i \in I$, one obtains the stiffness matrix $A \in \mathbb{R}^{I \times I}$ as



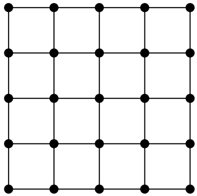
$\longrightarrow$



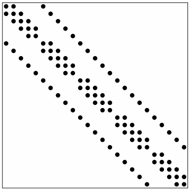
Consider

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

Using a uniform grid with step width h and standard piecewise linear finite elements with nodal points $x_i, i \in I$, one obtains the stiffness matrix $A \in \mathbb{R}^{I \times I}$ as



$\rightarrow$



Define the matrix graph $G(A) = (V_A, E_A)$ of A as

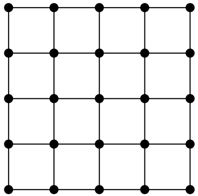
$$V_A := I, \quad E_A := \{(i,j) \ : \ i \neq j \wedge a_{ij} \neq 0\},$$

i.e. graph corresponds to sparsity pattern of stiffness matrix.

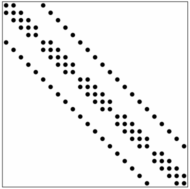
Consider

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

Using a uniform grid with step width h and standard piecewise linear finite elements with nodal points $x_i, i \in I$, one obtains the stiffness matrix $A \in \mathbb{R}^{I \times I}$ as



$\longleftarrow$



Define the matrix graph $G(A) = (V_A, E_A)$ of A as

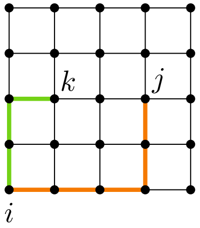
$$V_A := I, \quad E_A := \{(i,j) \;:\; i \neq j \wedge a_{ij} \neq 0\},$$

i.e. graph corresponds to sparsity pattern of stiffness matrix.

Define distance $\mathrm{dist}_G(i, j)$ between nodes $i, j \in I$ as length of shortest path in $G(A)$. Then, for $i, j \in I$ we have:

$$\|x_i - x_j\|_2 \leq \mathrm{dist}_G(i, j)h,$$

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

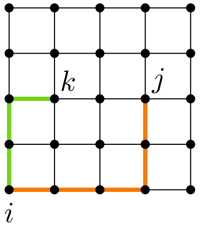


$$\|x_i - x_j\|_2 = \sqrt{13}h, \quad \mathrm{dist}_G(i, j) = 5$$
$$\|x_i - x_k\|_2 = \sqrt{5}h, \quad \mathrm{dist}_G(i, k) = 3$$

Define distance $\mathrm{dist}_G(i,j)$ between nodes $i,j \in I$ as length of shortest path in $G(A)$. Then, for $i,j \in I$ we have:

$$\|x_i - x_j\|_2 \leq \mathrm{dist}_G(i,j)h,$$

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

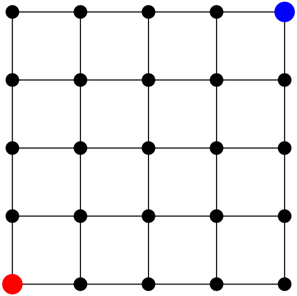


$$\|x_i - x_j\|_2 = \sqrt{13}h, \quad \mathrm{dist}_G(i,j) = 5$$
$$\|x_i - x_k\|_2 = \sqrt{5}h, \quad \mathrm{dist}_G(i,k) = 3$$

In model problem: since nodes in $G(A)$ with small distance are also geometrically neighboured, one can use graph distance to cluster indices.

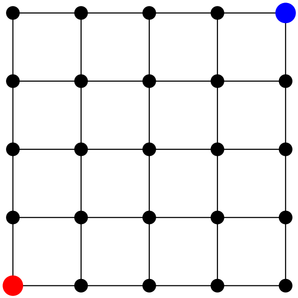
Algorithm:

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



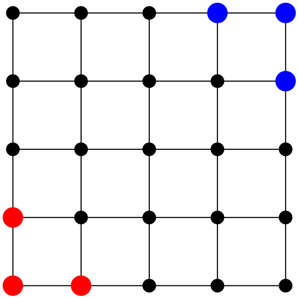
Algorithm:

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



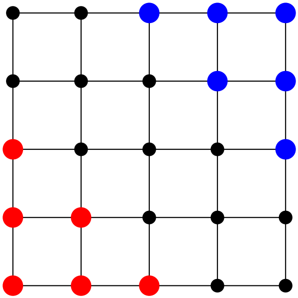
Algorithm:

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



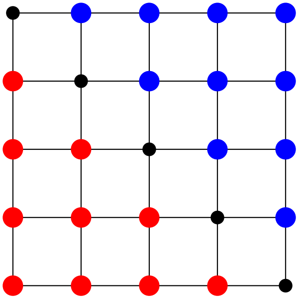
Algorithm:

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



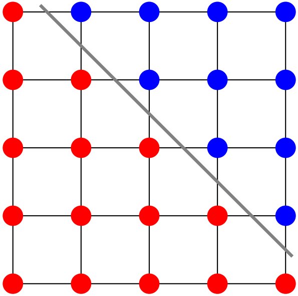
Algorithm:

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



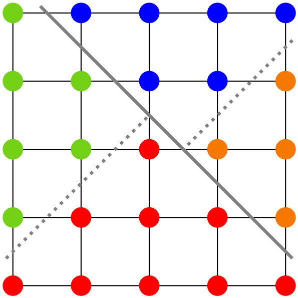
Algorithm:

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



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*

$$\#V_1 \sim \#V_2 \qquad \text{and}$$
$$\#\{(i, j) \in E \,:\, i \in V_1 \wedge j \in V_2\} = \min. \qquad \text{(edge-cut)}$$

A small edge-cut corresponds to a low-rank coupling of matrix blocks.

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*

$$\#V_1 \sim \#V_2 \qquad \text{and}$$
$$\#\{(i,j) \in E \ : \ i \in V_1 \wedge j \in V_2\} = \min . \quad \text{(edge-cut)}$$

A small edge-cut corresponds to a low-rank coupling of matrix blocks.

Although the graph partitioning problem is NP-hard good approximation algorithms exist, e.g. multilevel or spectral methods. Furthermore, they are available in open source packages, e.g. METIS, Chaco or Scotch.

Algebraic Admissibility

To apply the standard admissibility condition

$$\min(\mathrm{diam}(t), \mathrm{diam}(s)) \leq \eta \, \mathrm{dist}(t, s)$$

for a block cluster $(t, s) \in V \times V$, one needs to define distance and diameter of clusters in a graph.

To apply the standard admissibility condition

$$\min(\operatorname{diam}(t), \operatorname{diam}(s)) \leq \eta \operatorname{dist}(t, s)$$

for a block cluster $(t, s) \in V \times V$, one needs to define distance and diameter of clusters in a graph.

- For $V_1, V_2 \subset V$, the distance between V_1 and V_2 is defined as

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

- The diameter of a sub graph induced by $V' \subseteq V$ is defined as

$$\operatorname{diam}_G(V') := \max_{i,j \in V'} \operatorname{dist}_G(i, j).$$

To apply the standard admissibility condition

$$\min(\operatorname{diam}(t), \operatorname{diam}(s)) \leq \eta \operatorname{dist}(t, s)$$

for a block cluster $(t, s) \in V \times V$, one needs to define distance and diameter of clusters in a graph.

- For $V_1, V_2 \subset V$, the distance between V_1 and V_2 is defined as

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

- The diameter of a sub graph induced by $V' \subseteq V$ is defined as

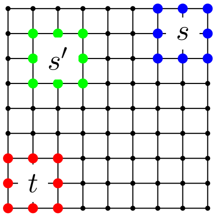
$$\operatorname{diam}_G(V') := \max_{i,j \in V'} \operatorname{dist}_G(i, j).$$

Problem: diameter and distance in G costs $\mathcal{O}\left(n^2\right)$.

Solution: approximate cluster diameter and construct cluster surrounding ensuring admissibility.

For testing admissibility of block cluster $(t, s) \in V \times V$

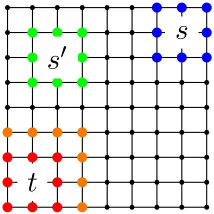
- choose $i \in t$ and compute $j \in t$ with $\mathrm{dist}_G(i, j) = \max$,
- $\mathrm{diam}_G(t) \leq 2\,\mathrm{dist}_G(i, j) =: \widetilde{\mathrm{diam}}$,
- build surrounding $\tilde{t}$ around t with $\frac{1}{\eta}\widetilde{\mathrm{diam}}$ layers,
- if $\tilde{t} \cap s = \emptyset$ then (t, s) is admissible.



Solution: approximate cluster diameter and construct cluster surrounding ensuring admissibility.

For testing admissibility of block cluster $(t, s) \in V \times V$

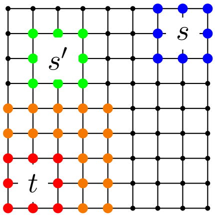
- choose $i \in t$ and compute $j \in t$ with $\mathrm{dist}_G(i, j) = \max$,
- $\mathrm{diam}_G(t) \leq 2\,\mathrm{dist}_G(i, j) =: \widetilde{\mathrm{diam}}$,
- build surrounding $\tilde{t}$ around t with $\frac{1}{\eta}\widetilde{\mathrm{diam}}$ layers,
- if $\tilde{t} \cap s = \emptyset$ then (t, s) is admissible.



Solution: approximate cluster diameter and construct cluster surrounding ensuring admissibility.

For testing admissibility of block cluster $(t, s) \in V \times V$

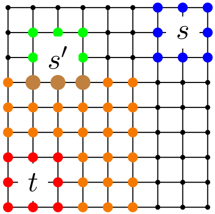
- choose $i \in t$ and compute $j \in t$ with $\mathrm{dist}_G(i, j) = \max$,
- $\mathrm{diam}_G(t) \leq 2\,\mathrm{dist}_G(i, j) =: \widetilde{\mathrm{diam}}$,
- build surrounding $\tilde{t}$ around t with $\frac{1}{\eta}\widetilde{\mathrm{diam}}$ layers,
- if $\tilde{t} \cap s = \emptyset$ then (t, s) is admissible.



Solution: approximate cluster diameter and construct cluster surrounding ensuring admissibility.

For testing admissibility of block cluster $(t, s) \in V \times V$

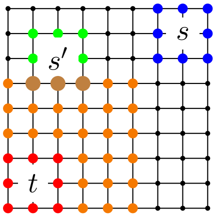
- choose $i \in t$ and compute $j \in t$ with $\mathrm{dist}_G(i, j) = \max$,
- $\mathrm{diam}_G(t) \leq 2\,\mathrm{dist}_G(i, j) =: \widetilde{\mathrm{diam}}$,
- build surrounding $\tilde{t}$ around t with $\frac{1}{\eta}\widetilde{\mathrm{diam}}$ layers,
- if $\tilde{t} \cap s = \emptyset$ then (t, s) is admissible.



Solution: approximate cluster diameter and construct cluster surrounding ensuring admissibility.

For testing admissibility of block cluster $(t, s) \in V \times V$

- choose $i \in t$ and compute $j \in t$ with $\mathrm{dist}_G(i, j) = \max$,
- $\mathrm{diam}_G(t) \le 2\,\mathrm{dist}_G(i, j) =: \widetilde{\mathrm{diam}}$,
- build surrounding $\tilde{t}$ around t with $\frac{1}{\eta}\widetilde{\mathrm{diam}}$ layers,
- if $\tilde{t} \cap s = \emptyset$ then (t, s) is admissible.



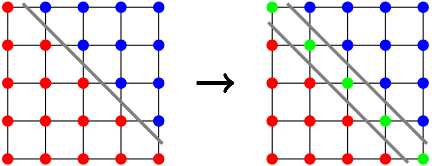
With usual FEM sparsity patterns, this procedure has complexity

$$\mathcal{O}\left(\#t\right).$$

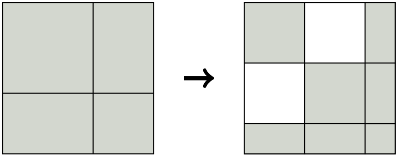
Nested Dissection

In nested dissection the two constructed sub graphs of a partition have to be separated by a (minimal) vertex separator.

Matrix graph:

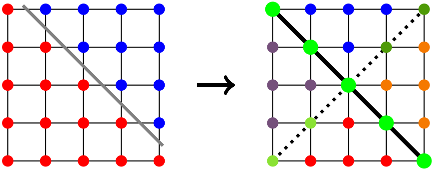


Matrix:

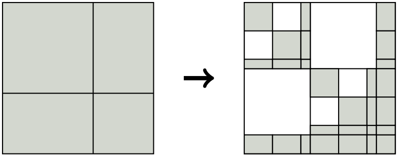


In nested dissection the two constructed sub graphs of a partition have to be separated by a (minimal) vertex separator.

Matrix graph:

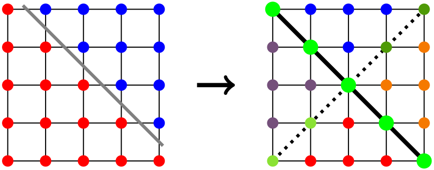


Matrix:

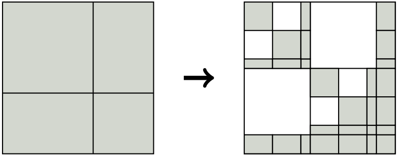


In nested dissection the two constructed sub graphs of a partition have to be separated by a (minimal) vertex separator.

Matrix graph:



Matrix:



Advantages of Nested Dissection

- zero blocks do not fill up during $\mathcal{H}$-LU factorisation,
- blocks can be computed in parallel.

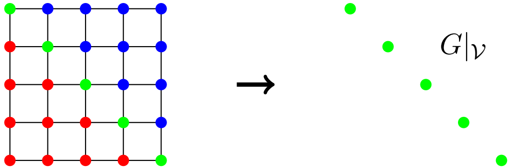
A vertex separator can be obtained by computing a vertex cover of the edge-cut between both node sets in a partition.

But for $\mathcal{H}$-matrices the vertex separator has to be further partitioned to form a cluster tree.

A vertex separator can be obtained by computing a vertex cover of the edge-cut between both node sets in a partition.

But for $\mathcal{H}$-matrices the vertex separator has to be further partitioned to form a cluster tree.

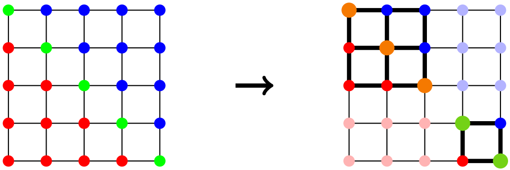
Problem: restricting G to nodes in vertex separator $\mathcal{V}$ might remove important edges, e.g.



A vertex separator can be obtained by computing a vertex cover of the edge-cut between both node sets in a partition.

But for $\mathcal{H}$-matrices the vertex separator has to be further partitioned to form a cluster tree.

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



Solution: modify previous BFS based algorithm to perform partitioning in a surrounding of the vertex separator.

Numerical Experiments

Solving model problem:

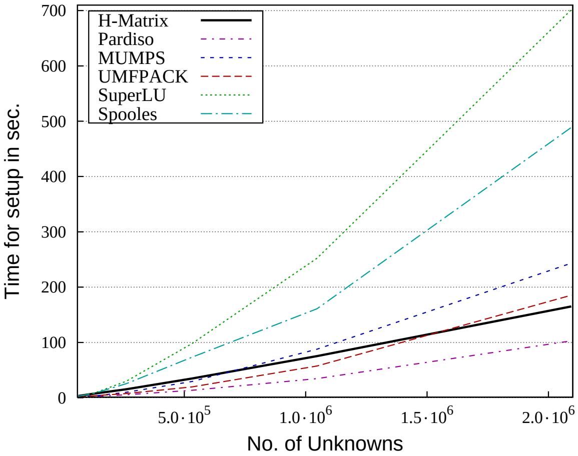
N	Geometric		Algebraic	
	Time (s)	Mem (MB)	Time (s)	Mem (MB)
253^2	0.9	51	1.3	47
358^2	1.9	86	2.9	94
511^2	4.5	212	6.5	198
729^2	9.6	371	15.0	402
1023^2	20.2	878	31.6	819
40^3	12.6	99	32.7	135
51^3	46.9	300	97.6	323
64^3	117.4	592	289.1	719
81^3	269.8	1410	804.3	1570
102^3	752.3	3020	1907.3	3370

Accuracy of $\mathcal{H}$-arithmetic chosen such that

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

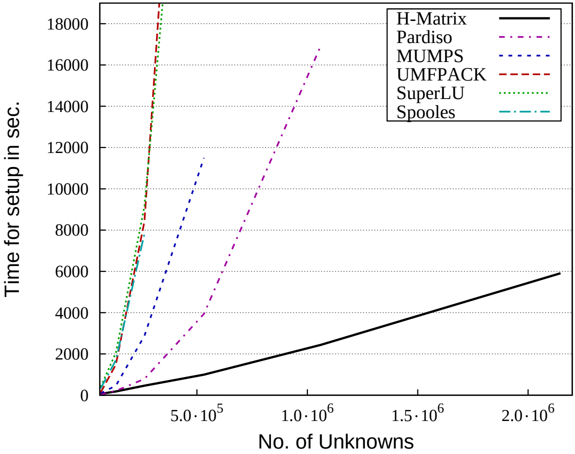
Solving

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

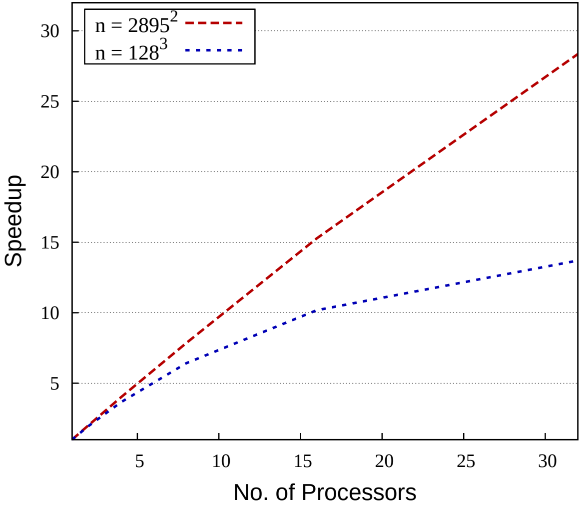


Solving

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



Parallel speedup for algebraic $\mathcal{H}$-LU factorisation in $\mathbb{R}^2$ and $\mathbb{R}^3$.



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.
- L. Grasedyck, W. Hackbusch and R. Kriemann,
  Performance of $\mathcal{H}$-LU Preconditioning for Sparse Matrices,
  to appear in "Computational Methods in Applied Mathematics".
- $\mathcal{H}$-Lib$^{\text{pro}}$
  http://www.hlibpro.org