Improving Coarsening Schemes for Hypergraph Partitioning by Exploiting Community Structure

SEA’17 · June 23, 2017
Tobias Heuer and Sebastian Schlag
Hypergraphs

- Generalization of graphs
  ⇒ hyperedges connect ≥ 2 nodes

- Graphs ⇒ dyadic (2-ary) relationships
- Hypergraphs ⇒ (d-ary) relationships

- Hypergraph $H = (V, E, c, \omega)$
  - Vertex set $V = \{1, ..., n\}$
  - Edge set $E \subseteq \mathcal{P}(V) \setminus \emptyset$
  - Node weights $c : V \rightarrow \mathbb{R}_{\geq 1}$
  - Edge weights $\omega : E \rightarrow \mathbb{R}_{\geq 1}$
Hypergraphs

- Generalization of graphs
  ⇒ hyperedges connect ≥ 2 nodes

- Graphs ⇒ dyadic (2-ary) relationships
- Hypergraphs ⇒ (d-ary) relationships

Hypergraph \( H = (V, E, c, ω) \)
- Vertex set \( V = \{1, ..., n\} \)
- Edge set \( E \subseteq \mathcal{P}(V) \setminus \emptyset \)
- Node weights \( c : V \rightarrow \mathbb{R}_{\geq 1} \)
- Edge weights \( ω : E \rightarrow \mathbb{R}_{\geq 1} \)

\[ |P| = \sum_{e \in E} |e| = \sum_{v \in V} d(v) \]
Hypergraph Partitioning Problem

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

- blocks $V_i$ are **roughly equal-sized**:

$$c(V_i) \leq (1 + \varepsilon) \left\lceil \frac{c(V)}{k} \right\rceil$$
Hypergraph Partitioning Problem

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

- blocks $V_i$ are **roughly equal-sized**:

$$c(V_i) \leq (1 + \varepsilon) \left\lceil \frac{c(V)}{k} \right\rceil$$
Hypergraph Partitioning Problem

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

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

- connectivity objective is **minimized**:

  ![Hypergraph Partitioning Diagram]
Hypergraph Partitioning Problem

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

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

- **connectivity** objective is minimized:
  \[ \sum_{e \in \text{cut}} (\lambda - 1) \omega(e) \]

**imbalance parameter**

**connectivity:**

# blocks connected by net $e$
Hypergraph Partitioning Problem

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

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

- connectivity objective is **minimized**:
  \[ \sum_{e \in \text{cut}} (\lambda - 1) \omega(e) = 6 \]

$\lambda$ is the number of blocks connected by a net $e$. 

Equation: 
\[ \sum_{e \in \text{cut}} (\lambda - 1) \omega(e) = 6 \]
Applications

VLSI Design

facilitate floorplanning & placement

Scientific Computing

Application Domain

Hypergraph Model

Goal

minimize communication
The Multilevel Framework

- input hypergraph
- contract
- match
- cluster

- output partition
- uncontract
- local search

- initial partitioning
This Talk: Coarsening Phase

input hypergraph

match / cluster

contract

output partition

local search

uncontract

initial partitioning

initial partitioning
Clustering-based Coarsening

Common Strategy: **avoid** global decisions $\leadsto$ **local**, greedy algorithms

**Objective:** identify highly connected vertices

```plaintext
foreach vertex v do
  cluster[v] := argmax rating(v,u) among neighbor u
```

(Visual representation of vertex clustering using hypergraph.)
Clustering-based Coarsening

Common Strategy: **avoid** global decisions \(\leadsto\) **local**, greedy algorithms

**Objective:** identify highly connected vertices

```
foreach vertex \(v\) do
  cluster[\(v\)] := \text{argmax} \ \text{rating}(\(v, u\))
```

**Main Design Goals:** [Karypis,Kumar 99]

1: reduce **size** of nets \(\leadsto\) easier local search
2: reduce **number** of nets \(\leadsto\) easier initial partitioning
3: maintain **structural similarity** \(\leadsto\) good coarse solutions
Clustering-based Coarsening

Main Design Goals:
1: reduce size of nets $\leadsto$ easier local search
2: reduce number of nets $\leadsto$ easier initial partitioning
Clustering-based Coarsening

Main Design Goals:
1: reduce size of nets \( \leadsto \) easier local search
2: reduce number of nets \( \leadsto \) easier initial partitioning

hypergraph-tailored rating functions:

\[
r(u, v) := \sum_{\text{net } e \text{ containing } u, v} \frac{\omega(e)}{|e|-1}
\]
Clustering-based Coarsening

Main Design Goals:

1: reduce **size** of nets $\leadsto$ easier local search

2: reduce **number** of nets $\leadsto$ easier initial partitioning

$\Rightarrow$ hypergraph-tailored rating functions:

$$r(u, v) := \sum_{\text{net } e \text{ containing } u, v} \frac{\omega(e)}{|e| - 1}$$

large number...
Clustering-based Coarsening

Main Design Goals:
1: reduce **size** of nets $\leadsto$ easier local search
2: reduce **number** of nets $\leadsto$ easier initial partitioning

$\mathbf{r}(u,v) := \sum_{\text{net } e \text{ containing } u,v} \frac{\omega(e)}{|e|-1}$

$\Rightarrow$ hypergraph-tailored rating functions:
Clustering-based Coarsening

Main Design Goals:
1: reduce size of nets \(\leadsto\) easier local search
2: reduce number of nets \(\leadsto\) easier initial partitioning

\[
r(u, v) := \sum_{\text{net } e \text{ containing } u, v} \frac{\omega(e)}{|e| - 1}
\]

\(\leadsto\) hypergraph-tailored rating functions:

- of heavy nets ...
- ... with small size
- large number ...
- containing \(u, v\)
Clustering-based Coarsening

Main Design Goals:
1: reduce size of nets \(\leadsto\) easier local search
2: reduce number of nets \(\leadsto\) easier initial partitioning

\[ r(u, v) := \sum_{\text{net } e \text{ containing } u, v} \frac{\omega(e)}{|e| - 1} \]

\(r(u, v)\) of heavy nets ...
\(\sum\) of heavy nets ...

... with small size
... with small size

large number ...
large number ...

hypergraph-tailored rating functions:
Clustering-based Coarsening

Main Design Goals:
1: reduce **size** of nets $\rightsquigarrow$ easier local search
2: reduce **number** of nets $\rightsquigarrow$ easier initial partitioning

$\Rightarrow$ hypergraph-tailored rating functions:

$$r(u, v) := \sum_{\text{net } e \text{ containing } u, v} \frac{\omega(e)}{|e|-1}$$

- of heavy nets ...
- ... with small size
- large number ...
- of heavy nets ...

3: maintain **structural similarity** $\rightsquigarrow$ good coarse solutions
   - prefer clustering over matching
   - ensure $\sim$balanced vertex weights
Clustering-based Coarsening

Main Design Goals:
1: reduce size of nets $\leadsto$ easier local search
2: reduce number of nets $\leadsto$ easier initial partitioning

\[ r(u, v) := \sum_{\text{net } e \text{ containing } u, v} \frac{\omega(e)}{|e| - 1} \]

hypergraph-tailored rating functions:
- large number ...
- of heavy nets ...
- ... with small size

3: maintain structural similarity $\leadsto$ good coarse solutions
   - prefer clustering over matching
   - ensure $\sim$ balanced vertex weights

enough?
What could possibly go wrong?

... a lot:

input

\[ \{u, v\} \]

obscured clusters
What could possibly go wrong?

... a lot:

- **issus**
  - maximal matching
  - prefer unclustered
  - random tie-breaking
  - heavy neighbors

- obscured clusters
  - \( \{u, v\} \)
  - input
  - \( n_1 \)
What could possibly go wrong?

... a lot:

ISSUES

maximal matching

prefer unclustered

Problem: relying only on local information!
Our Approach: Community-aware Coarsening

input

\{u, v\}

obscured clusters
Our Approach: Community-aware Coarsening

\[ \{u, v\} \]

SOLUTION

community detection
Our Approach: Community-aware Coarsening

**Framework:**
- preprocessing: determine **community structure**
- only allow **intra-community** contractions

![Diagram showing community detection and obscured clusters](image-url)
Our Approach: Community-aware Coarsening

Framework:
- preprocessing: determine **community structure**
- only allow **intra-community** contractions
Detecting Community Structure

**Goal:** partition graph into **natural** groups $C$

Community: internally **dense**, externally **sparse** subgraph
Detecting Community Structure

**Goal:** partition graph into **natural** groups $C$

(One) **Formalization:** [Newman,Girvan 04]

**Modularity** $\text{mod}(G, C) := \text{cov}(G, C) - \mathbb{E}[\text{cov}(G, C)]$

**Coverage** $\text{cov}(G, C) := \sum_{C \in C} \frac{|E(C)|}{|E|}$

Community: internally **dense**, externally **sparse** subgraph

fraction of intra-cluster edges
Detecting Community Structure

**Goal:** partition graph into **natural** groups $C$

![Graph diagram]

**Community:** internally **dense**, externally **sparse** subgraph

*(One) Formalization*: [Newman,Girvan 04]

**Modularity** $\text{mod}(G, C) := \text{cov}(G, C) - E[\text{cov}(G, C)]$

**Coverage** $\text{cov}(G, C) := \sum_{C \in C} \frac{|E(C)|}{|E|}$

Efficient Heuristic: Louvain Method (multilevel, local, greedy) [Blondel et al. 08]

- repeatedly move nodes to neighbor communities
- coarsen graph & repeat
Graph Representations of Hypergraphs

Hypergraph
Graph Representations of Hypergraphs

- destroys natural sparsity: \( \left( \frac{|e|}{2} \right) \) edges per net \( e \)
- exaggerates importance of large nets
Graph Representations of Hypergraphs

Hypergraph

Clique Graph

- destroys natural sparsity: \(|e|/2\) edges per net \(e\)
- exaggerates importance of large nets

Bipartite (Star) Graph

+ compact representation: \(\mathcal{O}(|P|)\) space
- nets become nodes & part of clustering
Bipartite Graphs: Modeling Peculiarities

Density: \( d := \frac{m}{n} = \frac{|P|/n}{|P|/m} = \frac{d(v)}{|e|} \)
Bipartite Graphs: Modeling Peculiarities

Density: $d := \frac{m}{n} = \frac{|P|/n}{|P|/m} = \frac{d(v)}{|e|}$

$\text{d} \approx 1$

$|V| \approx |E|$

Hypernodes ($\top$-nodes)
Hyperedges ($\perp$-nodes)
Bipartite Graphs: Modeling Peculiarities

Density: \( d := \frac{m}{n} = \frac{|P|/n}{|P|/m} = \frac{d(v)}{|e|} \)

\[
\begin{align*}
|V| & \approx |E| \\
\overline{d(v)} & \approx |e| \\
\end{align*}
\]

\[
\begin{align*}
|V| & \ll |E| \\
\overline{d(v)} & \gg |e| \\
\end{align*}
\]

Hypernodes (\( \top \)-nodes)

Hyperedges (\( \bot \)-nodes)
Bipartite Graphs: Modeling Peculiarities

Density: \( d := \frac{m}{n} = \frac{|P|/n}{|P|/m} = \frac{d(v)}{|e|} \)

- \( d \ll 1 \) : \( |V| \gg |E| \)
- \( d \approx 1 \) : \( |V| \sim |E| \)
- \( d \gg 1 \) : \( |V| \ll |E| \)

Hypernodes (\( \top \)-nodes)
Hyperedges (\( \perp \)-nodes)

\( \bar{d}(v) \ll |e| \)
\( \bar{d}(v) \approx |e| \)
\( \bar{d}(v) \gg |e| \)
Bipartite Graphs: Modeling Peculiarities

\[ d \ll 1 \quad |V| \gg |E| \]

Hypernodes (\(\top\)-nodes)

\[ d \approx 1 \quad |V| \simeq |E| \]

Hyperedges (\(\bot\)-nodes)

\[ d \gg 1 \quad |V| \ll |E| \]

addressed via edge weights:
Bipartite Graphs: Modeling Peculiarities

\[ d \ll 1 \quad \Rightarrow \quad |V| \gg |E| \]

\[ d \approx 1 \quad \Rightarrow \quad |V| \approx |E| \]

\[ d \gg 1 \quad \Rightarrow \quad |V| \ll |E| \]

Hypernodes (\(\top\)-nodes)

Hyperedges (\(\perp\)-nodes)

\[ d(v) \ll |e| \]

\[ d(v) \approx |e| \]

\[ d(v) \gg |e| \]

\[ \omega(v, e) := 1 \quad \text{baseline} \]
Bipartite Graphs: Modeling Peculiarities

\[
\begin{align*}
|V| & \gg |E| & |V| & \approx |E| & |V| & \ll |E| \\
\bar{d}(v) & \ll |e| & \bar{d}(v) & \approx |e| & \bar{d}(v) & \gg |e|
\end{align*}
\]

Hypernodes (\(\top\)-nodes)

Hyperedges (\(\bot\)-nodes)

\(\omega(v, e) := 1\) baseline

\(\omega_e(v, e) := \frac{1}{|e|}\) small nets \(\rightsquigarrow\) higher influence

addressed via \textbf{edge weights:}
Bipartite Graphs: Modeling Peculiarities

- **Hypernodes (⊤-nodes)**
  - $d \ll 1$
  - $|V| \gg |E|$
  - $d(v) \ll |e|$

- **Hyperedges (⊥-nodes)**
  - $d \approx 1$
  - $|V| \approx |E|$
  - $d(v) \approx |e|$

- **Hypernodes (⊤-nodes)**
  - $d \gg 1$
  - $|V| \ll |E|$
  - $d(v) \gg |e|$

$\omega(v, e) := 1$

- **Baseline**

$\omega_e(v, e) := \frac{1}{|e|}$

- **Small nets $\leadsto$ higher influence**

$\omega_{de}(v, e) := \frac{d(v)}{|e|}$

- **+ high degree $\leadsto$ higher influence**

$\Rightarrow$ addressed via **edge weights**:
Experiments – Benchmark Setup

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

- Hypergraphs:¹

<table>
<thead>
<tr>
<th>Application</th>
<th>VLSI</th>
<th>Sparse Matrix</th>
<th>SAT Solving</th>
</tr>
</thead>
<tbody>
<tr>
<td>Benchmark Set</td>
<td>ISPD98</td>
<td>UF-SPM</td>
<td>SAT14</td>
</tr>
<tr>
<td>Representation</td>
<td>direct</td>
<td>row-net</td>
<td>literal</td>
</tr>
<tr>
<td>Density Class</td>
<td>$d \approx 1$</td>
<td>$d \ll 1$, $d \approx 1$, $d \gg 1$</td>
<td>$d \gg 1$</td>
</tr>
<tr>
<td>Community Str.</td>
<td>✓</td>
<td>some instances</td>
<td>✓</td>
</tr>
<tr>
<td># Hypergraphs</td>
<td>18</td>
<td>184</td>
<td>92</td>
</tr>
</tbody>
</table>

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

- 8 hours time limit / instance

- Comparing KaHyPar-CA with:
  - KaHyPar-K
  - hMetis-R & hMetis-K
  - PaToH-Default & PaToH-Quality

¹ available @ https://algo2.iti.kit.edu/schlag/sea2017/
Comparison of Edge Weighting Schemes

- $d \ll 1$
- $d \approx 1$
- $d \gg 1$

Initial Cut Improvement [%]

Avg. Cut Improvement [%]

Min Cut Improvement [%]

$\text{CA}(\omega)$ $\text{CA}(\omega_e)$ $\text{CA}(\omega_{de})$ $\text{CA}(\omega)$ $\text{CA}(\omega_e)$ $\text{CA}(\omega_{de})$ $\text{CA}(\omega)$ $\text{CA}(\omega_e)$ $\text{CA}(\omega_{de})$
Comparison of Edge Weighting Schemes

$\begin{align*}
d &\ll 1 \\
$\begin{align*}
\text{Initial Cut Improv.} &\quad \text{Avg. Cut Improv.} \\
\text{Min Cut Improv.} &
\end{align*}
\end{align*}
$\begin{align*}
d &\approx 1 \\
\end{align*}$
$\begin{align*}
\text{Initial Cut Improv.} &\quad \text{Avg. Cut Improv.} \\
\text{Min Cut Improv.} &
\end{align*}$
$\begin{align*}
d &\gg 1 \\
\end{align*}$

\begin{align*}
\text{CA}(\omega) \quad \text{CA}(\omega_e) \quad \text{CA}(\omega_{de})
\end{align*}
Comparison of Edge Weighting Schemes

\[
\omega_{de}(v, e) := \frac{d(v)}{|e|}
\]

- Initial Cut Improv. [%]
- Avg. Cut Improv. [%]
- Min Cut Improv. [%]

\(d \ll 1\)
\(d \approx 1\)
\(d \gg 1\)
Comparison of Edge Weighting Schemes

- $\omega_{de}(v, e) := \frac{d(v)}{|e|}$

$\Delta$:
- Initial Cut Improv. [%]
- Avg. Cut Improv. [%]
- Min Cut Improv. [%]

$d \ll 1$
- $\Delta > 0$
- $\Delta < 0$

$d \approx 1$
- $\Delta > 0$
- $\Delta < 0$

$d \gg 1$
- $\Delta > 0$
- $\Delta < 0$
Comparison of Edge Weighting Schemes

$d \ll 1$

$\omega_{de}(v, e) := \frac{d(v)}{|e|}$

$\omega(v, e) := 1$

$d \approx 1$

$d \gg 1$

$\omega_{de}(v, e) := \frac{d(v)}{|e|}$

CA($\omega$) CA($\omega_{e}$) CA($\omega_{de}$)

CA($\omega$) CA($\omega_{e}$) CA($\omega_{de}$)

CA($\omega$) CA($\omega_{e}$) CA($\omega_{de}$)
Comparison of Edge Weighting Schemes

\[ \omega_{de}(v, e) := \frac{d(v)}{|e|} \]

\[ \omega(v, e) := 1 \]

\[ d \ll 1 \]
\[ d \approx 1 \]
\[ d \gg 1 \]
Comparison of Edge Weighting Schemes

\[ \omega_{de}(v, e) := \frac{d(v)}{|e|} \]

\[ \omega(v, e) := 1 \]

- Initial Cut Improv. [%]
- Avg. Cut Improv. [%]
- Min Cut Improv. [%]
Experimental Results – Partitioning Quality

Example

![Graph showing the partitioning quality for different algorithms. The graph plots the difference between the current partitioning and the best possible partitioning against the number of instances. The lines represent different algorithms: Algorithm 1 (red), Algorithm 2 (blue), and Algorithm 3 (green). The graph demonstrates the performance improvement of the algorithms as the number of instances increases.]
Experimental Results – Partitioning Quality

Example

1 - $\frac{1}{\text{Algorithm}}$

- Algorithm 1
- Algorithm 2
- Algorithm 3

# Instances
Experimental Results – Partitioning Quality

Example

1 − Best Algorithm

1.00
0.80
0.60
0.40
0.20
0.10
0.05
0.01
0.00

# Instances

1

100
200
300
400
600
800
1000
1200
1600
2000

Algorithm 1
Algorithm 2
Algorithm 3
Experimental Results – Partitioning Quality

Example

\[ 1 - \frac{\text{Best Algorithm}}{\text{Algorithm}} \]

# Instances

- Algorithm 1
- Algorithm 2
- Algorithm 3
Experimental Results – Partitioning Quality

Example

$1 - \frac{\text{Algorithm}}{\text{Best Algorithm}}$

$\# \text{ Instances}$

- Algorithm 1
- Algorithm 2
- Algorithm 3
Experimental Results - Quality

All Instances

$d \approx 1 :$ DAC2012

$d \approx 1 :$ ISPD98

$d \gg 1 :$ SAT14 primal
Experimental Results - Quality

\( d \gg 1 : \text{SAT14 literal} \)

\( d \ll 1 : \text{SAT14 dual} \)

Sparse Matrices

Web Social

1-(Best/Algorithm)

1-(Best/Algorithm)
## Experimental Results - Running Time

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Running Time [s]</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>All</td>
</tr>
<tr>
<td>KaHyPar</td>
<td>20.4</td>
</tr>
<tr>
<td>KaHyPar-CA</td>
<td>31.0</td>
</tr>
<tr>
<td>hMetis-R</td>
<td>79.2</td>
</tr>
<tr>
<td>hMetis-K</td>
<td>57.9</td>
</tr>
<tr>
<td>PaToH-Q</td>
<td>5.9</td>
</tr>
<tr>
<td>PaToH-D</td>
<td>1.2</td>
</tr>
<tr>
<td>Algorithm</td>
<td>Running Time [s]</td>
</tr>
<tr>
<td>-------------</td>
<td>-----------------</td>
</tr>
<tr>
<td></td>
<td>All</td>
</tr>
<tr>
<td>KaHyPar</td>
<td>20.4</td>
</tr>
<tr>
<td>KaHyPar-CA</td>
<td>31.0</td>
</tr>
<tr>
<td>hMetis-R</td>
<td>79.2</td>
</tr>
<tr>
<td>hMetis-K</td>
<td>57.9</td>
</tr>
<tr>
<td>PaToH-Q</td>
<td>5.9</td>
</tr>
<tr>
<td>PaToH-D</td>
<td>1.2</td>
</tr>
</tbody>
</table>
Conclusion & Discussion

KaHyPar-CA - Community-aware Coarsening

Community Detection via:
- modularity maximization
- Louvain Method (LM)
- bipartite, weighted graph

Future Work:
- speedup preprocessing: parallel LM
- resolution limit $\leadsto$ multi-resolution modularity
- other formalizations:
  - Infomap
  - Surprise

KaHyPar-Framework
Open-Source on Github:
https://git.io/vMBaR
References

[Newman, Girvan 04]

[Blondel et al. 08]

[Karypis, Kumar 99]
Taxonomy of Hypergraph Partitioning Tools

Recursive Bisection
- MLPart
- PaToH
- Sparse Matrices
- Mondriaan
- Zoltan (parallel)

Direct k-way
- hMetis-R
- VLSI
- hMetis-K
- Parkway
- kPaToH (multi-constraint)
- UMPa (multi-objective)
- KaHyPar-R (n-Level)
- KaHyPar-K (n-Level)

Years:
- 1998
- 1999
- 2005
- 2006
- 2008
- 2013
- 2016
- 2017