High Quality Hypergraph Partitioning

German-Israeli Winter School on Algorithms for Big Data · November 15, 2017
Robin Andre, Yaroslav Akhremtsev, Vitali Henne, Tobias Heuer, Henning Meyerhenke, Peter Sanders, Sebastian Schlag, Christian Schulz
Graphs and Hypergraphs

**Graph** \( G = (V, E) \)

- Models **relationships** between **objects**
- Dyadic (2-ary) relationships

**Hypergraph** \( H = (V, E) \)

- Generalization of a graph
  \( \Rightarrow \) hyperedges connect \( \geq 2 \) nodes
- Arbitrary (d-ary) relationships
- Edge set \( E \subseteq \mathcal{P}(V) \setminus \emptyset \)
ε-Balanced Hypergraph Partitioning

**Partition** hypergraph \( H = (V, E, c : V \rightarrow \mathbb{R}_{>0}, \omega : E \rightarrow \mathbb{R}_{>0}) \) into \( k \) disjoint blocks \( \Pi = \{V_1, \ldots, V_k\} \) such that

- Blocks \( V_i \) are **roughly equal-sized**: 
  \[
  c(V_i) \leq (1 + \varepsilon) \left\lceil \frac{c(V)}{k} \right\rceil
  \]

- **Objective** function on hyperedges is **minimized**
\(\varepsilon\)-Balanced Hypergraph Partitioning

Partition hypergraph \(H = (V, E, c : V \rightarrow R_{>0}, \omega : E \rightarrow R_{>0})\) into \(k\) disjoint blocks \(\Pi = \{V_1, \ldots, V_k\}\) such that

- Blocks \(V_i\) are **roughly equal-sized**:
  \[
  c(V_i) \leq (1 + \varepsilon) \left\lceil \frac{c(V)}{k} \right\rceil
  \]

- **Objective** function on hyperedges is **minimized**
ε-Balanced Hypergraph Partitioning

Partition hypergraph $H = (V, E, c : V \rightarrow \mathbb{R}_{>0}, \omega : E \rightarrow \mathbb{R}_{>0})$ into $k$ disjoint blocks $\Pi = \{V_1, \ldots, V_k\}$ such that

- Blocks $V_i$ are roughly equal-sized:
  $$c(V_i) \leq (1 + \varepsilon) \left\lceil \frac{c(V)}{k} \right\rceil$$

- Objective function on hyperedges is minimized

Common Objectives:

- cut: $\sum_{e \in \text{Cut}} \omega(e)$
\( \varepsilon \)-Balanced Hypergraph Partitioning

Partition hypergraph \( H = (V, E, c : V \rightarrow R_{>0}, \omega : E \rightarrow R_{>0}) \) into \( k \) disjoint blocks \( \Pi = \{V_1, \ldots, V_k\} \) such that

- Blocks \( V_i \) are roughly equal-sized:
  \[ c(V_i) \leq (1 + \varepsilon) \left\lceil \frac{c(V)}{k} \right\rceil \]

- Objective function on hyperedges is minimized

Common Objectives:
- cut: \( \sum_{e \in \text{Cut}} \omega(e) \)
- Connectivity: \( \sum_{e \in \text{cut}} (\lambda - 1) \omega(e) \)
\( \varepsilon \)-Balanced Hypergraph Partitioning

Partition hypergraph \( H = (V, E, c : V \rightarrow R_{>0}, \omega : E \rightarrow R_{>0}) \) into \( k \) disjoint blocks \( \Pi = \{ V_1, \ldots, V_k \} \) such that

- Blocks \( V_i \) are roughly equal-sized:
  \[
  c(V_i) \leq (1 + \varepsilon) \left\lceil \frac{c(V)}{k} \right\rceil
  \]

- Objective function on hyperedges is minimized

Common Objectives:

- \textbf{cut}: \( \sum_{e \in \text{Cut}} \omega(e) \)
- \textbf{Connectivity}: \( \sum_{e \in \text{cut}} (\lambda - 1) \omega(e) \)

\# blocks connected by \( e \)
Applications

VLSI Design

Warehouse Optimization

Complex Networks

Route Planning

Simulation

\[ \mathbb{R}^{n \times n} \ni Ax = b \in \mathbb{R}^n \]

Scientific Computing
Applications

- VLSI Design
- Warehouse Optimization
- Complex Networks
- Route Planning
- Simulation

\[ \mathbb{R}^{n \times n} \ni Ax = b \in \mathbb{R}^n \]
Parallel Sparse-Matrix Vector Product (SpM $\times$ V)

\[ y = A b \]

Setting:
- Repeated SpM $\times$ V on supercomputer
- $A$ is large $\Rightarrow$ distribute on multiple nodes
- Symmetric partitioning $\Rightarrow$ $y$ & $b$ divided conformally with $A$

[Catalyurek, Aykanat]
Parallel Sparse-Matrix Vector Product (SpM×V)

\[ y = A \cdot b \]

[Catalyürk, Aykanat]

**Task:** distribute \( A \) to nodes of supercomputer such that
- work is distributed **evenly**
- communication overhead is **minimized**

**Setting:**
- Repeated SpM×V on supercomputer
- \( A \) is large \( \Rightarrow \) distribute on multiple nodes
- Symmetric partitioning \( \Rightarrow \) \( y \) & \( b \) divided conformally with \( A \)
Naive Approach: Rowwise Decomposition

\[ A \in \mathbb{R}^{16 \times 16} \]
## Naive Approach: Rowwise Decomposition

Let \( A \in \mathbb{R}^{16 \times 16} \) be a matrix and partition it into four parts:

\[
\begin{array}{cccccccccccccccc}
1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 & 10 & 11 & 12 & 13 & 14 & 15 & 16 \\
\end{array}
\]

- \( P_1 \) contains rows 1 to 4
- \( P_2 \) contains rows 5 to 8
- \( P_3 \) contains rows 9 to 12
- \( P_4 \) contains rows 13 to 16

Each block is colored to illustrate the partitioning:

- Green for \( P_1 \)
- Orange for \( P_2 \)
- Blue for \( P_3 \)
- Red for \( P_4 \)

The matrix is partitioned rowwise, with each row assigned to one of the four partitions.
Naive Approach: Rowwise Decomposition

$A \in \mathbb{R}^{16 \times 16}$

Load Balancing?

⇒ 9

⇒ 12

⇒ 14

⇒ 12
Naive Approach: Rowwise Decomposition

$A \in \mathbb{R}^{16 \times 16}$

Load Balancing?

$\Rightarrow 9$

$\Rightarrow 12$

$\Rightarrow 14$

$\Rightarrow 12$

Communication Volume?

Commuication Volume?
Naive Approach: Rowwise Decomposition

\[ A \in \mathbb{R}^{16 \times 16} \]

Load Balancing?

\[ \Rightarrow 9 \]
\[ \Rightarrow 12 \]
\[ \Rightarrow 14 \]
\[ \Rightarrow 12 \]

Communication Volume?
Naive Approach: Rowwise Decomposition

\[ A \in \mathbb{R}^{16 \times 16} \]

\[
\begin{array}{cccccccccccccccc}
1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 & 10 & 11 & 12 & 13 & 14 & 15 & 16 \\
\hline
b & b & b & b & b & b & b & b & b & b & b & b & b & b & b & b \\
\end{array}
\]

Load Balancing?

\[ \Rightarrow 9 \]

\[ \Rightarrow 12 \]

\[ \Rightarrow 14 \]

\[ \Rightarrow 12 \]

Communication Volume?

\[ \Rightarrow 24 \text{ entries!} \]
Naive Approach: Rowwise Decomposition

\[ A \in \mathbb{R}^{16 \times 16} \]

Load Balancing?

\[ \Rightarrow 9 \]

\[ \Rightarrow 12 \]

\[ \Rightarrow 14 \]

\[ \Rightarrow 12 \]

Can we do better?

Communication Volume? \[ \Rightarrow 24 \text{ entries!} \]
From SpM $\times V$ to Hypergraph Partitioning

$A \in \mathbb{R}^{16 \times 16} \Rightarrow H = (V_R, E_C)$

- One vertex per row:
  $\Rightarrow V_R = \{v_1, v_2, \ldots, v_{16}\}$

- One hyperedge per column:
  $\Rightarrow E_C = \{e_1, e_2, \ldots, e_{16}\}$
From SpM $\times V$ to Hypergraph Partitioning

$$A \in \mathbb{R}^{16 \times 16} \Rightarrow H = (V_R, E_C)$$

- One vertex per row:
  $$\Rightarrow V_R = \{v_1, v_2, \ldots, v_{16}\}$$

- One hyperedge per column:
  $$\Rightarrow E_C = \{e_1, e_2, \ldots, e_{16}\}$$

$v_i \in V_R$:

- Inner product of row $i$ with $b$
  $$\Rightarrow c(v_i) := \# \text{ nonzeros}$$
From SpM $\times V$ to Hypergraph Partitioning

$A \in \mathbb{R}^{16 \times 16} \Rightarrow H = (V_R, E_C)$

- One vertex per row:
  $\Rightarrow V_R = \{v_1, v_2, \ldots, v_{16}\}$
- One hyperedge per column:
  $\Rightarrow E_C = \{e_1, e_2, \ldots, e_{16}\}$

$v_i \in V_R$:
- Inner product of row $i$ with $b$
  $\Rightarrow c(v_i) := \# \text{ nonzeros}$

$e_j \in E_C$:
- Set of vertices that need $b_j$
From SpM $\times$ V to Hypergraph Partitioning

\[ A \in \mathbb{R}^{16 \times 16} \Rightarrow H = (V_R, E_C) \]

- One vertex per row:
  \[ \Rightarrow V_R = \{v_1, v_2, \ldots, v_{16}\} \]
- One hyperedge per column:
  \[ \Rightarrow E_C = \{e_1, e_2, \ldots, e_{16}\} \]

\[ \begin{array}{ccccccccccc}
1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 & 10 & 11 & 12 & 13 & 14 & 15 & 16 \\
\end{array} \]

\[ \begin{array}{cccccccccccccccc}
\begin{pmatrix}
b & b & b & b & b & b & b & b & b & b & b & b & b & b & b & b \\
\end{pmatrix} \\
\end{array} \]

\[ \begin{array}{cccccccccccccccc}
1 & x & x & x & x & x & x & x & x & x & x & x & x & x & x & x \\
2 & x & x & x & x & x & x & x & x & x & x & x & x & x & x & x \\
3 & x & x & x & x & x & x & x & x & x & x & x & x & x & x & x \\
4 & x & x & x & x & x & x & x & x & x & x & x & x & x & x & x \\
5 & x & x & x & x & x & x & x & x & x & x & x & x & x & x & x \\
6 & x & x & x & x & x & x & x & x & x & x & x & x & x & x & x \\
7 & x & x & x & x & x & x & x & x & x & x & x & x & x & x & x \\
8 & x & x & x & x & x & x & x & x & x & x & x & x & x & x & x \\
9 & x & x & x & x & x & x & x & x & x & x & x & x & x & x & x \\
10 & x & x & x & x & x & x & x & x & x & x & x & x & x & x & x \\
11 & x & x & x & x & x & x & x & x & x & x & x & x & x & x & x \\
12 & x & x & x & x & x & x & x & x & x & x & x & x & x & x & x \\
13 & x & x & x & x & x & x & x & x & x & x & x & x & x & x & x \\
14 & x & x & x & x & x & x & x & x & x & x & x & x & x & x & x \\
15 & x & x & x & x & x & x & x & x & x & x & x & x & x & x & x \\
16 & x & x & x & x & x & x & x & x & x & x & x & x & x & x & x \\
\end{array} \]

**Solution:** $\epsilon$-balanced partition of $H$

- Balanced partition $\mapsto$ computational load balance
- Small $(\lambda - 1)$-cutsize $\mapsto$ minimizing communication volume
From SpM $\times$ V to Hypergraph Partitioning
From SpM × V to Hypergraph Partitioning
From $\text{SpM} \times \mathbf{V}$ to Hypergraph Partitioning
From SpM $\times$ V to Hypergraph Partitioning
From SpM $\times$ V to Hypergraph Partitioning
From SpM × V to Hypergraph Partitioning
From Hypergraph Partitioning to SpM $\times V$

<table>
<thead>
<tr>
<th></th>
<th>P₁</th>
<th>P₂</th>
<th>P₃</th>
<th>P₄</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
</tr>
<tr>
<td>2</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
</tr>
<tr>
<td>3</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
</tr>
<tr>
<td>4</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
</tr>
<tr>
<td>5</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
</tr>
<tr>
<td>6</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
</tr>
<tr>
<td>7</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
</tr>
<tr>
<td>8</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
</tr>
<tr>
<td>9</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
</tr>
<tr>
<td>10</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
</tr>
<tr>
<td>11</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
</tr>
<tr>
<td>12</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
</tr>
<tr>
<td>13</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
</tr>
<tr>
<td>14</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
</tr>
<tr>
<td>15</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
</tr>
<tr>
<td>16</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
</tr>
</tbody>
</table>
From Hypergraph Partitioning to SpM × V

Load Balancing?
From Hypergraph Partitioning to $\text{SpM} \times V$

Load Balancing?

$\Rightarrow 12$
$\Rightarrow 12$
$\Rightarrow 12$
$\Rightarrow 12$
From Hypergraph Partitioning to $\text{SpM} \times V$

Where are the cut-hyperedges?

Load Balancing?

$\Rightarrow 12$

$\Rightarrow 12$

$\Rightarrow 12$

$\Rightarrow 12$

Communication Volume?
From Hypergraph Partitioning to $\text{SpM} \times V$

Where are the cut-hyperedges?

P1

P2

P3

P4

Load Balancing?

⇒ 12

⇒ 12

⇒ 12

⇒ 12

Communication Volume? ⇒ 6 entries!
How does Hypergraph Partitioning work?
How does Hypergraph Partitioning work?

**Bad News:**
- Hypergraph Partitioning is $\text{NP}$-hard
- Even finding *good approximate* solutions for graphs is $\text{NP}$-hard
Successful Heuristic: Multilevel Paradigm

Input Hypergraph

Coarsening

match /

cluster

contract

···
Successful Heuristic: Multilevel Paradigm

Input Hypergraph

Coarsening

match / cluster

contract

Initial Partitioning
Successful Heuristic: Multilevel Paradigm

Input Hypergraph

Coarsening

match / cluster

contract

Output Partition

Uncoarsening

local search

uncontract

Initial Partitioning

···
Taxonomy of Hypergraph Partitioning Tools

Recursive Bisection
- MLPart
- PaToH
- Sparse Matrices
- Mondriaan
- Zoltan

Direct k-way
- VLSI
- hMetis-R
- hMetis-K
- Parkway
- Parallel
- UMPa
- Multi-objective

Years:
- 1998
- 1999
- 2005
- 2006
- 2008
- 2013
Taxonomy of Hypergraph Partitioning Tools

**Recursive Bisection**
- MLPart
- PaToH
- Sparse Matrices
- Mondriaan

**Direct k-way**
- VLSI
- hMetis-R
- hMetis-K
- Parkway
- UMPa multi-objective

**n-Level**
- KaHyPar-R

**Parallel**
- Zoltan

**Time Periods**
- 1998
- 1999
- 2005
- 2006
- 2008
- 2013
- 2016
- 2017
Why Yet Another Multilevel Algorithm?

Input Hypergraph

Coarsening

match / cluster

contract

Uncoarsening

Output Partition

local search

uncontract

Initial Partitioning
Why Yet Another Multilevel Algorithm?

**Tradeoff:**

# levels ↑:
- + Quality
- – Running time

---

Input Hypergraph → Output Partition

Coarsening
- match /
- cluster
- contract

Uncoarsening
- local search
- uncontract
Why Yet Another Multilevel Algorithm?

Tradeoff:
# levels $\uparrow$:
- + Quality
- – Running time

Karlsruhe Hypergraph Partitioning
⇒ Evade tradeoff $\rightsquigarrow$ n levels [ALENEX’16]
⇒ Combine high quality with good performance
KaHyPar: Novel Algorithmic Ingredients

Coarsening

Input Hypergraph

match /

contract

cluster

local search

uncontract

Output Partition

Initial Partitioning
KaHyPar: Novel Algorithmic Ingredients

Min-Hash Based Sparsification

[ALENEX'17]
KaHyPar: Novel Algorithmic Ingredients

Min-Hash Based Sparsification
[ALENEX’17]

Community-Aware Coarsening
[SEA’17]
KaHyPar: Novel Algorithmic Ingredients

Min-Hash Based Sparsification

Community-Aware Coarsening

Fast $n$-Level Coarsening

Output Partition

local search

uncontract

Initial Partitioning
KaHyPar: Novel Algorithmic Ingredients

- Min-Hash Based Sparsification [ALENEX’17]
- Community-Aware Coarsening [SEA’17]
- Fast n-Level Coarsening [ALENEX’17]
- Engineered k-way FM [ALENEX’17]

Gain-Cache of •:
KaHyPar: Novel Algorithmic Ingredients

- Min-Hash Based Sparsification [ALENEX’17]
- Community-Aware Coarsening [SEA’17]
- Fast $n$-Level Coarsening [ALENEX’17]
- Engineered $k$-way FM [ALENEX’17]

Input Hypergraph

Initial Partitioning

Fast $n$-Level Coarsening

Coarsening

Min-Hash Based Sparsification

Max-Flow Min-Cut Refinement

[Heuer, Master’s Thesis (upcoming)]

Output Partition

Gain-Cache of $\omega$:

$\omega(e_1)$

$\omega(e_2)$

$\omega(e_3)$

$e'$

$e''$

Local search

Community-Aware Coarsening

$V_1$

$V_2$

$V_3$

$V_4$
KaHyPar: Novel Algorithmic Ingredients

Min-Hash Based Sparsification [ALENEX'17]

Community-Aware Coarsening [SEA'17]

Fast n-Level Coarsening [ALENEX'17]

Algorithm Configuration

Algorithm $A \leftarrow \{\text{Config } C_1, \text{Config } C_2\}$

Max-Flow Min-Cut Refinement [Heuer, Master's Thesis (upcoming)]

Engineered $k$-way FM [ALENEX'17]

Gain-Cache of $\bullet$:

1 2 3 4

2 3 4 1

Algorithm $A \leftarrow \{\text{Config } C_1, \text{Config } C_2\}$

Min-Hash Based Sparsification [ALENEX'17]

Community-Aware Coarsening [SEA'17]

Fast n-Level Coarsening [ALENEX'17]
KaHyPar: Novel Algorithmic Ingredients

Min-Hash Based Sparsification [ALENEX’17]

Community-Aware Coarsening [SEA’17]

Fast \(n\)-Level Coarsening [ALENEX’17]

Algorithm A \(\leftarrow\) \{Config \(C_1\), Config \(C_2\)

Algorithm Configuration
[Öhl, Bachelor’s Thesis (upcoming)]

Max-Flow Min-Cut Refinement [Heuer, Master’s Thesis (upcoming)]

Engineered \(k\)-way FM [ALENEX’17]

Gain-Cache of \(\bullet\):

initial partitioning
Latest Experimental Results

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>$t[s]$</th>
</tr>
</thead>
<tbody>
<tr>
<td>KaHyPar-CA</td>
<td>21.0</td>
</tr>
<tr>
<td>KaHyPar-F</td>
<td>46.8</td>
</tr>
<tr>
<td>hMetis-R</td>
<td>63.1</td>
</tr>
</tbody>
</table>

Infeasible solutions:

- 0.00
- 0.01
- 0.05
- 0.10
- 0.20
- 0.40
- 0.60
- 0.80
- 1.00
KaHyPar

- **n-Level** Partitioning Framework
- **Objectives:**
  - Cut
  - Connectivity ($\lambda - 1$)
- **Partitioning Modes:**
  - Recursive bisection
  - Direct $k$-way
- **Upcoming Features:**
  - Evolutionary algorithm
  - Flow-based refinement
  - Advanced local search algorithms
- [http://www.kahypar.org](http://www.kahypar.org)
References


