High-Quality (Hyper)Graph Partitioning

ICIAM · Combinatorial Scientific Computing · July 15th, 2019

Y. Akhremtsev, T. Heuer, P. Sanders, S. Schlag, C. Schulz, D. Seemaier, D. Strash
Research Areas in Peter Sanders’ Group

- Route Planning
- Text Indexing
- SAT Solving
- Parallel Sorting
- Communication-Efficient Algorithms
- Shared-Memory Data Structures
- Hypergraph Partitioning
- Graph Generators
This Talk: Hypergraph & Graph Partitioning

Graph Partitioning

Route Planning

Text Indexing

SAT Solving

Parallel Sorting

Communication-Efficient Algorithms

Graph Generators

Shared-Memory Data Structures

Hypergraph Partitioning
Research Methodology

Algorithm Engineering

- Application
- Practice
- Theory
- Model
- Design
- Implementation
- Analysis
- Experiment
Graphs and Hypergraphs

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

- **vertices**
- **edges**
- 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 \( E \subseteq \mathcal{P}(V) \setminus \emptyset \)
\( \varepsilon \)-Balanced Hypergraph Partitioning (HGP)

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
ε-Balanced Hypergraph Partitioning (HGP)

Partition hypergraph $H = (V, E, c : V \to \mathbb{R}_{>0}, \omega : E \to \mathbb{R}_{>0})$ 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 (HGP)

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 HGP Objectives:
- Cut-Net: \( \sum_{e \in \text{Cut}} \omega(e) \)
ε-Balanced Hypergraph Partitioning (HGP)

Partition hypergraph \( H = (V, E, c : V \rightarrow \mathbb{R}^+, \omega : E \rightarrow \mathbb{R}^+) \) 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 HGP Objectives:
- Cut-Net: \( \sum_{e \in \text{Cut}} \omega(e) \)
- Connectivity: \( \sum_{e \in \text{cut}} (\lambda - 1) \omega(e) \)
ε-Balanced Hypergraph Partitioning (HGP)

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 HGP Objectives:

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

# blocks connected by $e$
\( \varepsilon \)-Balanced Hypergraph Partitioning (HGP)

Partition hypergraph \( H = (V, E, c : V \to \mathbb{R}_{>0}, \omega : E \to \mathbb{R}_{>0}) \) 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 HGP Objectives:

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

\# blocks connected by \( e \)

\( \Rightarrow \) Both revert to edge-cut for graphs
Applications

- VLSI Design
- Warehouse Optimization
- Complex Networks
- Route Planning
- Scientific Computing

\[ \mathbb{R}^{n \times n} \ni Ax = b \in \mathbb{R}^n \]
High-Quality Hypergraph Partitioning
Successful Heuristic: Multilevel Paradigm

Coarsening

Input Hypergraph

match /

cluster

contract

...
Successful Heuristic: Multilevel Paradigm

Input Hypergraph

Coarsening

match /

cluster

contract

Initial Partitioning
Successful Heuristic: Multilevel Paradigm

Coarsening

- match / cluster
- contract

Uncoarsening

- local search
- uncontract

Input Hypergraph

Output Partition

Initial Partitioning
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

Coarsening

match / contract

cluster

local search

Uncoarsening

uncontract

Input Hypergraph

Initial Partitioning

Output Partition
Why Yet Another Multilevel Algorithm?

Tradeoff:

# levels ↗:
- + Quality
- – Running time

\[
\text{Karlsruhe Hypergraph Partitioning} \Rightarrow \textbf{Evade} \text{ tradeoff} \rightsquigarrow n \text{ levels} \ [\text{ALENEX'16}] \\
\Rightarrow \text{Combine high quality with good performance}
\]
KaHyPar: Novel Algorithmic Ingredients

Coarsening

Input Hypergraph

match /
contract
cluster

local search

uncontract

Output Partition

Initial Partitioning

Initial Partitioning

10  Sebastian Schlag – High-Quality (Hyper)Graph Partitioning
Institute of Theoretical Informatics
Algorithmics Group
KaHyPar: Novel Algorithmic Ingredients

Min-Hash Based Sparsification

[ALENEX'17]

Coarsening

match / cluster

contract

local search

uncontract

Output Partition

Initial Partitioning

Input Hypergraph

Initial Partitioning

Min-Hash Based Sparsification

KaHyPar: Novel Algorithmic Ingredients

[ALENEX'17]

Coarsening

match / cluster

contract

local search

uncontract

Output Partition

Initial Partitioning

Input Hypergraph

Initial Partitioning
KaHyPar: Novel Algorithmic Ingredients

Min-Hash Based Sparsification

[ALENEX’17]

Community-Aware Coarsening

[SEA’17]

Output Partition

local search

uncontract

contract

Initial Partitioning

Coarsening
KaHyPar: Novel Algorithmic Ingredients

- Min-Hash Based Sparsification [ALENEX'17]
- Community-Aware Coarsening [SEA'17]
- Fast $n$-Level Coarsening [ALENEX'16, ALENEX'17]

Coarsening → Output Partition

- local search
- uncontract

Input Hypergraph → Initial Partitioning

Initial Partitioning → Fast $n$-Level Coarsening

KaHyPar: Novel Algorithmic Ingredients

Institute of Theoretical Informatics
Algorithmics Group
KaHyPar: Novel Algorithmic Ingredients

Min-Hash Based Sparsification  
[ALENEX'17]

Community-Aware Coarsening  
[SEA'17]

Fast n-Level Coarsening  
[ALENEX'16, ALENEX'17]

Engineered k-way FM  
[ALENEX'17]
KaHyPar: Novel Algorithmic Ingredients

- **Min-Hash Based Sparsification**
  - [ALENEX'17]

- **Community-Aware Coarsening**
  - [SEA'17]

- **Fast n-Level Coarsening**
  - [ALENEX'16, ALENEX'17]

- **Max-Flow Min-Cut Refinement**
  - [SEA'18, JEA'19]

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

Diagram showing the process of KaHyPar with phases including initial partitioning, fast n-level coarsening, coarsening, contract, uncontract, output partition, and gain-cache of vertices.
KaHyPar: Novel Algorithmic Ingredients

- Min-Hash Based Sparsification [ALENEX'17]
- Community-Aware Coarsening [SEA'17]
- Fast $n$-Level Coarsening [ALENEX'16, ALENEX'17]
- Memetic Multilevel Algorithm [GECCO'18]
- Max-Flow Min-Cut Refinement [SEA'18, JEA'19]
- Engineered $k$-way FM [ALENEX'17]

Initial Partitioning → Fast $n$-Level Coarsening → Coarsening → Community-Aware Coarsening → Min-Hash Based Sparsification → Memetic Multilevel Algorithm → Max-Flow Min-Cut Refinement → Engineered $k$-way FM → Gain-Cache of $\omega(e)$
Experiments – Benchmark Setup

- System: 1 core of 2 Intel Xeon E5-2670 @ 2.6 Ghz, 64 GB RAM

- # Hypergraphs: [publicly available]
  - SuiteSparse Matrix Collection 184
  - SAT Competition 2014 (3 representations) 92
  - ISPD98 & DAC2012 VLSI Circuits 28

- \( k \in \{2, 4, 8, 16, 32, 64, 128\} \) with imbalance: \( \varepsilon = 3\% \)

- Comparing KaHyPar with:
  - hMetis-R & hMetis-K
  - PaToH-Default & PaToH-Quality
  - HYPE
  - Zoltan-AlgD
Experiments: Connectivity Optimization

⇒ Similar results for cut-net optimization
Parallel Shared-Memory Graph Partitioning
Parallel GP: Coarsening [EuroPar’18]

**Algorithm:** Parallel label propagation [SM’16] with improved load balancing
Parallel GP: Coarsening [EuroPar’18]

**Algorithm:** Parallel label propagation [SM’16] with improved load balancing

**Problem:** Vertex-based distribution $\Rightarrow$ bad load-balance

**Solution:** Edge-based distribution

- Packets $P$: $\sqrt{|E|} \leq \sum_{v \in P} d(v) \leq \sqrt{|E|} + \Delta$, $\Delta = \max_{v \in V} d(v)$
Parallel Initial Partitioning using KaHIP [SEA’14]

Select the best partition of $G$

Uncoarsening
Parallel GP: Refinement [EuroPar’18]

Algorithms:
- Parallel label propagation
- Parallel localized $k$-way local search
  - minimal coordination of searches
  - serialized execution of final moves
Experiments: Solution Quality
38 Graphs with $k = \{16, 64\}$
Experiments: Speedup & Running Time

4 Socket Machine with 79 Threads

Cumulative: \((x, y) \rightarrow \text{speedup/ running time for graphs with } |E| \geq x = y\)
Scalable Edge Partitioning
The Edge Partitioning Problem

Partition edge set of graph $G = (V, E, c, \omega)$ into $k$ disjoint blocks $\Pi = \{E_1, \ldots, E_k\}$ such that

- Blocks $E_i$ are roughly equal-sized:
  \[ \omega(E_i) \leq (1 + \varepsilon) \left\lceil \frac{\omega(E)}{k} \right\rceil \]

- minimize vertex cut:
  \[ \sum_{v \in V} |I(v)| - 1 \]
The Edge Partitioning Problem

Partition edge set of graph $G = (V, E, c, \omega)$ into $k$ disjoint blocks $\Pi = \{E_1, \ldots, E_k\}$ such that

- Blocks $E_i$ are roughly equal-sized:
  \[
  \omega(E_i) \leq (1 + \varepsilon) \left\lceil \frac{\omega(E)}{k} \right\rceil
  \]

- minimize vertex cut:
  \[
  \sum_{v \in V} |I(v)| - 1
  \]

# blocks with edges incident to $v$
The Edge Partitioning Problem

Partition edge set of graph $G = (V, E, c, \omega)$ into $k$ disjoint blocks $\Pi = \{E_1, \ldots, E_k\}$ such that

- Blocks $E_i$ are roughly equal-sized:
  $$\omega(E_i) \leq (1 + \varepsilon) \left\lceil \frac{\omega(E)}{k} \right\rceil$$

- minimize vertex cut:
  $$\sum_{v \in V} |I(v)| - 1$$

# blocks with edges incident to $v$

Motivation [Gonzalez et al.'12]:
- edge-centric distributed computations
- combat shortcomings of TLAV approaches
- duplicate node-centric computations
The Edge Partitioning Problem

Partition edge set of graph \( G = (V, E, c, \omega) \) into \( k \) disjoint blocks \( \Pi = \{E_1, \ldots, E_k\} \) such that

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

- Minimize vertex cut:
  \[ \sum_{v \in V} |I(v)| - 1 \]

**Motivation** [Gonzalez et al.'12]:
- edge-centric distributed computations
- combat shortcomings of TLAV approaches
- duplicate node-centric computations
Edge Partitioning Algorithms

Sequential

Quality

Running Time

KaHyPar

hMETIS

SPAC+

PaToH

SPAC+

Metis

NE

Sequential

Quality

Running Time

KaHyPar

hMETIS

SPAC+

PaToH

SPAC+

Metis

NE
Edge Partitioning Algorithms

Sequential

<table>
<thead>
<tr>
<th>Quality</th>
<th>Running Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>KaHyPar</td>
<td>hMETIS</td>
</tr>
<tr>
<td>PaToH</td>
<td>SPAC+</td>
</tr>
<tr>
<td>Metis</td>
<td></td>
</tr>
<tr>
<td>NE</td>
<td></td>
</tr>
</tbody>
</table>

Hypergraph Model:
- Graph edge $\sim$ vertex
- Graph node $\sim$ hyperedge
- Optimize connectivity
Edge Partitioning Algorithms

Sequential

Quality

KaHyPar

hMETIS

PaToH

SPAC+

KaHIP

NE

Split-And-Connect (SPAC) [Li et al.’17]:

- Build auxiliary graph
- Use vertex partitioning algorithm

Running Time

KaHyPar

hMETIS

PaToH

SPAC+

KaHIP

NE

SPAC+

Metis
Edge Partitioning Algorithms

Sequential

Quality

KaHyPar
hMETIS
SPAC+
PaToH
SPAC+
Metis
NE

Running Time

Distributed

Quality

Zoltan
JaBeJa-VC

Running Time
Edge Partitioning Algorithms

Sequential

Quality

KaHyPar

hMETIS

SPAC+

PaToH

KaHIP

SPAC++

Metis

NE

simple, greedy heuristics

Running Time

Distributed

Quality

Zoltan

JaBeJa-VC

Running Time

Ne: simple, greedy heuristics
Edge Partitioning Algorithms

Sequential

- Quality
  - KaHyPar
  - hMETIS
  - SPAC+
  - KaHIP
  - PaToH
  - SPAC+
  - Metis
  - NE

Running Time

Distributed

- Quality
  - Our Contributions [ALENEX’19]
    - dSPAC+
    - ParHIP-Eco
  - dSPAC+
  - ParMetis
  - SPAC+
  - Metis
  - KaHIP
  - hMETIS
  - KaHyPar

Running Time

Our Contributions

- Zoltan
- JaBeJa-VC
Experiments: Benchmark Setup

- Test suite: 70 graphs
  - Walshaw Graph Archive
  - Sparse Matrix-Vector Multiplication
  - Web & Social Graphs
  - Random Geometric Graphs
- \( k \in \{2, 4, 8, 16, 32, 64, 128\} \)
- Imbalance: \( \epsilon = 3\% \)
- Averages of 5 repetitions
- Sequential: 1 core
- Distributed: 32 * 20 cores

Competitors:

- KaHyPar-MF
- PaToH
- Zoltan
- Zoltan-AlgD
- hMetis-\{R, K\}
- JaBeJa-VC
- NE
- SPAC + KaHIP
- SPAC + Metis
- dSPAC + ParHIP
- dSPAC + ParMetis

HGP’s
Experiments: Sequential HGP

![Graph showing performance of different algorithms](image)

- hMetis-K
- PaToH
- hMetis-R
- Zoltan-AlgD
- KaHyPar
- Zoltan

fraction for which solver $\leq \tau \times$ best

$\tau$

1 1.05 1.1 1.5 2 10 100 inf.
Experiments: Sequential HGP & SPAC+X

fraction for which solver ≤ τ × best

JaBeJa-VC  SPAC+METIS
SPAC+KaHIP  NE
KaHyPar

τ

1 1.05 1.1 1.5 2 10 100 inf.
Experiments: Sequential Running Time

- hMETIS-{R, K}
- PaToH
- Zoltan[-AlgD]
- KaHyPar
- KaHIP
- METIS
- NE
- HGP
- SPAC
- EP

Running Time [s]

1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25

Sebastian Schlag – High-Quality (Hyper)Graph Partitioning
Experiments: Sequential Running Time

- HGP
- SPAC
- EP

KaHyPar

hMETIS-{R, K}
PaToH
Zoltan[-AlgD]
KaHyPar
KaHIP
METIS
NE

Running Time [s]
Experiments: Distributed HGP & dSPAC+X

![Graph showing performance of different solvers](image)

- **dSPAC+ParHIP-Eco**
- **dSPAC+ParHIP-Fast**
- **dSPAC+ParMETIS**
- **Zoltan**
Experiments: Distributed Running Time

- dSPAC
- dHGP

Running Time [s]

ParHIP-Fast, ParHIP-Eco, ParMETIS, Zoltan
Experiments: Distributed Running Time

- dSPAC
- dHGP

<table>
<thead>
<tr>
<th>ParHIP-Fast</th>
<th>ParHIP-Eco</th>
<th>ParMETIS</th>
<th>Zoltan</th>
</tr>
</thead>
</table>

Running Time [s]
Conclusion & Outlook

High-Quality Graph & Hypergraph Partitioning Frameworks:
- KaHIP – http://algo2.iti.kit.edu/kahip/
- KaHyPar – http://www.kahypar.org

Future Work:
- Shared-Memory HGP
- Distributed-Memory HGP
- Shift focus towards fast (H)GP algorithms with reasonable quality

(Personal) Open Questions:
- What would benefit the CSC community?
- What are "difficult" instances?