Using Steiner Trees in Hypergraph Partitioning

August 15, 2023
Tobias Heuer

Institute of Theoretical Informatics, Algorithm Engineering
Hypergraphs

- Generalization of graphs
  ⇒ Edge can connect **more than 2 nodes**

- Hypergraph \( H = (V, E, c, \omega) \)
  - Node set \( V = \{1, \ldots, n\} \)
  - Hyperedge set \( E \subseteq \mathcal{P}(V) \setminus \emptyset \) (also called *nets*)
  - Node weights \( c : V \rightarrow \mathbb{R}_{\geq 1} \)
  - Edge weights \( \omega : E \rightarrow \mathbb{R}_{\geq 1} \)

- Node \( v \in e \) is also called a **pin**
- Number of pins \( p := \sum_{e \in E} |e| \)
Hypergraphs

- Generalization of graphs
  ⇒ Edge can connect more than 2 nodes

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

- Node $v \in e$ is also called a pin
- Number of pins $p := \sum_{e \in E} |e|$
Hypergraphs

- Generalization of graphs
  \[ \Rightarrow \text{Edge can connect more than 2 nodes} \]

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

- Node \( v \in e \) is also called a pin
- Number of pins \( p := \sum_{e \in E} |e| \)
\( \varepsilon \)-Balanced 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**:
  \[
  c(V_i) \leq (1 + \varepsilon) \left\lceil \frac{c(V)}{k} \right\rceil
  \]
ε-Balanced 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:**

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

imbalance parameter
**ε-Balanced 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:

  \[
  \sum_{e \in E} (\lambda(e) - 1) \, \omega(e)
  \]

  # blocks connected by hyperedge \( e \)
ε-Balanced 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:
  \[ \sum_{e \in E} (\lambda(e) - 1) \omega(e) \]

\( \lambda(e) = 3 \)
ε-Balanced 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:
  \[
  c(V_i) \leq (1 + \epsilon) \lceil c(V) / k \rceil
  \]

- Connectivity objective is minimized:
  \[
  \sum_{e \in E} (\lambda(e) - 1) \omega(e)
  \]

Hypergraph Partitioning is NP-hard \[\text{Len90}\] ⇒ Heuristic solutions are used in practice

Heuristic solutions are used in practice

Hypergraph Partitioning is NP-hard \[\text{Len90}\]
VLSI Design

Logical Circuit
VLSI Design

Logical Circuit

VLSI Design
Find a realization of a logical circuit on a physical layout
VLSI Design

Find a realization of a logical circuit on a physical layout
VLSI Design

Find a realization of a logical circuit on a physical layout

Logical Circuit

Cell Locations

Routing channels for wires
VLSI Design
VLSI Design

Logical Circuit

Routing Phase
VLSI Design

Routing is a 3-dimensional problem

Layer 1
Layer 2
Layer 3

Routing Phase
Common Objectives in VLSI Design:
- Wire-Length Minimization
- Signal Delay/Timing Minimization
- Power Consumption Minimization
Common Objectives in VLSI Design:

- **Wire-Length Minimization**
- Signal Delay/Timing Minimization
- Power Consumption Minimization

Wire-Length Minimization implicitly minimizes:

- Signal Delays
- Power Consumption
- Total Layout Area
Placement via Hypergraph Partitioning
Placement via Hypergraph Partitioning
Placement via Hypergraph Partitioning

Logical Circuit

Hypergraph Model
Placement via Hypergraph Partitioning
Placement via Hypergraph Partitioning

Physical Layout

Hypergraph Model

Bipartitioning
Placement via Hypergraph Partitioning

Physical Layout

Assign cells to subregions based on bipartition

Hypergraph Model

Bipartitioning
Placement via Hypergraph Partitioning

Assign cells to subregions based on bipartition

Bipartitioning
Placement via Hypergraph Partitioning

Physical Layout

Assign cells to subregions based on bipartition

Hypergraph Model

Hyperpartitioning
Placement via Hypergraph Partitioning

Subregions are small enough
⇒ Use end case placer
Placement via Hypergraph Partitioning

Physical Layout

Routing Phase

Hypergraph Model
Placement via Hypergraph Partitioning

- Physical Layout
- Hypergraph Model

Cut-Net/Connectivity Minimization $\neq$ Wire-Length Minimization
Placement via Hypergraph Partitioning

Our Goal: Whatever Objective Function = Wire-Length Minimization ✓
KaHyPar – Github Issue #126

Region 1
Region 2

$e_1$

Region 3
Region 4

Region 1
Region 2

$e_2$

Region 3
Region 4
KaHyPar – Github Issue #126

\[ \lambda(e_1) = 2 = 2 = \lambda(e_2) \]
KaHyPar – Github Issue #126

wire-length(e₁) < wire-length(e₂)
Can we encode structural properties of the routing layout into the objective function?
Process Mapping

Graph

Distributed Computing Cluster

PE 1

PE 2

PE 3

PE 4

PE 5
Process Mapping

Graph

Distributed Computing Cluster

Communication Link
Process Mapping

Graph

Distributed Computing Cluster

Communication Link

Communication Cost
Process Mapping
Process Mapping

Graph

Distributed Computing Cluster

PE 1

PE 2

PE 3

PE 4

PE 5
Process Mapping

Graph

Distributed Computing Cluster

minimize \[ \sum_{\{u,v\} \in E} \text{DIST}(u, v) \cdot \omega(u, v) \]

Shortest distance between PEs to which \( u \) and \( v \) are assigned
Process Mapping

What is the shortest distance induced by a hyperedge on the target graph?

minimize \[ \sum_{\{u,v\} \in E} \text{DIST}(u, v) \cdot \omega(u, v) \]
The Steiner Tree Packing Problem

Given

- a weighted graph $G = (V, E, \omega)$
- $N$ terminal sets $T_1, \ldots, T_N \subseteq V$
The Steiner Tree Packing Problem

Given
- a weighted graph \( G = (V, E, \omega) \)
- \( N \) terminal sets \( T_1, \ldots, T_N \subseteq V \)

The Steiner tree packing problem asks for
- \( N \) edge-disjoint trees \( S_1, \ldots, S_N \subseteq E \)

Routing layout
- hyperedges
- wires
The Steiner Tree Packing Problem

Given
- a weighted graph $G = (V, E, \omega)$
- $N$ terminal sets $T_1, \ldots, T_N \subseteq V$

The Steiner tree packing problem asks for
- $N$ edge-disjoint trees $S_1, \ldots, S_N \subseteq E$

such that
- each $S_i$ spans $T_i$
- $\sum_{i=1}^{N} \omega(S_i)$ is minimal

routing layout
hyperedges
wire-length minimization
The Steiner Tree Packing Problem

Given
- a weighted graph $G = (V, E, \omega)$
- $N$ terminal sets $T_1, \ldots, T_N \subseteq V$

The Steiner tree packing problem asks for
- $N$ edge-disjoint trees $S_1, \ldots, S_N \subseteq E$

such that
- each $S_i$ spans $T_i$
- $\sum_{i=1}^{N} \omega(S_i)$ is minimal

Problem Variations
- **Steiner tree problem:**
  for a single terminal set $T \subseteq V$ $\Rightarrow$ **NP-hard**
- **Minimum spanning tree problem**
  $T = V \Rightarrow$ solvable in polynomial time
The Steiner Tree Packing Problem

Given
- a weighted graph $G = (V, E, \omega)$
- $N$ terminal sets $T_1, \ldots, T_N \subseteq V$

The Steiner tree packing problem asks for
- $N$ edge-disjoint trees $S_1, \ldots, S_N \subseteq E$

such that
- each $S_i$ spans $T_i$
- $\sum_{i=1}^{N} \omega(S_i)$ is minimal

Routing layouts are large!
Can we reduce the complexity of the problem?

Problem Variations
- Steiner tree problem:
  - for a single terminal set $T \subseteq V \Rightarrow$ NP-hard
- Minimum spanning tree problem
  - $T = V \Rightarrow$ solvable in polynomial time
Global Routing

Finding a feasible routing is typically decomposed into two steps: global and detailed routing.
Global Routing

Finding a feasible routing is typically decomposed into two steps: global and detailed routing.
Global Routing

Finding a feasible routing is typically decomposed into two steps: **global** and **detailed** routing.
Global Routing

Finding a feasible routing is typically decomposed into two steps: *global* and *detailed* routing.
Global Routing

Finding a feasible routing is typically decomposed into two steps: **global** and **detailed** routing.

**Capacity** \( c(u, v) = \) maximum number of wires that can cross the border.
Global Routing

Finding a feasible routing is typically decomposed into two steps: **global** and **detailed** routing.

**Distance** \( \omega(u, v) = \) distance between centers of both regions.

**Capacity** \( c(u, v) = \) maximum number of wires that can cross the border.
Global Routing

Finding a feasible routing is typically decomposed into two steps: **global** and **detailed** routing.

**Distance** \( \omega(u, v) = \) distance between centers of both regions

**Capacity** \( c(u, v) = \) maximum number of wires that can cross the border

Given:
- a weighted graph \( G = (V, E, c, \omega) \)
- \( N \) terminal sets \( T_1, \ldots, T_N \subseteq V \)

The **global Steiner tree packing problem** asks for
- \( N \) edge-disjoint trees \( S_1, \ldots, S_N \subseteq E \) such that
  - each \( S_i \) spans \( T_i \)
  - \( \forall e \in E : |\{ S_i | e \in S_i \}| \leq c(e) \)
  - \( \sum_{i=1}^{N} \omega(S_i) \) is minimal

Routing layout
- hyperedges
- wires
- each wire connects its cells

Capacity constraint
- wire-length minimization
The Steiner Tree Metric

Given
- a weighted hypergraph $H = (V, E, c, \omega)$
- a weighted target graph $G = (V, \mathcal{E}, d)$
- $d(u, v) := \text{distance or cost of edge } \{u, v\} \in \mathcal{E}$
The Steiner Tree Metric

Given
- a weighted hypergraph $H = (V, E, c, \omega)$
- a weighted target graph $G = (V, \mathcal{E}, d)$
  - $d(u, v) := \text{distance or cost of edge } \{u, v\} \in \mathcal{E}$

we want to find
- an $\epsilon$-balanced mapping $\Pi : V \rightarrow V$
The Steiner Tree Metric

Given

- a weighted hypergraph $H = (V, E, c, \omega)$
- a weighted target graph $G = (V, \mathcal{E}, d)$
- $d(u, v) := \text{distance or cost of edge } \{u, v\} \in \mathcal{E}$

we want to find

- an $\epsilon$-balanced mapping $\Pi : V \rightarrow V$

such that the Steiner tree metric

$$f_{\text{ST}} := \sum_{e \in E} \text{DIST}(\Lambda(e)) \cdot \omega(e)$$

is minimized.
The Steiner Tree Metric

Given
- a weighted hypergraph \( H = (V, E, c, \omega) \)
- a weighted target graph \( G = (V, \mathcal{E}, d) \)
- \( d(u, v) := \text{distance or cost of edge } \{u, v\} \in \mathcal{E} \)

we want to find
- an \( \epsilon \)-balanced mapping \( \Pi : V \to V \)

such that the Steiner tree metric

\[
f_{ST} := \sum_{e \in E} \text{DIST}(\Lambda(e)) \cdot \omega(e)
\]

is minimized.

\( \text{DIST}(\Lambda(e)) \) is the weight of the optimal Steiner Tree connecting the blocks \( \Lambda(e) \) spanned by \( e \) on \( G \).
Evaluating $f_{ST}$

- **Bad News**: Evaluating $f_{ST}$ requires to solve an **NP-hard** problem for each hyperedge.
Evaluating $f_{\text{ST}}$

- **Bad News**: Evaluating $f_{\text{ST}}$ requires to solve an **NP-hard** problem for each hyperedge.
- **Good News**: VLSI instances have **many small** hyperedges and only a few really large hyperedges.
Evaluating $f_{ST}$

- **Bad News**: Evaluating $f_{ST}$ requires to solve an **NP-hard** problem for each hyperedge.
- **Good News**: VLSI instances have many small hyperedges and only a few really large hyperedges.

**Idea**: Precompute all Steiner trees up to a certain size $t$.

- If $\lambda(e) \leq t$ ⇒ use precomputed Steiner tree
- If $\lambda(e) > t$
  - Compute 2-approximation ⇒ MST computation in metric completion of $G$
  - Cache MST computation in a hash table for subsequent retrievals

$O(k^3 + k^2(2^t - t) + k(3^t - 2^{t+1} + 3))$

$O(\lambda(e)^2 + \lambda(e) \log \lambda(e))$
Evaluating $f_{\text{ST}}$

- **Bad News**: Evaluating $f_{\text{ST}}$ requires to solve an **NP-hard** problem for each hyperedge.
- **Good News**: VLSI instances have many small hyperedges and only a few really large hyperedges.

**Idea**: Precompute all Steiner trees up to a certain size $t = 4$

- If $\lambda(e) \leq t \Rightarrow$ use precomputed Steiner tree
- If $\lambda(e) > t$
  - Compute 2-approximation $\Rightarrow$ MST computation in metric completion of $G$
  - Cache MST computation in a hash table for subsequent retrievals

$O(k^3 + k^2(2^t - t) + k(3^t - 2^{t+1} + 3))$

$O(\lambda(e)^2 + \lambda(e) \log \lambda(e))$
Evaluating $f_{ST}$

**Bad News:** Evaluating $f_{ST}$ requires to solve an NP-hard problem for each hyperedge.

**Good News:** VLSI instances have many small hyperedges and only a few really large hyperedges.

**Idea:** Precompute all Steiner trees up to a certain size $t$
- If $\lambda(e) \leq t \Rightarrow$ use precomputed Steiner tree
- If $\lambda(e) > t$
  - Compute 2-approximation
  - Cache MST computation in a hash table for subsequent retrievals

\[
O(k^3 + k^2(2^t - t) + k(3^t - 2t^t + 3))
\]

\[
O(\lambda(e)^2 + \lambda(e) \log \lambda(e))
\]
Evaluating $f_{\text{ST}}$

- **Bad News**: Evaluating $f_{\text{ST}}$ requires to solve an NP-hard problem for each hyperedge.

- **Good News**: VLSI instances have many small hyperedges and only a few really large hyperedges.

**Idea**: Precompute all Steiner trees up to a certain size $t$

- If $\lambda(e) \leq t$ ⇒ use precomputed Steiner tree
- If $\lambda(e) > t$
  - Compute 2-approximation ⇒ MST computation in metric completion of $G$
  - Cache MST computation in a hash table for subsequent retrievals

$$O(k^3 + k^2(2^t - t) + k(3^t - 2^{t+1} + 3))$$

$$O(\lambda(e)^2 + \lambda(e) \log \lambda(e))$$

99.1% (median)
Evaluating $f_{ST}$

- **Bad News**: Evaluating $f_{ST}$ requires to solve an NP-hard problem for each hyperedge.
- **Good News**: VLSI instances have many small hyperedges and only a few really large hyperedges.

**Idea**: Precompute all Steiner trees up to a certain size $t$.
- If $\lambda(e) \leq t$, use precomputed Steiner tree.
- If $\lambda(e) > t$, compute 2-approximation.
- Cache MST computation in a hash table for subsequent retrievals.

$O(k^3 + k^2(2^t - t) + k(3^t - 2^{t+1} + 3) + k(3t - 2t + 1 + 3))$

If $\lambda(e) > t$, 99.8% (median) of the queries are **cache hits**.

99.1% (median)
**Evaluating $f_{ST}$**

- **Bad News:** Evaluating $f_{ST}$ requires to solve an NP-hard problem for each hyperedge.
- **Good News:** VLSI instances have many small hyperedges and only a few really large hyperedges.

**Idea:** Precompute all Steiner trees up to a certain size $t$.

- If $\lambda(e) \leq t \Rightarrow$ use precomputed Steiner tree.
- If $\lambda(e) > t$

  - Compute 2-approximation $\Rightarrow$ MST computation in metric completion of $G$.
  - Cache MST computation in a hash table for subsequent retrievals.

\[
\text{Time} = O(k^3 + k^2(2^t - t) + k(3^t - 2^{t+1} + 3))
\]

\[
\text{Space} = O(\lambda(e)^2 + \lambda(e) \log \lambda(e))
\]

- If $\lambda(e) > t$, 99.8% (median) of the queries are cache hits.
- Largest cache miss rate is 6.9% (median).

99.1% (median)
Multilevel Scheme

Input Hypergraph

Coarsening

contract

cluster
Multilevel Scheme

Input Hypergraph

Coarsening

Cluster

Contract

Initial Partitioning
Multilevel Scheme

Coarsening

- Input Hypergraph
- Cluster
- Contract
- Initial Partitioning

Uncoarsening

- Local Search
- Uncontract

1. August 15, 2023
2. Tobias Heuer – Using Steiner Trees in Hypergraph Partitioning
3. Institute of Theoretical Informatics, Algorithm Engineering
Mt-KaHyPar: Algorithmic Components

Coarsening

- Input Hypergraph
- Contract
- Cluster

Initial Partitioning

Uncoarsening

- Local search
- Uncontract
Mt-KaHyPar: Algorithmic Components

Multilevel Coarsening [ALENEX’21]

$n$-level Coarsening [ALENEX’22]

Input Hypergraph

Uncoarsening

Initial Partitioning
Mt-KaHyPar: Algorithmic Components

Multilevel Coarsening [ALENEX'21]

\[ n \text{-level Coarsening [ALENEX'22]} \]

Parallel Recursive Bipartitioning with Work-Stealing [ALENEX'21]

\[ k = 4 \]

Input Hypergraph

Multilevel Coarsening [ALENEX'21]

\[ n \text{-level Coarsening [ALENEX'22]} \]

Parallel Recursive Bipartitioning with Work-Stealing [ALENEX'21]

\[ k = 4 \]

Task Queue

Thread 1: [C1, B2, B2, B2, B2]
Thread 2: [C1, B2, B2, B2, B2]
Thread 3: [C1, B2, B2, B2, B2]
Thread 4: [C1, B2, B2, B2, B2]

Task Queue

Thread 1: [C1, B2, B2, B2, B2]
Thread 2: [C1, B2, B2, B2, B2]
Thread 3: [C1, B2, B2, B2, B2]
Thread 4: [C1, B2, B2, B2, B2]

Input Hypergraph

Multilevel Coarsening [ALENEX'21]

\[ n \text{-level Coarsening [ALENEX'22]} \]

Parallel Recursive Bipartitioning with Work-Stealing [ALENEX'21]

\[ k = 4 \]

Task Queue

Thread 1: [C1, B2, B2, B2, B2]
Thread 2: [C1, B2, B2, B2, B2]
Thread 3: [C1, B2, B2, B2, B2]
Thread 4: [C1, B2, B2, B2, B2]

Task Queue

Thread 1: [C1, B2, B2, B2, B2]
Thread 2: [C1, B2, B2, B2, B2]
Thread 3: [C1, B2, B2, B2, B2]
Thread 4: [C1, B2, B2, B2, B2]

Input Hypergraph

Multilevel Coarsening [ALENEX'21]

\[ n \text{-level Coarsening [ALENEX'22]} \]

Parallel Recursive Bipartitioning with Work-Stealing [ALENEX'21]

\[ k = 4 \]

Task Queue

Thread 1: [C1, B2, B2, B2, B2]
Thread 2: [C1, B2, B2, B2, B2]
Thread 3: [C1, B2, B2, B2, B2]
Thread 4: [C1, B2, B2, B2, B2]

Task Queue

Thread 1: [C1, B2, B2, B2, B2]
Thread 2: [C1, B2, B2, B2, B2]
Thread 3: [C1, B2, B2, B2, B2]
Thread 4: [C1, B2, B2, B2, B2]
Initial Partitioning: Two-Phase Approach
Initial Partitioning: Two-Phase Approach
Initial Partitioning: Two-Phase Approach

- Find Initial Mapping
- Compute Initial Partition (optimizing connectivity metric)
Initial Partitioning: Two-Phase Approach

Find Initial Mapping

Compute Initial Partition (optimizing connectivity metric)

Contraction
Initial Partitioning: Two-Phase Approach

Find Initial Mapping

Compute Initial Partition (optimizing connectivity metric)

One-to-One Process Mapping Problem (OPMP)

Contraction
Initial Partitioning: Two-Phase Approach

Find Initial Mapping

Compute Initial Partition (optimizing connectivity metric)

One-to-One Process Mapping Problem (OPMP)

Greedy Mapping + Local Search

Contraction
Mt-KaHyPar: Algorithmic Components

Multilevel Coarsening [ALENEX'21]

Input Hypergraph

n-level Coarsening [ALENEX'22]

Uncoarsening

Parallel Recursive Bipartitioning with Work-Stealing [ALENEX'21]

$k = 4$

Parallel Recursion
Mt-KaHyPar: Algorithmic Components

**Multilevel Coarsening [ALENEX’21]**

**n-level Coarsening [ALENEX’22]**

**Parallel Recursive Bipartitioning with Work-Stealing [ALENEX’21]**

**k-Way FM Algorithm [ALENEX’21]**

**Flow-Based Refinement [SEA’22]**
Experimental Setup

Machine
- Intel Xeon Gold, 2 sockets, 20 cores @ 2.1 Ghz, 96 GB RAM

Benchmark Set
- 150 hypergraphs: [publicly available]
  - ISPD98 18
  - DAC2012 VLSI Circuits 10
  - Titan23 Benchmark Suite 22
  - SuiteSparse Matrix Collection 50
  - SAT Competition 2014 50
Experimental Setup

Machine
- Intel Xeon Gold, 2 sockets, 20 cores @ 2.1 Ghz, 96 GB RAM

Benchmark Set
- 150 hypergraphs: [publicly available]
  - ISPD98 18
  - DAC2012 VLSI Circuits 10
  - Titan23 Benchmark Suite 22
  - SuiteSparse Matrix Collection 50
  - SAT Competition 2014 50
Experimental Setup

Machine
- Intel Xeon Gold, 2 sockets, 20 cores @ 2.1 Ghz, 96 GB RAM

Benchmark Set
- 150 hypergraphs: [publicly available]
  - ISPD98: 18
  - DAC2012 VLSI Circuits: 10
  - Titan23 Benchmark Suite: 22
  - SuiteSparse Matrix Collection: 50
  - SAT Competition 2014: 50

Input Parameters
- Imbalance $\varepsilon = 3\%$
- Mapping onto complete rectangular $N \times M$ grid graphs
- Edge weights are choosen uniformly at random between 1 and 10
- 5 random seeds
Experimental Setup

Machine
- Intel Xeon Gold, 2 sockets, 20 cores @ 2.1 Ghz, 96 GB RAM

Benchmark Set
- 150 hypergraphs: [publicly available]
  - ISPD98 18
  - DAC2012 VLSI Circuits 10
  - Titan23 Benchmark Suite 22
  - SuiteSparse Matrix Collection 50
  - SAT Competition 2014 50

Input Parameters
- Imbalance $\varepsilon = 3\%$
- Mapping onto complete rectangular $N \times M$ grid graphs
- Edge weights are chosen uniformly at random between 1 and 10
- 5 random seeds

Algorithm Configuration
- Mt-KaHyPar-D (fast configuration)
- Mt-KaHyPar-Q (Mt-KaHyPar-D + flow-based refinement)
Experimental Setup

Machine
- Intel Xeon Gold, 2 sockets, 20 cores @ 2.1 Ghz, 96 GB RAM

Benchmark Set
- 150 hypergraphs: [publicly available]
  - ISPD98
  - DAC2012 VLSI Circuits
  - Titan23 Benchmark Suite
  - SuiteSparse Matrix Collection
  - SAT Competition 2014

Input Parameters
- Imbalance $\varepsilon = 3\%$
- Mapping onto complete rectangular $N \times M$ grid graphs
- Edge weights are chosen uniformly at random between 1 and 10
- 5 random seeds

Algorithm Configuration
- Mt-KaHyPar-D (fast configuration)
- Mt-KaHyPar-Q (Mt-KaHyPar-D + flow-based refinement)

Competitors
- Two-Phase Approach (2P)
- Mt-KaHyPar-D$_{2P}$, Mt-KaHyPar-Q$_{2P}$, $k$KaHyPar$_{2P}$, hMetis-R$_{2P}$

4 nodes
8 nodes
16 nodes
32 nodes
64 nodes
Experimental Results
Experimental Results

\[ p_{\text{Algo}}(\tau) = \frac{|\{I \in \mathcal{I} \mid \text{Algo}(I) \leq \tau \cdot \text{Best}(I)\}|}{|\mathcal{I}|} \]
Experimental Results

\[ p_{\text{Algo}}(\tau) = \frac{|\{ I \in \mathcal{I} \mid \text{Algo}(I) \leq \tau \cdot \text{Best}(I)\}|}{|\mathcal{I}|} \]

Benchmark Set

Quality produced by algorithm for instance \( I \)

Quality of best solution produced by any algorithm for instance \( I \)

Fraction of Instances

Quality Relative to Best [\( \tau \)]
Experimental Results

\[ p_{\text{Algo}}(\tau) = \frac{|\{ I \in \mathcal{I} \mid \text{Algo}(I) \leq \tau \cdot \text{Best}(I)\}|}{|\mathcal{I}|} \]

For \( \tau = 1 \) (x-axis), the plot shows the fraction of instances (y-axis) where an algorithm performs best.
Experimental Results

\[ p_{\text{Algo}}(\tau) = \left| \{ I \in \mathcal{I} \mid \text{Algo}(I) \leq \tau \cdot \text{Best}(I) \} \right| / |\mathcal{I}| \]

For \( \tau = 1.05 \) (x-axis), the plot shows the fraction of instances (y-axis) for which an algorithm is at most 5\% worse than the best solution.
**Experimental Results**

\[ p_{\text{Algo}}(\tau) = \frac{|\{ I \in \mathcal{I} \mid \text{Algo}(I) \leq \tau \cdot \text{Best}(I)\}|}{|\mathcal{I}|} \]

![Graph showing experimental results](image)
Experimental Results

No improvement over two-phase approaches for small target graphs with 4 and 8 nodes

$k \in \{4, 8\}$

![Graph showing fraction of instances vs quality relative to best (τ)](image)

Legend:
- Mt-KaHyPar-D
- Mt-KaHyPar-D_{2p}
- Mt-KaHyPar-Q
- Mt-KaHyPar-Q_{2p}
- hMetis-R_{2p}
- kKaHyPar_{2p}
Experimental Results

$k \in \{16, 32, 64\}$

![Graph showing experimental results with curves for different values of $k$.]

- Red: Mt-KaHyPar-D
- Green: Mt-KaHyPar-D_{2P}
- Blue: Mt-KaHyPar-Q
- Violet: Mt-KaHyPar-Q_{2P}
- Orange: hMetis-R_{2P}
- Yellow: $k$ KaHyPar_{2P}
Experimental Results

Median improvement of Mt-KaHyPar-Q over Algorithm Med. Impr. [%]

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Med. Impr. [%]</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mt-KaHyPar-D&lt;sub&gt;2P&lt;/sub&gt;</td>
<td>10</td>
</tr>
<tr>
<td>Mt-KaHyPar-Q&lt;sub&gt;2P&lt;/sub&gt;</td>
<td>5.5</td>
</tr>
<tr>
<td>hMetis-R&lt;sub&gt;2P&lt;/sub&gt;</td>
<td>5.2</td>
</tr>
<tr>
<td>Mt-KaHyPar-D</td>
<td>4.5</td>
</tr>
<tr>
<td>kKaHyPar&lt;sub&gt;2P&lt;/sub&gt;</td>
<td>3.8</td>
</tr>
</tbody>
</table>

$k \in \{16, 32, 64\}$
Experimental Results

Median improvement of Mt-KaHyPar-Q over Algorithm Med. Impr. [%]

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Med. Impr. [%]</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mt-KaHyPar-D₂ₚ</td>
<td>10</td>
</tr>
<tr>
<td>Mt-KaHyPar-Q₂ₚ</td>
<td>5.5</td>
</tr>
<tr>
<td>hMetis-R₂ₚ</td>
<td>5.2</td>
</tr>
<tr>
<td>Mt-KaHyPar-D</td>
<td>4.5</td>
</tr>
<tr>
<td>kKaHyPar₂ₚ</td>
<td>3.8</td>
</tr>
</tbody>
</table>

Median improvement of Mt-KaHyPar-Q over Algorithm Med. Impr. [%]

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Med. Impr. [%]</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mt-KaHyPar-D₂ₚ</td>
<td>0.8</td>
</tr>
<tr>
<td>Mt-KaHyPar-D</td>
<td>2.6</td>
</tr>
<tr>
<td>Mt-KaHyPar-Q₂ₚ</td>
<td>2.7</td>
</tr>
<tr>
<td>Mt-KaHyPar-Q</td>
<td>6.5</td>
</tr>
<tr>
<td>kKaHyPar₂ₚ</td>
<td>3.8</td>
</tr>
<tr>
<td>hMetis-R₂ₚ</td>
<td>59.2</td>
</tr>
<tr>
<td>hMetis-R₂ₚ</td>
<td>71.1</td>
</tr>
</tbody>
</table>

$k \in \{16, 32, 64\}$
Experimental Results

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Med. Impr. [%]</th>
<th>Algorithm</th>
<th>gmean t[s]</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mt-KaHyPar-D&lt;sub&gt;2P&lt;/sub&gt;</td>
<td>10</td>
<td>Mt-KaHyPar-D&lt;sub&gt;2P&lt;/sub&gt;</td>
<td>0.8</td>
</tr>
<tr>
<td>Mt-KaHyPar-Q&lt;sub&gt;2P&lt;/sub&gt;</td>
<td>5.5</td>
<td>Mt-KaHyPar-D</td>
<td>2.6</td>
</tr>
<tr>
<td>hMetis-R&lt;sub&gt;2P&lt;/sub&gt;</td>
<td>5.2</td>
<td>Mt-KaHyPar-Q&lt;sub&gt;2P&lt;/sub&gt;</td>
<td>2.7</td>
</tr>
<tr>
<td>Mt-KaHyPar-D</td>
<td>4.5</td>
<td>kKaHyPar&lt;sub&gt;2P&lt;/sub&gt;</td>
<td>6.5</td>
</tr>
<tr>
<td>kKaHyPar&lt;sub&gt;2P&lt;/sub&gt;</td>
<td>3.8</td>
<td>hMetis-R&lt;sub&gt;2P&lt;/sub&gt;</td>
<td>59.2</td>
</tr>
</tbody>
</table>

Median improvement of Mt-KaHyPar-Q over Algorithm gmean t[s]

- Mt-KaHyPar-D<sub>2P</sub> 10
- Mt-KaHyPar-Q<sub>2P</sub> 5.5
- hMetis-R<sub>2P</sub> 5.2
- Mt-KaHyPar-D 4.5
- kKaHyPar<sub>2P</sub> 3.8

Slowdown:
- Mt-KaHyPar-D: 3.2
- Mt-KaHyPar-Q: 2.3
**Experimental Results**

Median improvement of Mt-KaHyPar-Q over Algorithm Med. Impr. [%]

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Med. Impr. [%]</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mt-KaHyPar-D&lt;sub&gt;2P&lt;/sub&gt;</td>
<td>10</td>
</tr>
<tr>
<td>Mt-KaHyPar-Q&lt;sub&gt;2P&lt;/sub&gt;</td>
<td>5.5</td>
</tr>
<tr>
<td>hMetis-R&lt;sub&gt;2P&lt;/sub&gt;</td>
<td>5.2</td>
</tr>
<tr>
<td>Mt-KaHyPar-D&lt;sup&gt;2&lt;/sup&gt;</td>
<td>4.5</td>
</tr>
<tr>
<td>kKaHyPar&lt;sub&gt;2P&lt;/sub&gt;</td>
<td>3.8</td>
</tr>
</tbody>
</table>

Median improvement of Mt-KaHyPar-Q over Algorithm Med. Impr. [%]

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Med. Impr. [%]</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mt-KaHyPar-D&lt;sub&gt;2P&lt;/sub&gt;</td>
<td>10</td>
</tr>
<tr>
<td>Mt-KaHyPar-D&lt;sup&gt;2&lt;/sup&gt;</td>
<td>4.5</td>
</tr>
<tr>
<td>kKaHyPar&lt;sub&gt;2P&lt;/sub&gt;</td>
<td>3.8</td>
</tr>
</tbody>
</table>

Algorithm gmean t[s]

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>gmean t[s]</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mt-KaHyPar-D&lt;sub&gt;2P&lt;/sub&gt;</td>
<td>0.8</td>
</tr>
<tr>
<td>Mt-KaHyPar-D&lt;sup&gt;2&lt;/sup&gt;</td>
<td>2.6</td>
</tr>
<tr>
<td>hMetis-R&lt;sub&gt;2P&lt;/sub&gt;</td>
<td>5.2</td>
</tr>
<tr>
<td>Mt-KaHyPar-Q&lt;sub&gt;2P&lt;/sub&gt;</td>
<td>2.7</td>
</tr>
<tr>
<td>Mt-KaHyPar-Q&lt;sup&gt;2&lt;/sup&gt;</td>
<td>6.5</td>
</tr>
<tr>
<td>kKaHyPar&lt;sub&gt;2P&lt;/sub&gt;</td>
<td>3.8</td>
</tr>
<tr>
<td>hMetis-R&lt;sub&gt;2P&lt;/sub&gt;</td>
<td>59.2</td>
</tr>
<tr>
<td>kKaHyPar&lt;sub&gt;2P&lt;/sub&gt;</td>
<td>71.1</td>
</tr>
</tbody>
</table>

Almost order of magnitude faster
## Experimental Results

### Median improvement of \( \text{Mt-KaHyPar-Q} \) over \( k\text{KaHyPar}_{2P} \)

<table>
<thead>
<tr>
<th>( k )</th>
<th>( \text{VLSI} )</th>
<th>( \text{SPM} )</th>
<th>( \text{SAT} )</th>
<th>( \text{ALL} )</th>
</tr>
</thead>
<tbody>
<tr>
<td>4</td>
<td>-1.1</td>
<td>0</td>
<td>-0.8</td>
<td>-0.7</td>
</tr>
<tr>
<td>8</td>
<td>0.5</td>
<td>1.3</td>
<td>0.6</td>
<td>0.9</td>
</tr>
<tr>
<td>16</td>
<td>4.2</td>
<td>1.7</td>
<td>2.1</td>
<td>2.9</td>
</tr>
<tr>
<td>32</td>
<td>5.5</td>
<td>3.7</td>
<td>3.4</td>
<td>4.7</td>
</tr>
<tr>
<td>64</td>
<td>7</td>
<td>2.7</td>
<td>3</td>
<td>4.4</td>
</tr>
</tbody>
</table>

### Median improvement of \( \text{Mt-KaHyPar-Q} \) over \( \text{Mt-KaHyPar-D} \) and \( \text{hMetis-R} \) over \( k\text{KaHyPar}_{2P} \)

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Med. Impr. [%]</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mt-KaHyPar-D(_{2P})</td>
<td>10</td>
</tr>
<tr>
<td>Mt-KaHyPar-Q(_{2P})</td>
<td>5.5</td>
</tr>
<tr>
<td>hMetis-R(_{2P})</td>
<td>5.2</td>
</tr>
</tbody>
</table>

### Median improvement of \( \text{Mt-KaHyPar-Q} \) over \( \text{Mt-KaHyPar-D} \) and \( \text{hMetis-R} \) over \( k\text{KaHyPar}_{2P} \)

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>gmean t[s]</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mt-KaHyPar-D(_{2P})</td>
<td>0.8</td>
</tr>
<tr>
<td>Mt-KaHyPar-Q(_{2P})</td>
<td>2.6</td>
</tr>
<tr>
<td>hMetis-R(_{2P})</td>
<td>2.7</td>
</tr>
<tr>
<td>( k\text{KaHyPar}_{2P})</td>
<td>6.5</td>
</tr>
<tr>
<td>Mt-KaHyPar-D</td>
<td>59.2</td>
</tr>
<tr>
<td>Mt-KaHyPar-Q</td>
<td>71.1</td>
</tr>
</tbody>
</table>

### Median improvement of \( \text{Mt-KaHyPar-Q} \) over \( k\text{KaHyPar}_{2P} \)

<table>
<thead>
<tr>
<th>( k )</th>
<th>( \text{VLSI} )</th>
<th>( \text{SPM} )</th>
<th>( \text{SAT} )</th>
<th>( \text{ALL} )</th>
</tr>
</thead>
<tbody>
<tr>
<td>4</td>
<td>-1.1</td>
<td>0.5</td>
<td>4.2</td>
<td>2.9</td>
</tr>
<tr>
<td>8</td>
<td>0</td>
<td>1.3</td>
<td>1.7</td>
<td>4.7</td>
</tr>
<tr>
<td>16</td>
<td>4.2</td>
<td>1.7</td>
<td>2.1</td>
<td>2.9</td>
</tr>
<tr>
<td>32</td>
<td>5.5</td>
<td>3.7</td>
<td>3.4</td>
<td>4.7</td>
</tr>
<tr>
<td>64</td>
<td>7</td>
<td>2.7</td>
<td>3</td>
<td>4.4</td>
</tr>
</tbody>
</table>

### Graphical representation of experimental results
Experimental Results

Median improvement of $\text{Mt-KaHyPar-Q}$ over $k\text{KaHyPar}_{2P}$

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Med. Impr. [%]</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mt-KaHyPar-D$_{2P}$</td>
<td>10</td>
</tr>
<tr>
<td>Mt-KaHyPar-Q$_{2P}$</td>
<td>5.5</td>
</tr>
<tr>
<td>hMetis-R$_{2P}$</td>
<td>5.2</td>
</tr>
<tr>
<td>Mt-KaHyPar-D</td>
<td>4.5</td>
</tr>
<tr>
<td>$k\text{KaHyPar}_{2P}$</td>
<td>3.8</td>
</tr>
</tbody>
</table>

Median improvement of $\text{Mt-KaHyPar-Q}$ over $k\text{KaHyPar}_{2P}$ for $k = 4, 8, 16, 32, 64$

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Med. Impr. [%]</th>
</tr>
</thead>
<tbody>
<tr>
<td>VLSI</td>
<td>$-1.1$</td>
</tr>
<tr>
<td>SPM</td>
<td>$0$</td>
</tr>
<tr>
<td>SAT</td>
<td>$-0.8$</td>
</tr>
<tr>
<td>ALL</td>
<td>$-0.7$</td>
</tr>
</tbody>
</table>

Median improvement of $\text{Mt-KaHyPar-D}_2P$ over $\text{Mt-KaHyPar-Q}_2P$:

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>gmean t[s]</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\text{Mt-KaHyPar-D}_2P$</td>
<td>0.8</td>
</tr>
<tr>
<td>$\text{Mt-KaHyPar-Q}_2P$</td>
<td>2.7</td>
</tr>
<tr>
<td>$k\text{KaHyPar}_{2P}$</td>
<td>59.2</td>
</tr>
<tr>
<td>hMetis-R$_{2P}$</td>
<td>71.1</td>
</tr>
</tbody>
</table>

Improvements most pronounced on VLSI instances and larger target graphs.
Conclusion

Contributions

- **Generalization** of the **process mapping problem** from graphs to hypergraphs
- HGP formulation that **accurately models wire-lengths** in the global routing problem
- **First** direct $k$-way multilevel mapping algorithm for optimizing the Steiner tree metric
Conclusion

Contributions

- **Generalization** of the process mapping problem from graphs to hypergraphs
- HGP formulation that **accurately models wire-lengths** in the global routing problem
- **First** direct $k$-way multilevel mapping algorithm for optimizing the Steiner tree metric

Future Work

- Mapping hypergraphs onto **larger targets graphs**
  - Compute Steiner trees on-demand
  - Replace Steiner tree metric with an MST-based metric
  - Global multisectioning
Conclusion

Contributions

- **Generalization** of the *process mapping problem* from graphs to hypergraphs
- HGP formulation that *accurately models wire-lengths* in the global routing problem
- **First** direct $k$-way multilevel mapping algorithm for optimizing the Steiner tree metric

Future Work

- Mapping hypergraphs onto *larger targets graphs*
  - Compute Steiner trees on-demand
  - Replace Steiner tree metric with an MST-based metric
  - Global multisectioning
- **Global routing algorithm** based on our algorithm that leverages placement and routing in one step