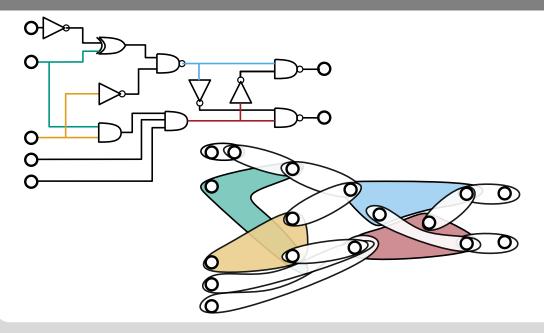# High Quality Hypergraph Partitioning via Max-Flow-Min-Cut Computations

Master Thesis · February 16, 2018
Tobias Heuer

INSTITUTE OF THEORETICAL INFORMATICS · ALGORITHMICS GROUP

Outline

Task

Developing a **refinement** algorithm based on **Max-Flow-Min-Cut** computations for the *n*-level hypergraph partitioner **KaHyPar**.

Institute of Theoretical Informatics
Algorithmics Group

Outline

Task

Developing a **refinement** algorithm based on **Max-Flow-Min-Cut** computations for the n-level hypergraph partitioner **KaHyPar**.
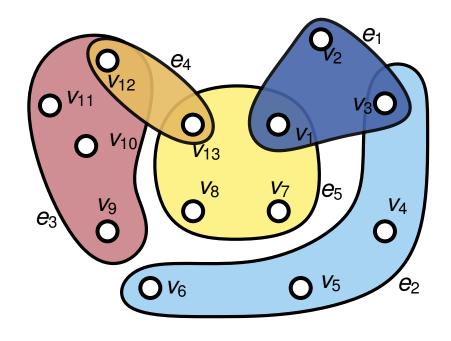
Contributions

- Outperforms 5 different systems on 73% of 3216 benchmark instances

- Improve quality of *KaHyPar* by 2.5%, while only incurring a slowdon by a factor of 2

- Comparable running time to *hMetis* and outperforms it on 84% of the instances

Institute of Theoretical Informatics
Algorithmics Group

Hypergraphs

- Generalization of graphs
  $\Rightarrow$ hyperedges connect ≥ 2 nodes

- Graphs $\Rightarrow$ dyadic (**2-ary**) relationships

- Hypergraphs $\Rightarrow$ (**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 $\omega : E \rightarrow \mathbb{R}_{\geq 1}$

Institute of Theoretical Informatics
Algorithmics Group

Hypergraphs

- Generalization of graphs
  $\Rightarrow$ hyperedges connect ≥ 2 nodes

- Graphs $\Rightarrow$ dyadic (**2-ary**) relationships

- Hypergraphs $\Rightarrow$ (**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 $c : V \to \mathbb{R}_{\geq 1}$
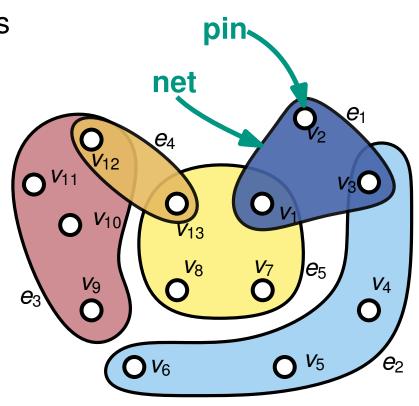 - Edge weights $\omega : E \to \mathbb{R}_{\geq 1}$

- $|P| = \sum_{e \in E} |e| = \sum_{v \in V} d(v)$

Institute of Theoretical Informatics
Algorithmics Group

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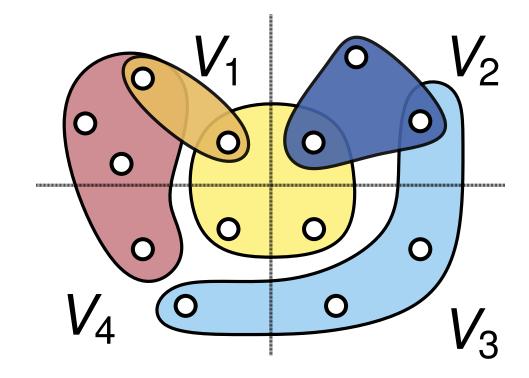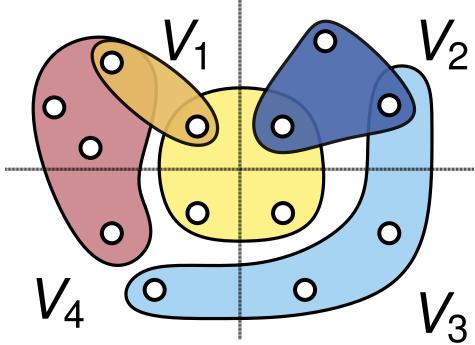
Partition hypergraph $H = (V, E, c, \omega)$ into k non-empty disjoint blocks $\Pi = \{V_1, \ldots, V_k\}$ such that:

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

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

Institute of Theoretical Informatics
Algorithmics Group

Hypergraph Partitioning Problem

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

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

imbalance parameter

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

V_1 V_2 V_3 V_4

Institute of Theoretical Informatics
Algorithmics Group

Hypergraph Partitioning Problem

[from SEA'17]

Partition hypergraph $H = (V, E, c, \omega)$ into k non-empty disjoint blocks $\Pi = \{V_1, \ldots, V_k\}$ such that:
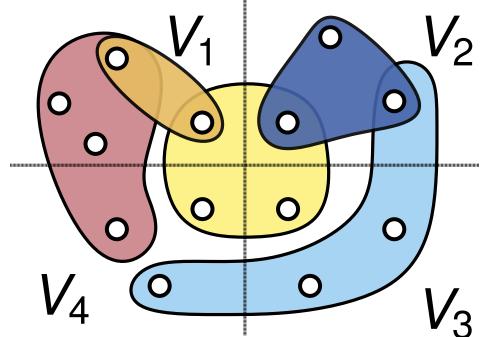
- blocks V_i are **roughly equal-sized**:

imbalance parameter

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

- **connectivity** objective is **minimized**:

V_1 V_2 V_4 V_3

Tobias Heuer – High Quality Hypergraph Partitioning via Max-Flow-Min-Cut Computations

Institute of Theoretical Informatics
Algorithmics Group

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

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

imbalance parameter

$$c(V_i) \leq (1 + \varepsilon) \left\lceil \frac{c(V)}{k} \right\rceil$$
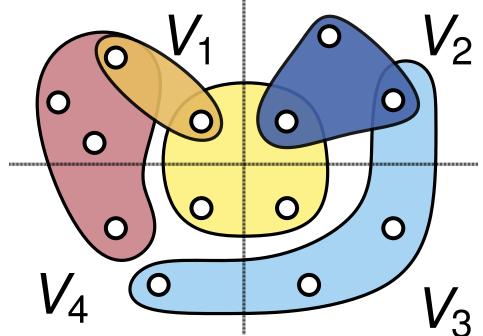
- **connectivity** objective is **minimized**:

$$\sum_{e \in \text{cut}} (\lambda - 1)\, \omega(e)$$
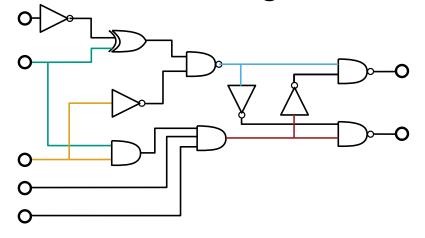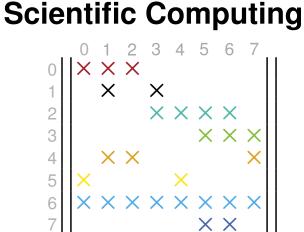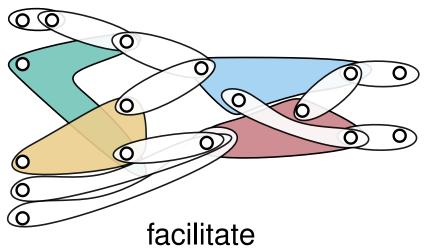
connectivity:
 # blocks connected by net e

Institute of Theoretical Informatics
Algorithmics Group

Hypergraph Partitioning Problem

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

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

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

imbalance parameter

- **connectivity** objective is **minimized**:

$$\sum_{e \in \text{cut}} (\lambda - 1)\, \omega(e) = 6$$

connectivity:
 # blocks connected by net e

Institute of Theoretical Informatics
Algorithmics Group

Applications

VLSI Design

Scientific Computing
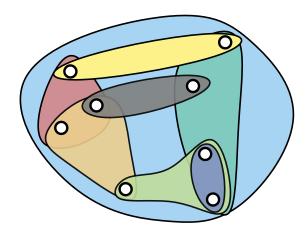
Application Domain

Hypergraph Model

Goal

facilitate floorplanning & placement
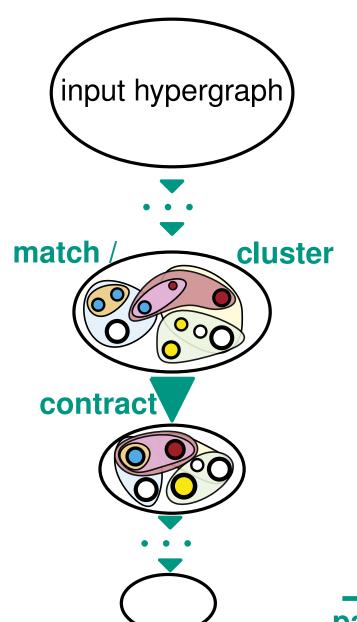
minimize communication

Institute of Theoretical Informatics
Algorithmics Group
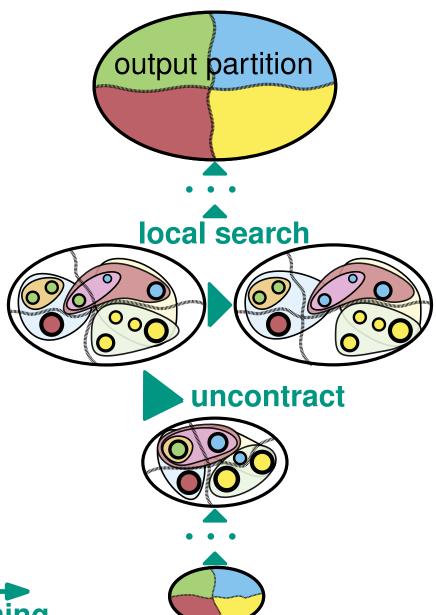
The Multilevel Framework
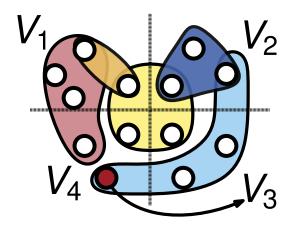
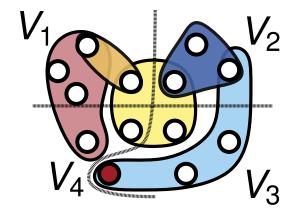Institute of Theoretical Informatics
Algorithmics Group

FM Algorithm

- **Move**-based heuristic that **greedily** move vertices between blocks based on **local** informations of incident nets
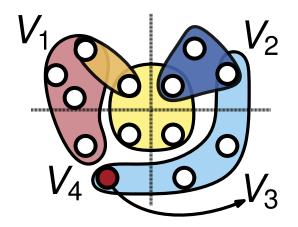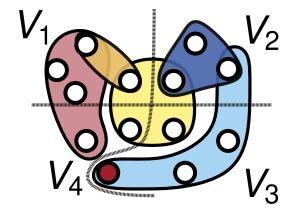

Moving ● from V_4 to V_3 reduces cut by 1

FM Algorithm

■ **Move**-based heuristic that **greedily** move vertices between blocks based on **local** informations of incident nets
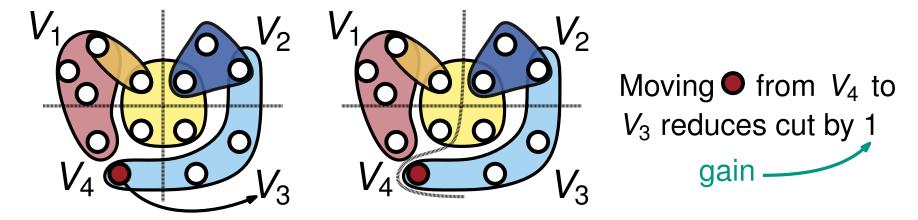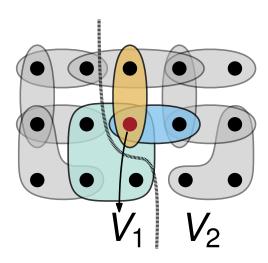

Moving ● from V_4 to V_3 reduces cut by 1

gain

Institute of Theoretical Informatics
Algorithmics Group

FM Algorithm

■ **Move**-based heuristic that **greedily** move vertices between blocks based on **local** informations of incident nets

Moving ● from V_4 to V_3 reduces cut by 1

gain

■ Performs moves of vertices with **maximum gain** in each step

■ All modern hypergraph partitioners implements variations of the *FM* algorithm
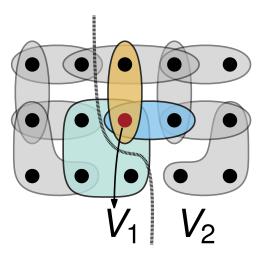
Institute of Theoretical Informatics
Algorithmics Group

FM Algorithm - Disadvantages

- Only incorparates **local** informations about the problem structure

  - Heavily depends on *initial partition*

  - In multilevel context: Depends on quality of *coarsening*

Institute of Theoretical Informatics
Algorithmics Group

FM Algorithm - Disadvantages

- Only incorparates **local** informations about the problem structure

  - Heavily depends on *initial partition*
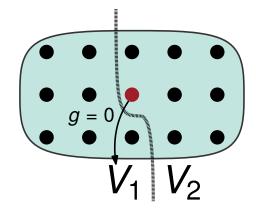
  - In multilevel context: Depends on quality of *coarsening*

- Large hyperedges induce **Zero-Gain** moves

  - Quality mainly depends on random decisions made within the algorithm

Institute of Theoretical Informatics
Algorithmics Group

Flow-based Approaches

Tobias Heuer – High Quality Hypergraph Partitioning via Max-Flow-Min-Cut Computations

Institute of Theoretical Informatics
Algorithmics Group

Flows

Given a graph $G = (V, E, u)$ and two nodes $s, t \in V$

- $u : E \rightarrow \mathbb{R}_+$ is the **capacity** function

- s and t are called **source** and **sink**

Institute of Theoretical Informatics
Algorithmics Group

Flows

Given a graph $G = (V, E, u)$ and two nodes $s, t \in V$

- $u : E \rightarrow \mathbb{R}_+$ is the **capacity** function
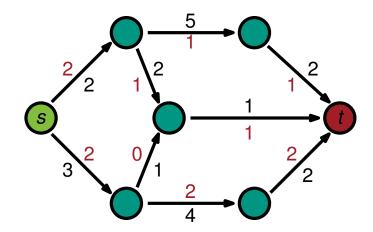
- s and t are called **source** and **sink**

A valid **flow** is a function $f : E \rightarrow \mathbb{R}_+$ with the constraints:

- $\forall (v, w) \in E : f(v, w) \leq u(v, w)$

- $\forall v \in V \setminus \{s, t\} : \sum_{(w,v) \in E} f(w, v) = \sum_{(v,w) \in E} f(v, w)$
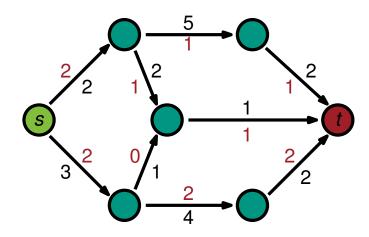
The value of the flow is $|f| = \sum_{(s,v) \in E} f(s, v)$

Institute of Theoretical Informatics
Algorithmics Group

Flows

The **residual capacity** $r_f : V \times V \to \mathbb{R}_+$ is defined as follows:

- $\forall (v, w) \in E : r_f(v, w) = u(v, w) - f(v, w)$

- $\forall (v, w) \in E :$ If $f(v, w) > 0$ and $u(w, v) = 0$, then $r_f(w, v) = f(v, w)$

Institute of Theoretical Informatics
Algorithmics Group

Flows

The **residual capacity** $r_f : V \times V \to \mathbb{R}_+$ is defined as follows:

- $\forall (v, w) \in E : r_f(v, w) = u(v, w) - f(v, w)$

- $\forall (v, w) \in E :$ If $f(v, w) > 0$ and $u(w, v) = 0$, then $r_f(w, v) = f(v, w)$

The **residual graph** $G_f = (V, E_f, r_f)$ contains all edges $(v, w) \in V \times V$ with $r_f(v, w) > 0$

Institute of Theoretical Informatics
Algorithmics Group

Flows

The **residual capacity** $r_f : V \times V \to \mathbb{R}_+$ is defined as follows:

- $\forall (v, w) \in E : r_f(v, w) = u(v, w) - f(v, w)$

- $\forall (v, w) \in E :$ If $f(v, w) > 0$ and $u(w, v) = 0$, then $r_f(w, v) = f(v, w)$
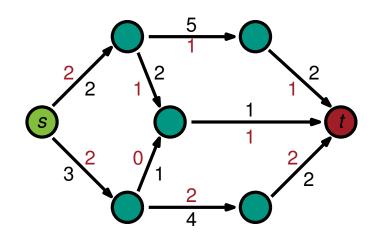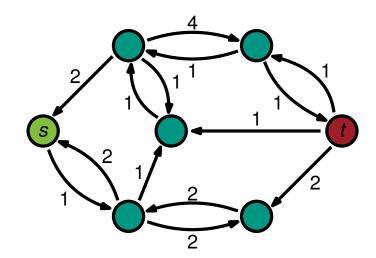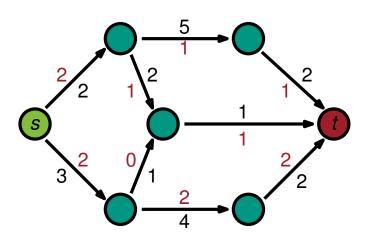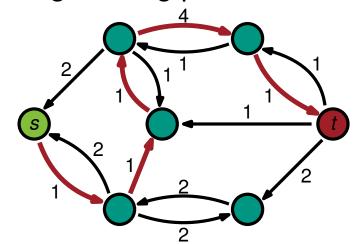
The **residual graph** $G_f = (V, E_f, r_f)$ contains all edges $(v, w) \in V \times V$ with $r_f(v, w) > 0$

- An **augmenting path** is a path in G_f from s to t

- f is a **maximum flow**, if there is no augmenting path from s to t in G_f

Institute of Theoretical Informatics
Algorithmics Group
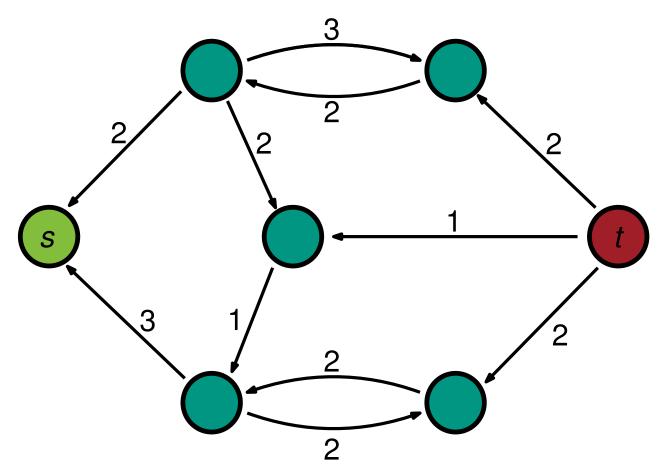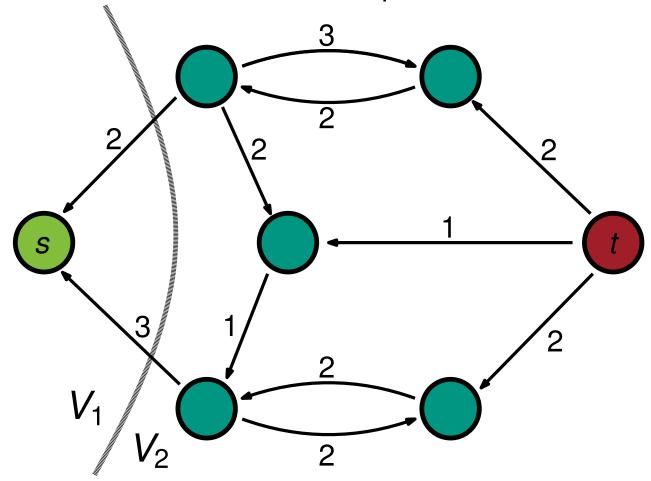
All nodes *reachable* from *s* are part of V_1 and $V_2 = V \setminus V_1$

Residual Graph G_f of a maximum flow *f*
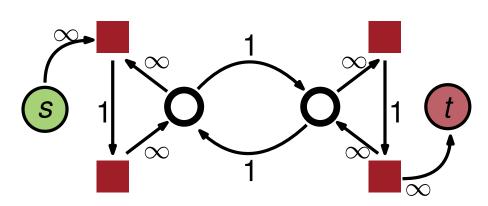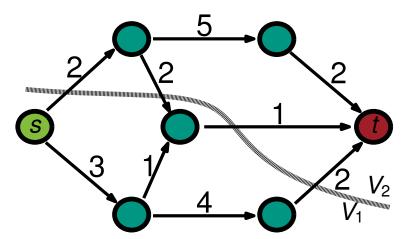
Institute of Theoretical Informatics
Algorithmics Group

All nodes *reachable* from s are part of V_1 and $V_2 = V \setminus V_1$

V_1

V_2

Residual Graph G_f of a maximum flow f

Institute of Theoretical Informatics
Algorithmics Group

Our Flow-Based Refinement Framework

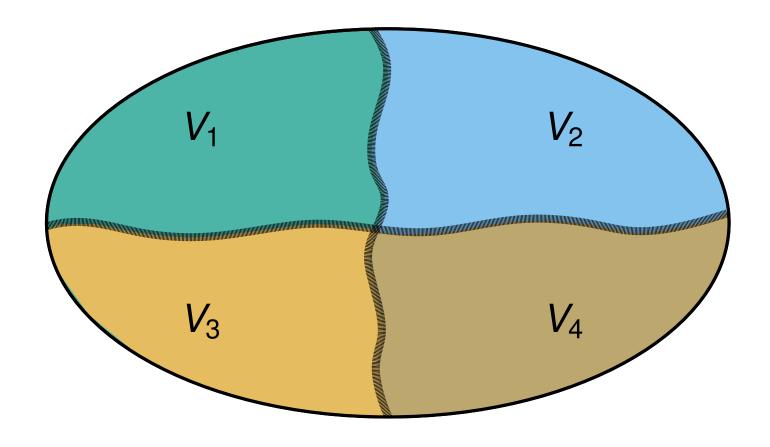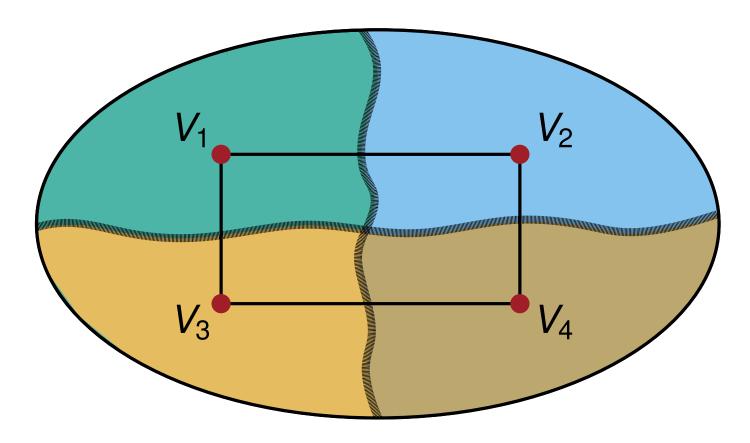Select two adjacent blocks for refinement

Build Flow Problem

Solve Flow Problem

Find feasible minimum cut
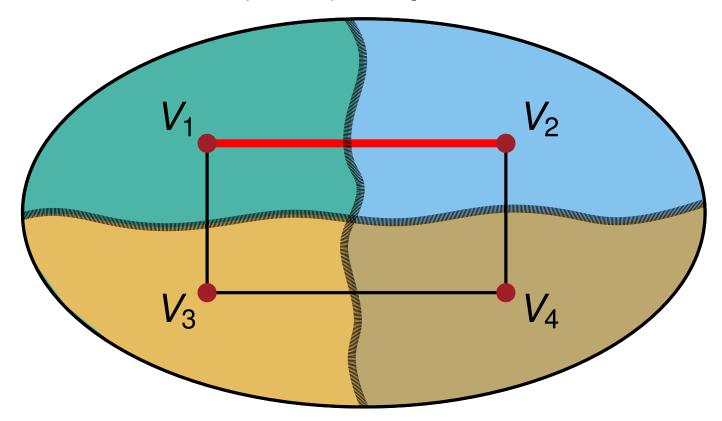
Institute of Theoretical Informatics
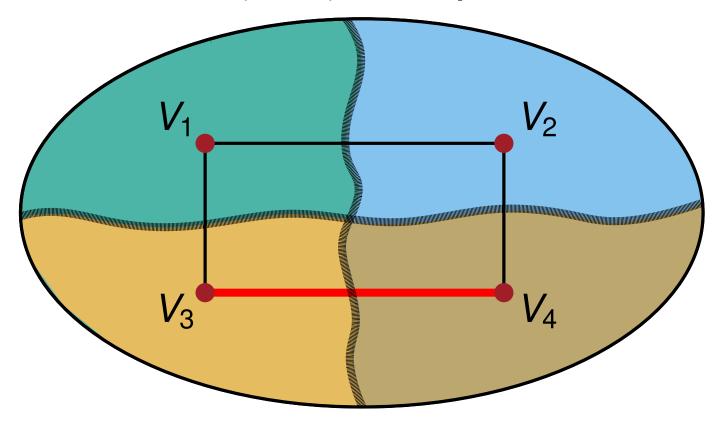Algorithmics Group

Our Flow-Based Refinement Framework

Select two adjacent blocks for refinement

Build Flow Problem

Solve Flow Problem

Find feasible minimum cut

Institute of Theoretical Informatics
Algorithmics Group

Active Block Scheduling

Institute of Theoretical Informatics
Algorithmics Group

Active Block Scheduling

Build Quotient Graph

Institute of Theoretical Informatics
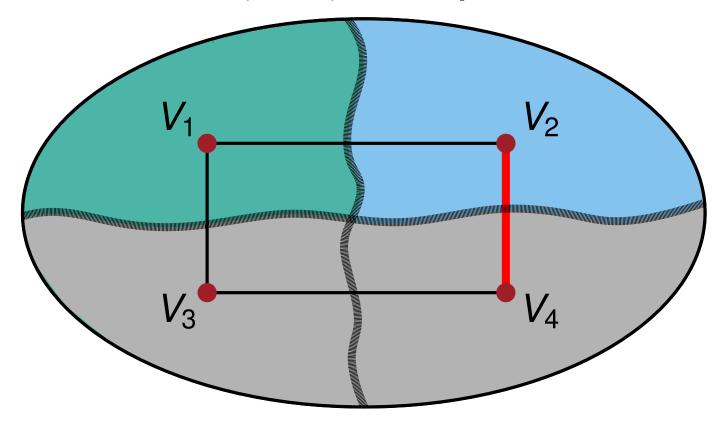Algorithmics Group

Active Block Scheduling
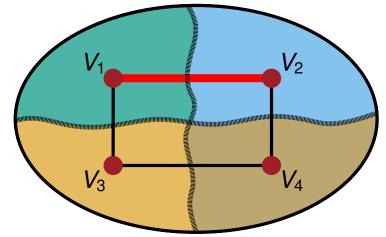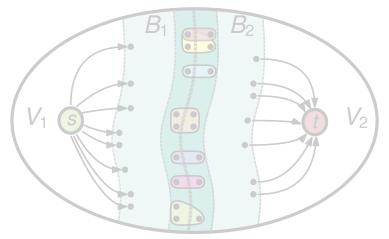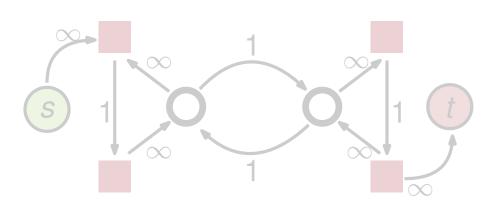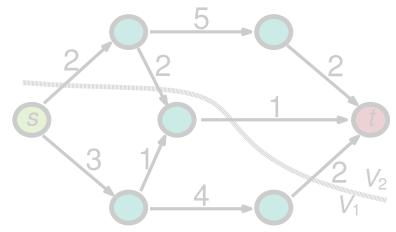
Round 1 $refine(V_1, V_2) =$ **Improvement!**

Using 2-way refinement algorithm for **active** blocks of the quotient graph

Institute of Theoretical Informatics
Algorithmics Group

Active Block Scheduling
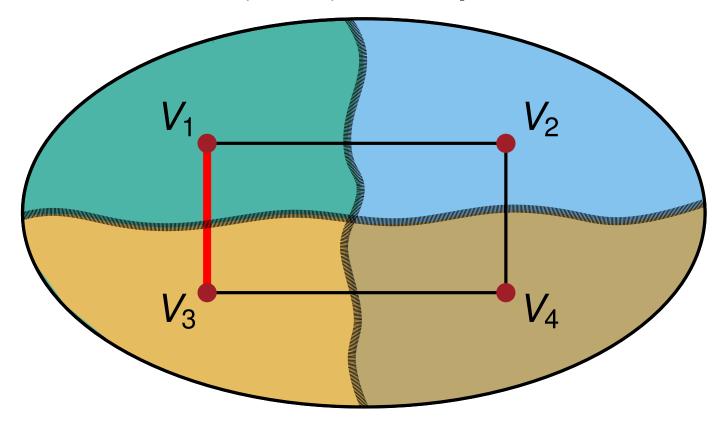
Round 1   *refine*(V_3, V_4) = **No Improvement!**

Using 2-way refinement algorithm for **active** blocks of the quotient graph

Institute of Theoretical Informatics
Algorithmics Group

Active Block Scheduling

Round 1 $refine(V_1, V_3) =$ **No Improvement!**

Using 2-way refinement algorithm for **active** blocks of the quotient graph

Institute of Theoretical Informatics
Algorithmics Group

Active Block Scheduling

Round 1   Boundary did not change $\Rightarrow$ Mark block as **inactive**

Using 2-way refinement algorithm for **active** blocks of the quotient graph
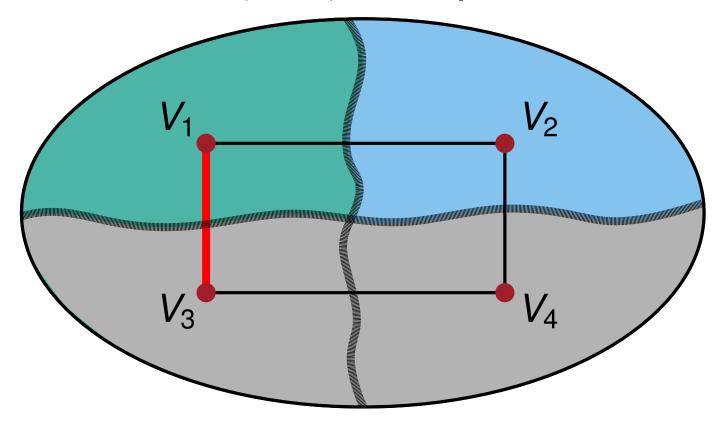
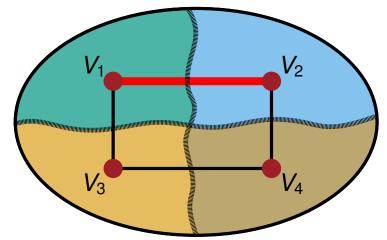**Round 2**   *refine*$(V_1, V_2) =$ **No Improvement!**

Using 2-way refinement algorithm for **active** blocks of the quotient graph

Institute of Theoretical Informatics
Algorithmics Group

Active Block Scheduling

Round 2   *refine*(V_1, V_3) = **No Improvement!**

Using 2-way refinement algorithm for **active** blocks of the quotient graph

Institute of Theoretical Informatics
Algorithmics Group

Active Block Scheduling

Round 2   *refine*(V_2, V_4) = **No Improvement!**
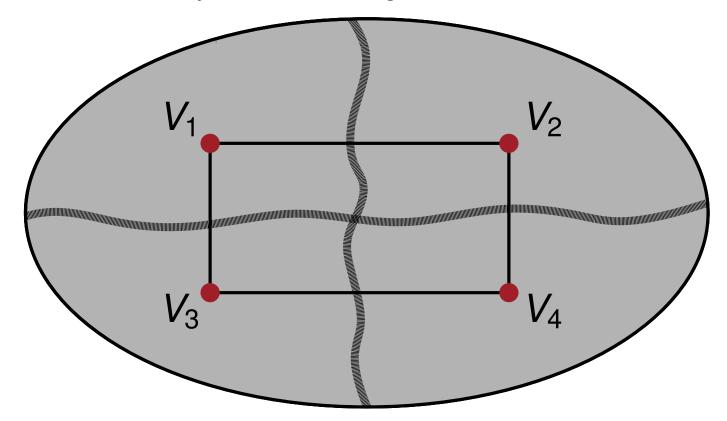
Using 2-way refinement algorithm for **active** blocks of the quotient graph

Institute of Theoretical Informatics
Algorithmics Group

Active Block Scheduling

Round 2   Boundary did not change $\Rightarrow$ Mark block as **inactive**
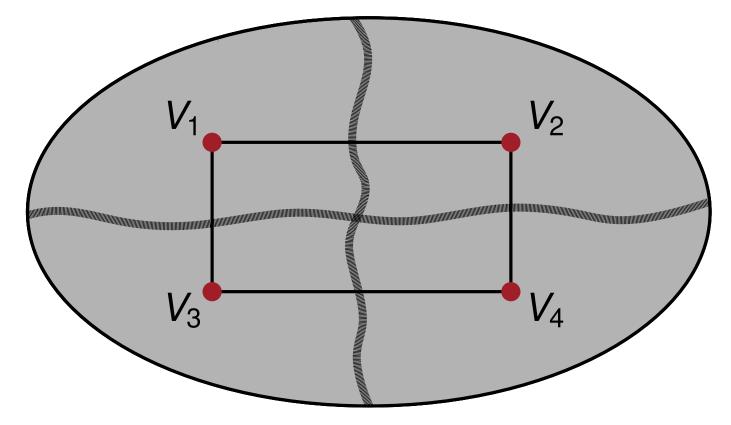
Using 2-way refinement algorithm for **active** blocks of the quotient graph

Institute of Theoretical Informatics
Algorithmics Group

Active Block Scheduling

Round 2   All blocks are **inactive** $\Rightarrow$ Algorithm terminates
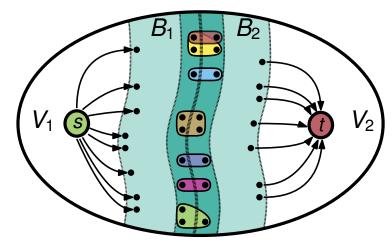
Using 2-way refinement algorithm for **active** blocks of the quotient graph

Institute of Theoretical Informatics
Algorithmics Group

Our Flow-Based Refinement Framework

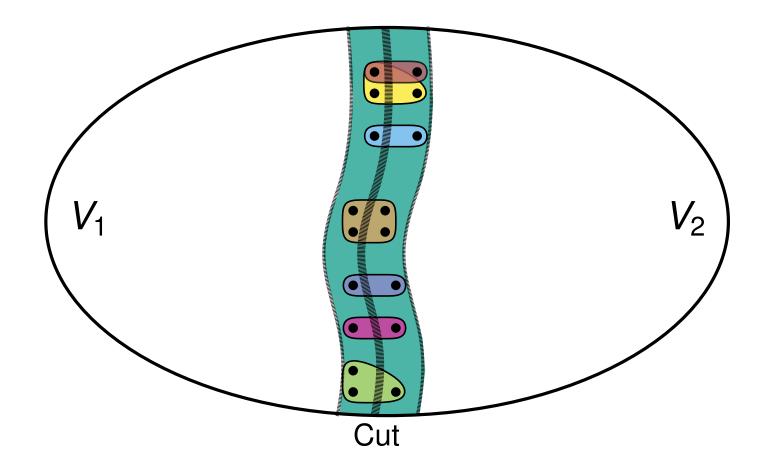Select two adjacent blocks for refinement
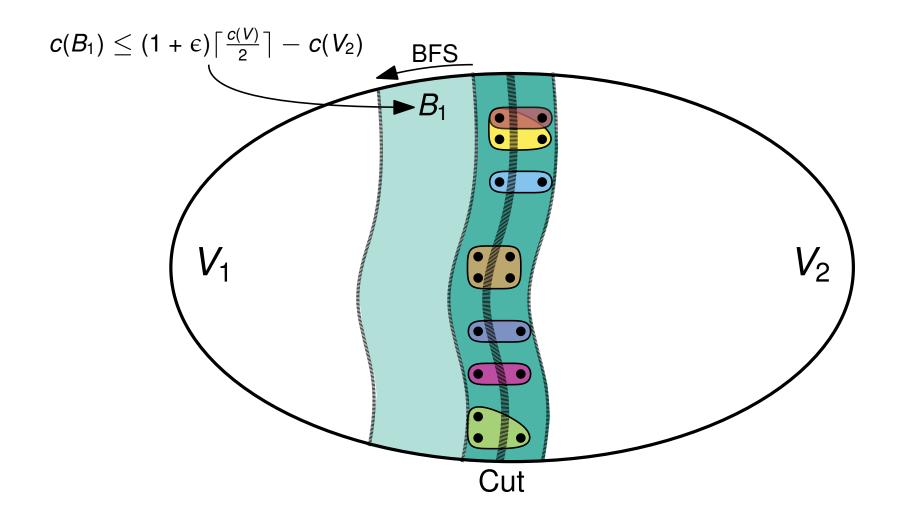

Build Flow Problem
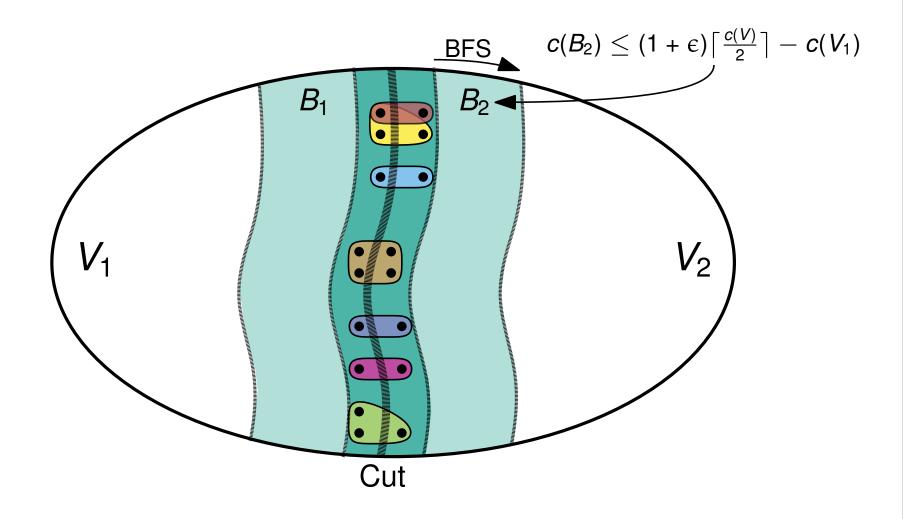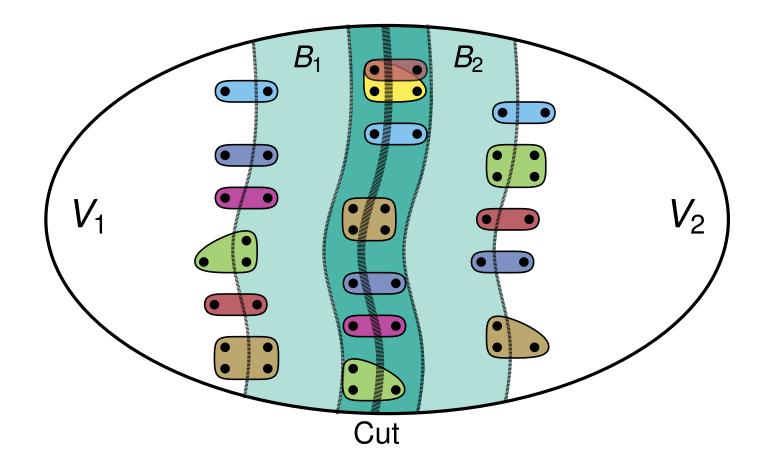

Solve Flow Problem

Find feasible minimum cut

Institute of Theoretical Informatics
Algorithmics Group

Our Flow-Based Refinement Framework
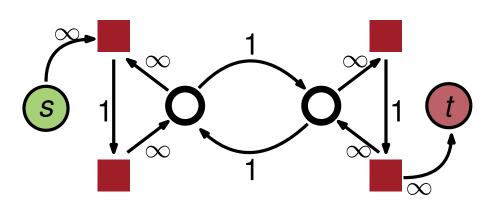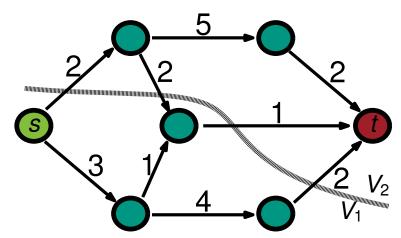

Select two adjacent blocks for refinement

Build Flow Problem

Solve Flow Problem

Find feasible minimum cut

Institute of Theoretical Informatics
Algorithmics Group

Adaptive Flow Iterations

V_1

V_2

Cut

$$c(B_1) \leq (1 + \epsilon)\left\lceil \frac{c(V)}{2} \right\rceil - c(V_2)$$

BFS

B_1

V_1

V_2

Cut

Institute of Theoretical Informatics
Algorithmics Group

Adaptive Flow Iterations

$$c(B_2) \leq (1 + \epsilon)\left\lceil \frac{c(V)}{2} \right\rceil - c(V_1)$$

BFS

B_1 B_2

V_1 V_2

Cut

Adaptive Flow Iterations

Institute of Theoretical Informatics
Algorithmics Group

Adaptive Flow Iterations

Tobias Heuer – High Quality Hypergraph Partitioning via Max-Flow-Min-Cut Computations

Institute of Theoretical Informatics
Algorithmics Group

Adaptive Flow Iterations
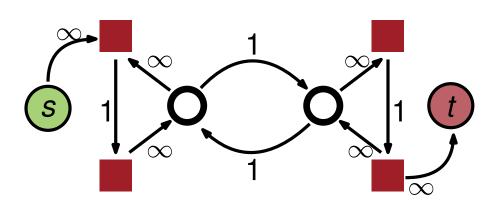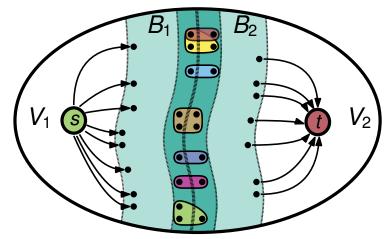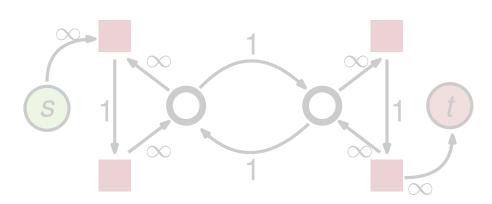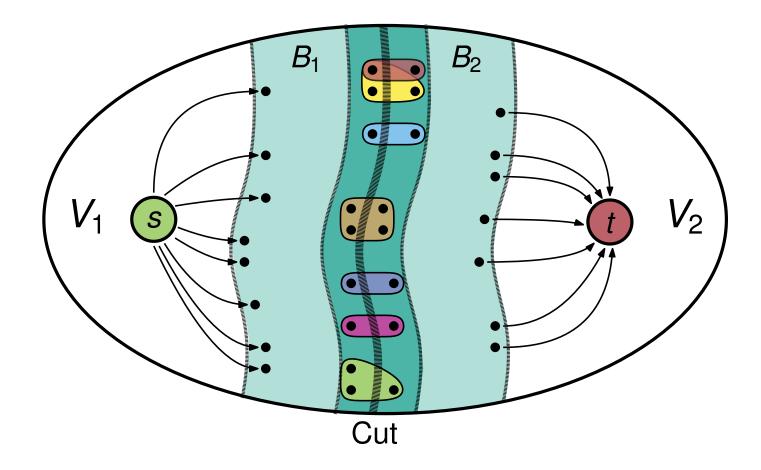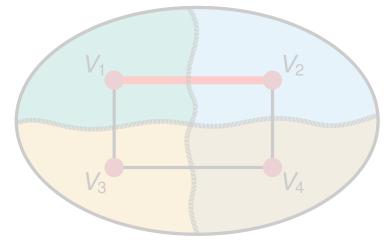
Use $\epsilon' = \alpha\epsilon$ instead of ϵ

$$c(B_1) \leq (1 + \epsilon')\left\lceil \frac{c(V)}{2} \right\rceil - c(V_2)$$

$$c(B_2) \leq (1 + \epsilon')\left\lceil \frac{c(V)}{2} \right\rceil - c(V_1)$$

Institute of Theoretical Informatics
Algorithmics Group
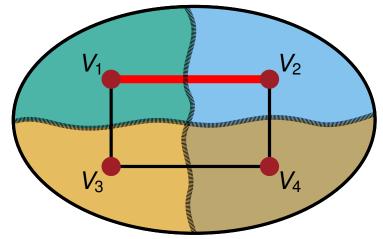
Use $\epsilon' = \alpha\epsilon$ instead of ϵ

- If improvement, $\alpha = \max(2\alpha, \alpha')$ where α' is a predefined upper bound
- Otherwise, $\alpha = \min(\frac{\alpha}{2}, 1)$

Institute of Theoretical Informatics
Algorithmics Group
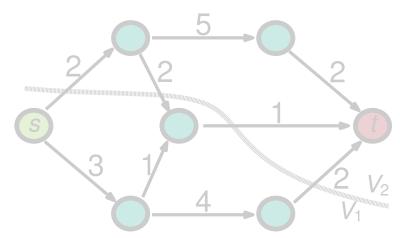
Our Flow-Based Refinement Framework
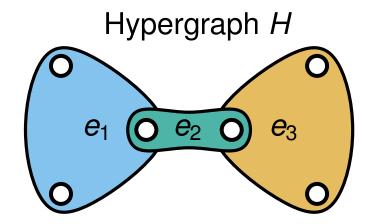

Select two adjacent blocks for refinement
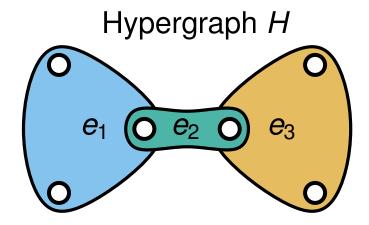

Build Flow Problem
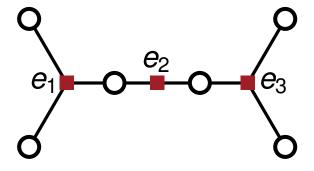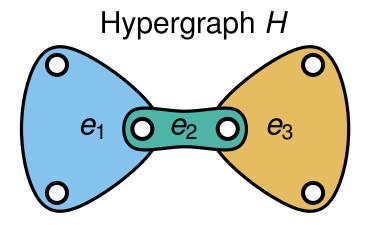

Solve Flow Problem

Find feasible minimum cut

Institute of Theoretical Informatics
Algorithmics Group

Our Flow-Based Refinement Framework

Select two adjacent blocks for refinement

Build Flow Problem

Solve Flow Problem

Find feasible minimum cut

Institute of Theoretical Informatics
Algorithmics Group

Hypergraph Flow Network

Hypergraph H

Hypergraph Flow Network

Hypergraph H

Bipartite Graph $G_*(H)$

Institute of Theoretical Informatics
Algorithmics Group

Hypergraph Flow Network

Hypergraph H

Bipartite Graph $G_*(H)$

Vertex Separator Problem

Institute of Theoretical Informatics
Algorithmics Group

Hypergraph Flow Network

Hypergraph H

Bipartite Graph $G_*(H)$

Vertex Separator Problem

Vertex Separator Transformation

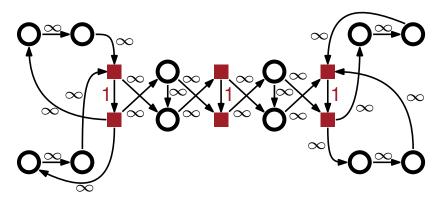Tobias Heuer – High Quality Hypergraph Partitioning via Max-Flow-Min-Cut Computations

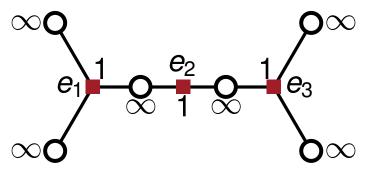Institute of Theoretical Informatics
Algorithmics Group

Hypergraph Flow Network

Hypergraph H

Bipartite Graph $G_*(H)$

Vertex Separator Problem

Vertex Separator Transformation

Lawler Network

Institute of Theoretical Informatics
Algorithmics Group

Hypergraph Flow Network

Hypergraph H

Bipartite Graph $G_*(H)$

Vertex Separator Problem

Vertex Separator Transformation

Lawler Network

Incoming hyperedge node e_2'
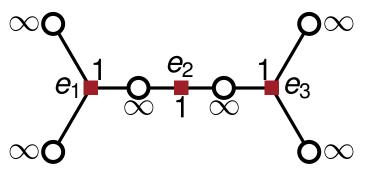
Institute of Theoretical Informatics
Algorithmics Group

Hypergraph Flow Network

Hypergraph H

Bipartite Graph $G_*(H)$

Vertex Separator Problem

Vertex Separator Transformation

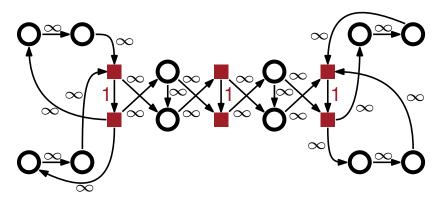## Lawler Network

Incoming hyperedge node e_2'
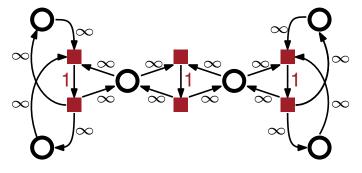

Outgoing hyperedge node e_2''

Institute of Theoretical Informatics
Algorithmics Group

Hypergraph Flow Network - Graph Edges

Hypergraph H

Lawler Network

Institute of Theoretical Informatics
Algorithmics Group

Hypergraph Flow Network - Graph Edges

Hypergraph H

e_1 e_2 e_3

Lawler Network

Graph Edge

Tobias Heuer – High Quality Hypergraph Partitioning via Max-Flow-Min-Cut Computations

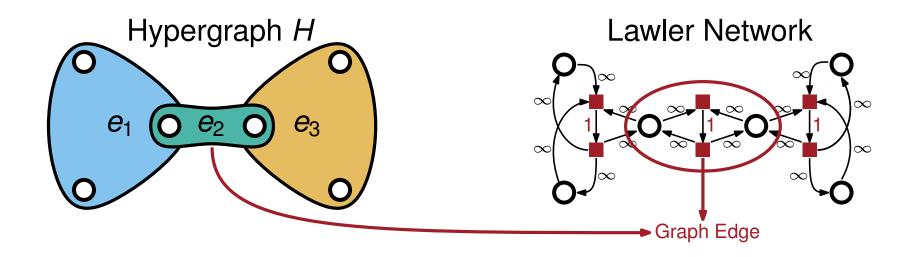Institute of Theoretical Informatics
Algorithmics Group

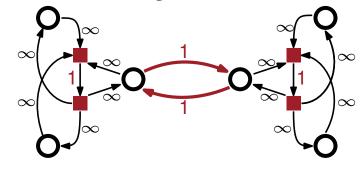# Hypergraph Flow Network - Graph Edges

Hypergraph H

Lawler Network

Graph Edge

Wong Network

Institute of Theoretical Informatics
Algorithmics Group

Hypergraph Flow Network - Low Degree Vertices

Tobias Heuer – High Quality Hypergraph Partitioning via Max-Flow-Min-Cut Computations

Institute of Theoretical Informatics
Algorithmics Group

Hypergraph Flow Network - Low Degree Vertices

Hypergraph H

Bipartite Graph $G_*(H)$

e_1 e_2 e_3

Not part of a minimum (s, t)-vertex separator

Institute of Theoretical Informatics
Algorithmics Group

Hypergraph Flow Network - Low Degree Vertices

Hypergraph H

e_1 e_2 e_3

e_1 —1— e_2 —1— e_3

1

Remove all vertices by adding a clique

Institute of Theoretical Informatics
Algorithmics Group

Hypergraph Flow Network - Low Degree Vertices

Hypergraph H

e_1 e_2 e_3

e_1 $\overset{1}{\rule{0pt}{0pt}}$ $\overset{e_2}{\underset{1}{\rule{0pt}{0pt}}}$ $\overset{1}{\rule{0pt}{0pt}}$ e_3

Remove all vertices by adding a clique

Our Network

Institute of Theoretical Informatics
Algorithmics Group

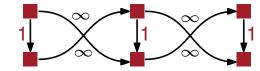# Hypergraph Flow Network - Low Degree Vertices
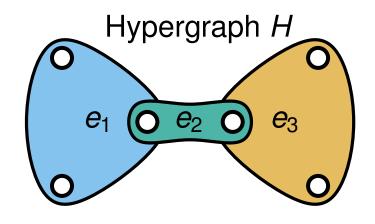
Hypergraph H

e_1 e_2 e_3
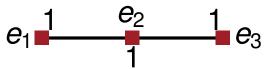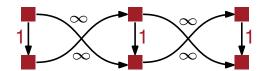

Remove all vertices by adding a clique
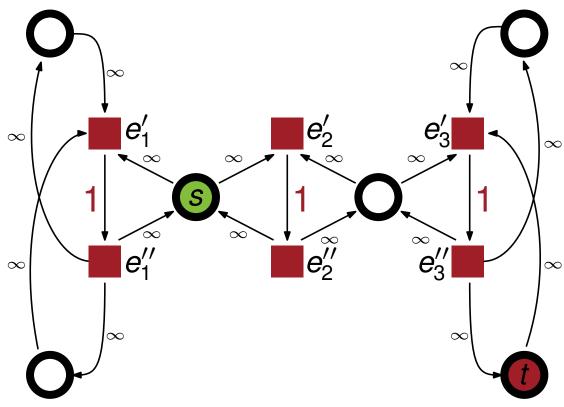
Our Network

A hypernode v induces . . .

- . . . $2d(v)$ edges in the Lawler Network
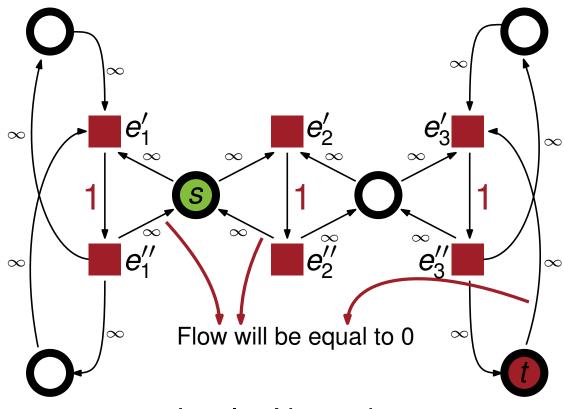
- . . . $d(v)(d(v) - 1)$ edges in our network

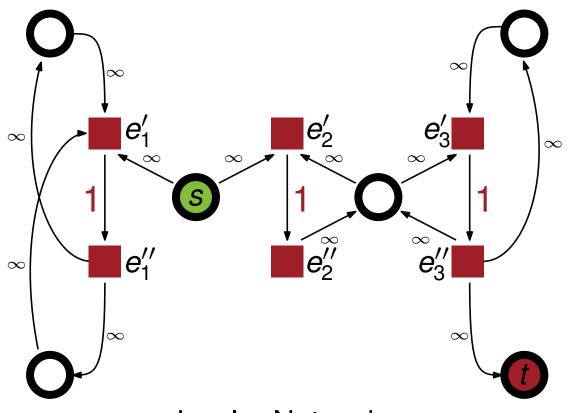If $d(v) \leq 3$, then $d(v)(d(v) - 1) \leq 2d(v)$

Institute of Theoretical Informatics
Algorithmics Group

Removing Source and Sink Vertices

Lawler Network

Institute of Theoretical Informatics
Algorithmics Group

Removing Source and Sink Vertices

Flow will be equal to 0

Lawler Network

Institute of Theoretical Informatics
Algorithmics Group

Lawler Network

Institute of Theoretical Informatics
Algorithmics Group
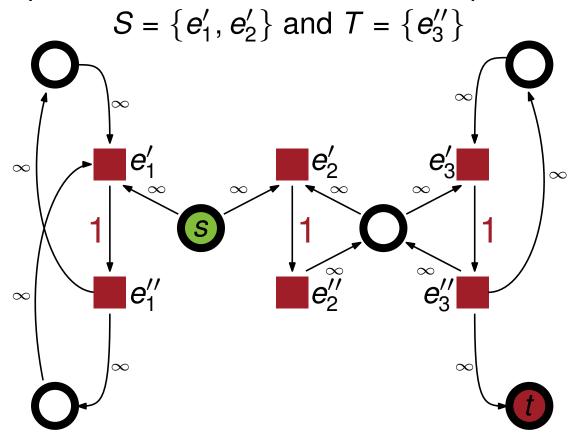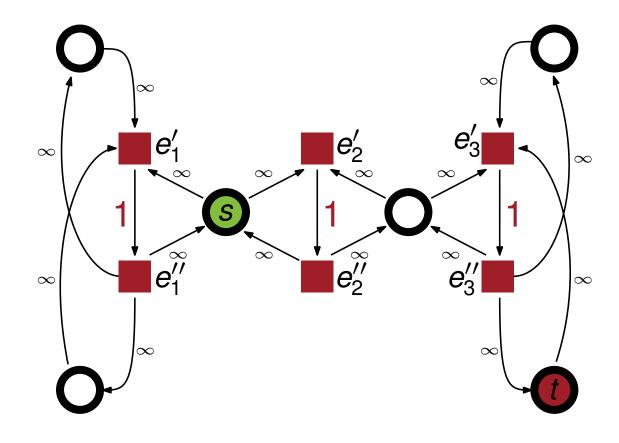
Corresponds to *Multi-Source Multi-Sink* problem with
$S = \left\{ e_1', e_2' \right\}$ and $T = \left\{ e_3'' \right\}$
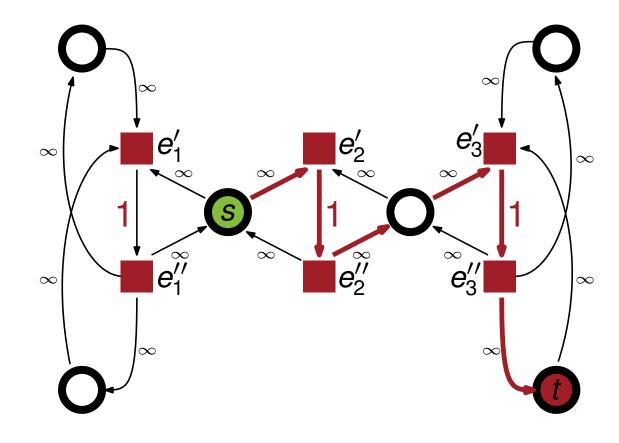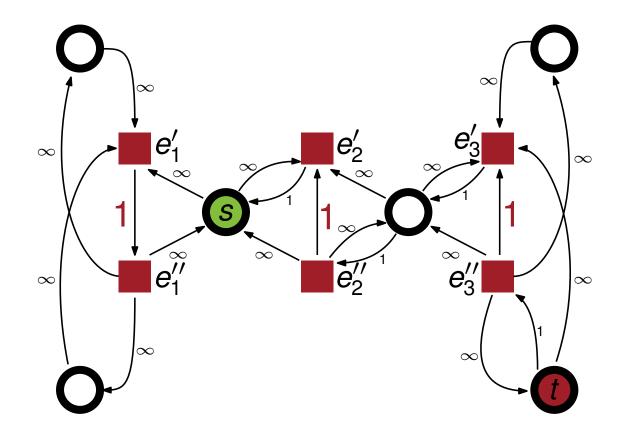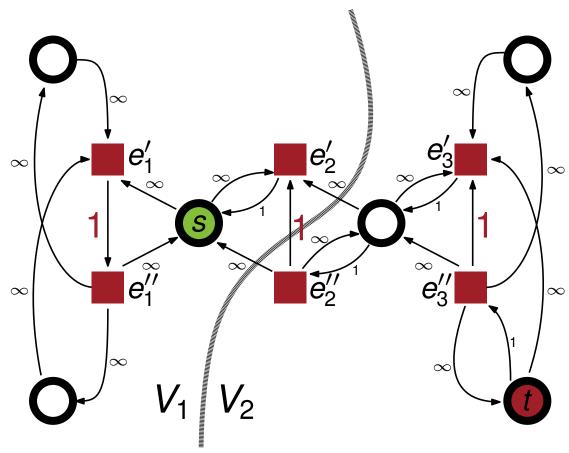
Lawler Network

Institute of Theoretical Informatics
Algorithmics Group

Institute of Theoretical Informatics
Algorithmics Group

Institute of Theoretical Informatics
Algorithmics Group

Institute of Theoretical Informatics
Algorithmics Group

All nodes *reachable* from s are part of V_1 and $V_2 = V \setminus V_1$

Institute of Theoretical Informatics
Algorithmics Group
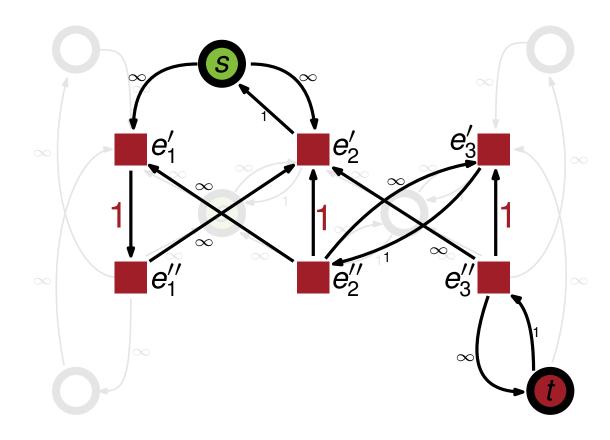
Minimum (s, t)-Bipartition
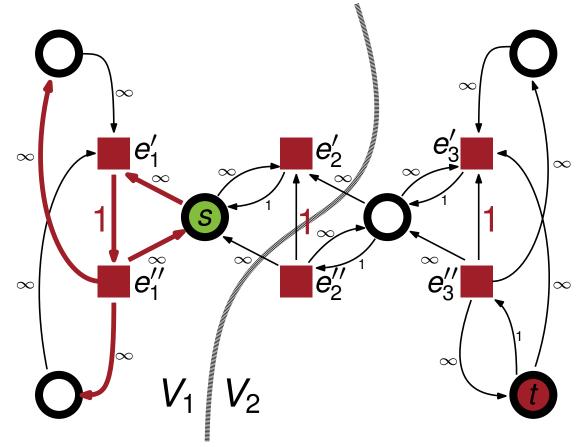
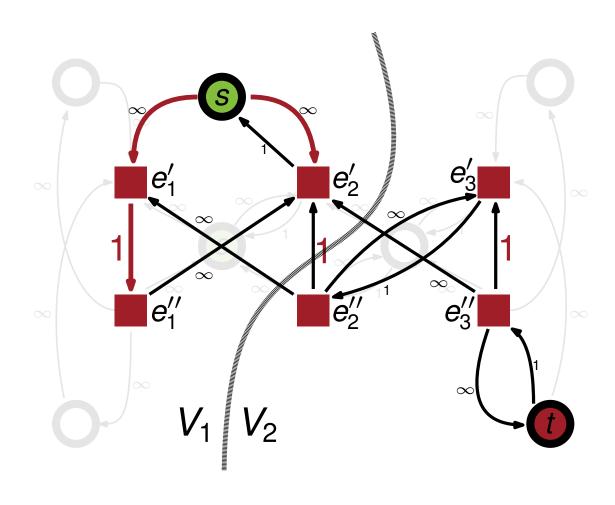For each hypernode $v \in V_1$, there exists at least one $e \in I(v)$ with $e'' \in V_1$

All nodes *reachable* from s are part of V_1 and $V_2 = V \setminus V_1$

Institute of Theoretical Informatics
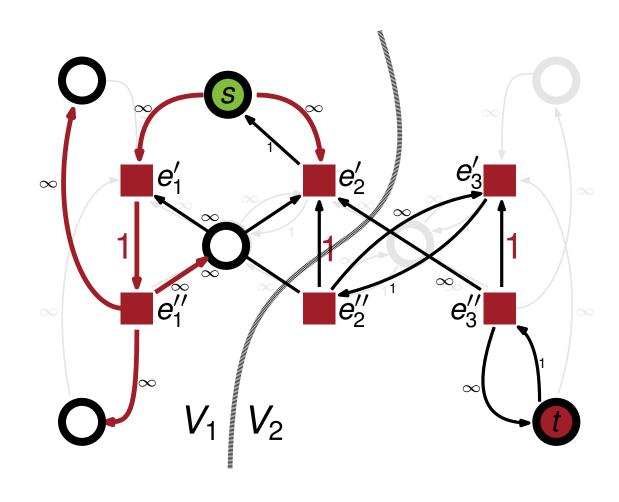Algorithmics Group

Institute of Theoretical Informatics
Algorithmics Group

Minimum (s, t)-Bipartition
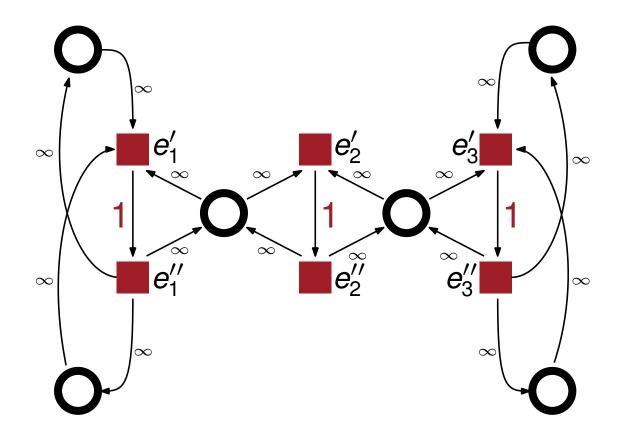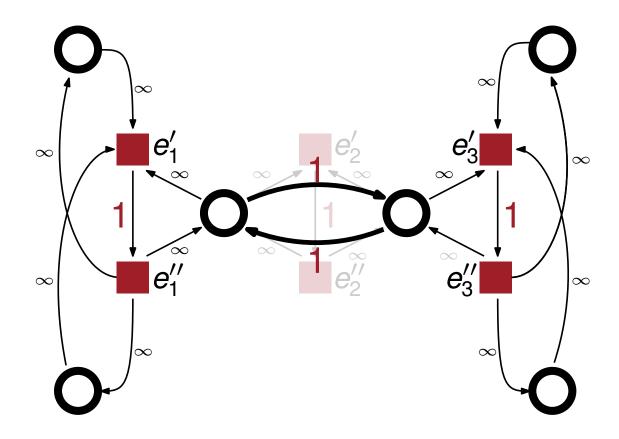
Lawler Network

Wong Network
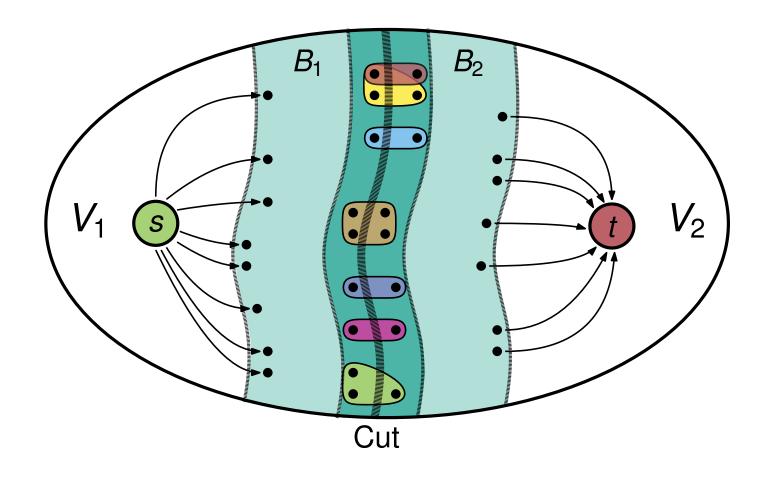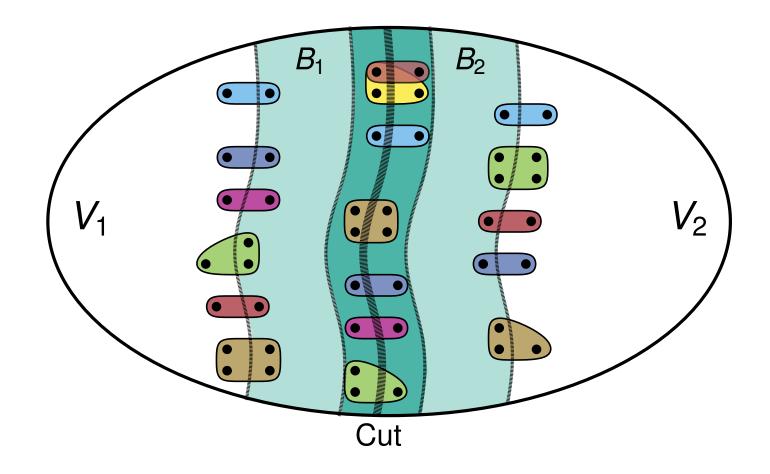
Institute of Theoretical Informatics
Algorithmics Group

Our Network

Hybrid Network

Institute of Theoretical Informatics
Algorithmics Group

Institute of Theoretical Informatics
Algorithmics Group

Institute of Theoretical Informatics
Algorithmics Group

Optimized Flow Problem Modeling Approach

Modeling Approach in *KaFFPa*

Institute of Theoretical Informatics
Algorithmics Group

Optimized Flow Problem Modeling Approach

Modeling Approach in *KaHyPar*

Institute of Theoretical Informatics
Algorithmics Group
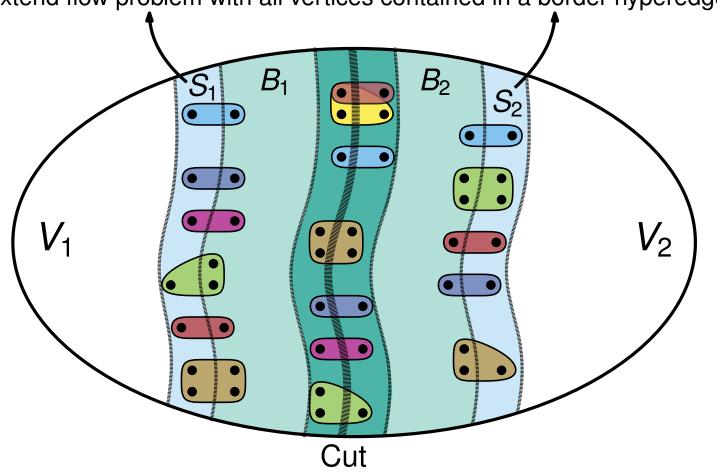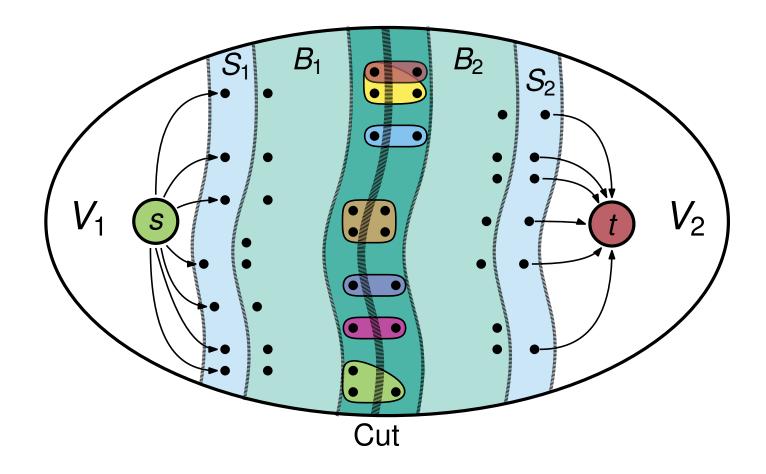
Optimized Flow Problem Modeling Approach
Modeling Approach in *KaHyPar*

Extend flow problem with all vertices contained in a border hyperedge
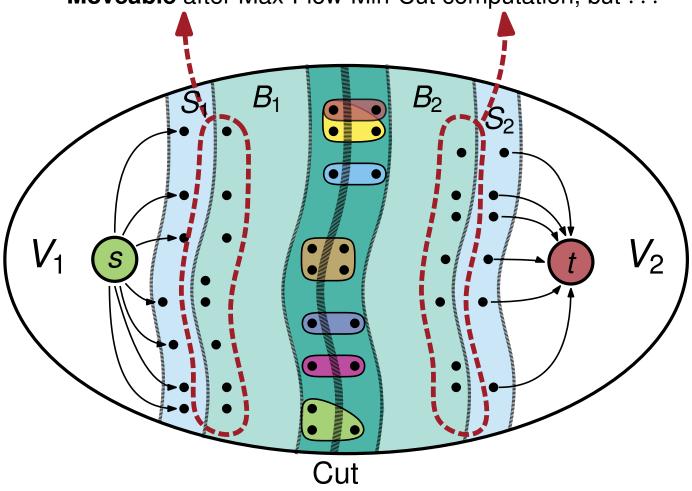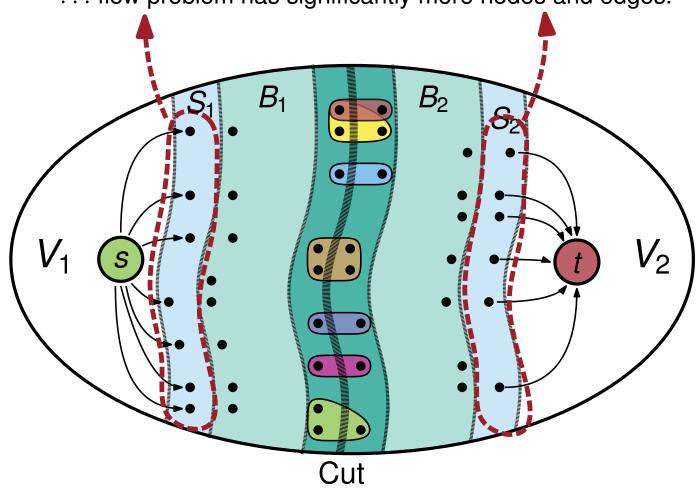
S_1 B_1 B_2 S_2

V_1 V_2

Cut

Institute of Theoretical Informatics
Algorithmics Group

Institute of Theoretical Informatics
Algorithmics Group

Optimized Flow Problem Modeling Approach

Modeling Approach in *KaHyPar*

Institute of Theoretical Informatics
Algorithmics Group
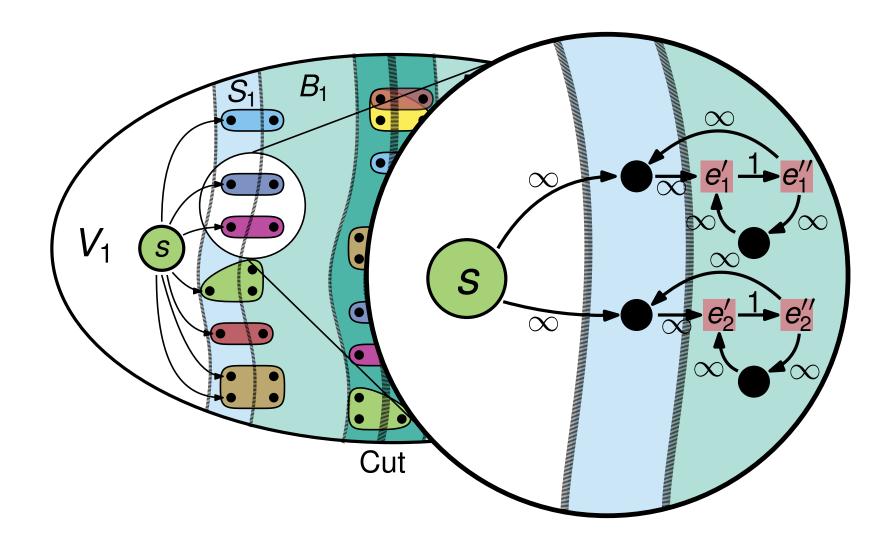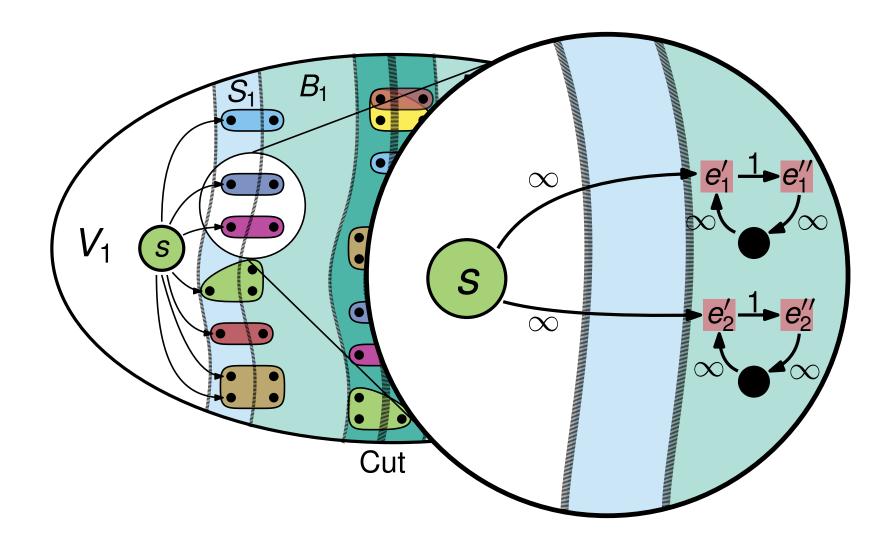
. . . flow problem has significantly more nodes and edges.
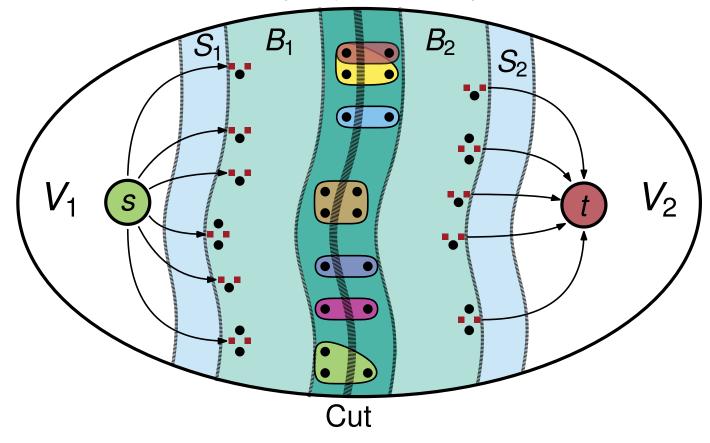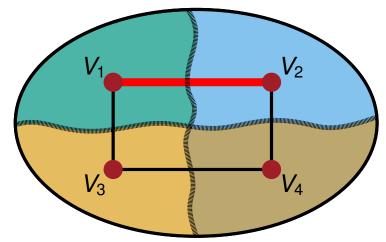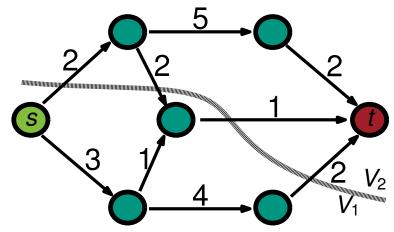
Optimized Flow Problem Modeling Approach
Modeling Approach in *KaHyPar*

Optimized Flow Problem Modeling Approach
Modeling Approach in *KaHyPar*

Institute of Theoretical Informatics
Algorithmics Group

$$S = \{\, e' \mid e \in I(S_1) \,\}$$

$$T = \{\, e'' \mid e \in I(S_2) \,\}$$

Cut

Institute of Theoretical Informatics
Algorithmics Group

Our Flow-Based Refinement Framework

Select two adjacent blocks for refinement

Build Flow Problem

Solve Flow Problem

Find feasible minimum cut

Institute of Theoretical Informatics
Algorithmics Group

Our Flow-Based Refinement Framework

Select two adjacent blocks for refinement

Build Flow Problem

Solve Flow Problem

Find feasible minimum cut

Institute of Theoretical Informatics
Algorithmics Group

Most Balanced Minimum Cut

One maximum flow *f* has enough information to enumerate all minimum (*s*, *t*)-cuts

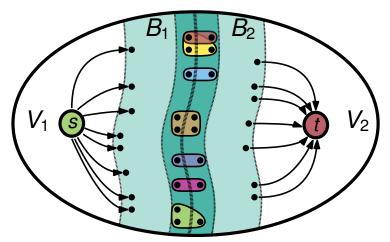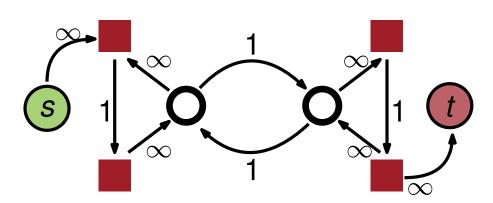Flow Graph

Residual Graph

Institute of Theoretical Informatics
Algorithmics Group

Most Balanced Minimum Cut

One maximum flow *f* has enough information to enumerate all minimum (*s*, *t*)-cuts

Flow Graph
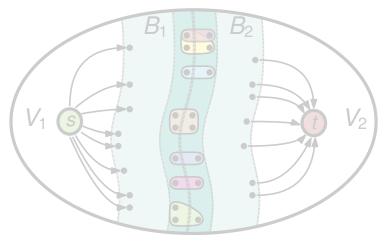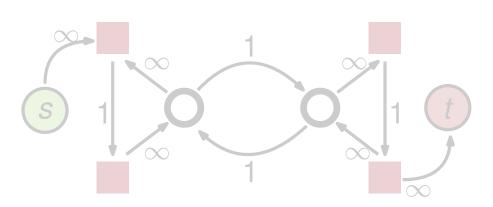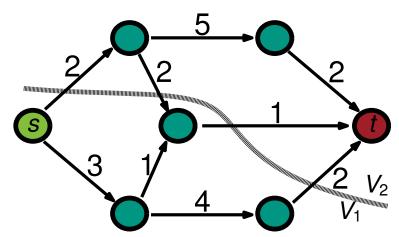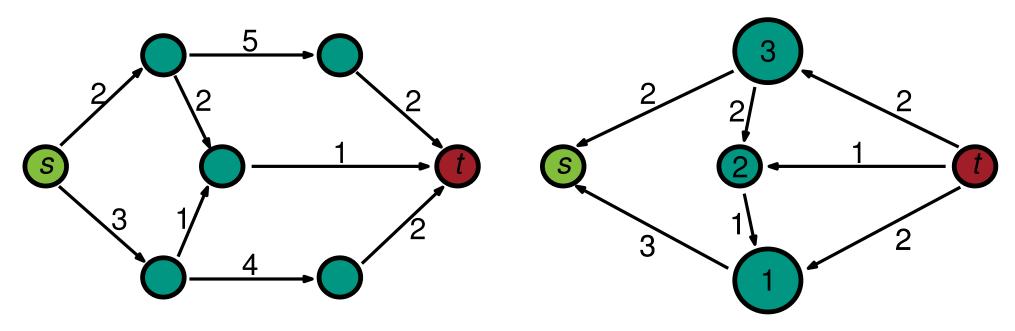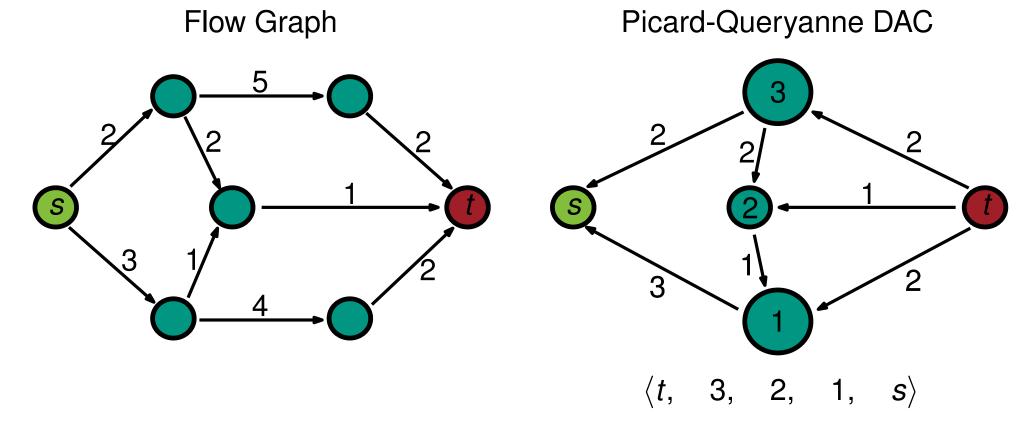
Picard-Queryanne DAC

Contract all *strongly connected components* in the residual graph

Institute of Theoretical Informatics
Algorithmics Group

Most Balanced Minimum Cut

One maximum flow *f* has enough information to enumerate all minimum (*s*, *t*)-cuts

Flow Graph

Picard-Queryanne DAC

$\langle t, \quad 3, \quad 2, \quad 1, \quad s \rangle$

Find *topological order*

Institute of Theoretical Informatics
Algorithmics Group

Most Balanced Minimum Cut

One maximum flow *f* has enough information to enumerate all minimum (*s*, *t*)-cuts

Flow Graph

Picard-Queryanne DAC

$\langle t, \quad 3, \quad 2, \quad 1, \quad s \rangle$

Sweep through **reverse** topological order

Institute of Theoretical Informatics
Algorithmics Group

Most Balanced Minimum Cut

One maximum flow f has enough information to enumerate all minimum (s, t)-cuts

Flow Graph

Picard-Queryanne DAC

$\langle t, \quad 3, \quad 2, \quad 1, \; | \; s \rangle$

Sweep through **reverse** topological order

Institute of Theoretical Informatics
Algorithmics Group

Most Balanced Minimum Cut

One maximum flow *f* has enough information to enumerate all minimum (*s*, *t*)-cuts

Flow Graph

Picard-Queryanne DAC

$\langle t, \quad 3, \quad 2, \; | \; 1, \quad s \rangle$

Sweep through **reverse** topological order

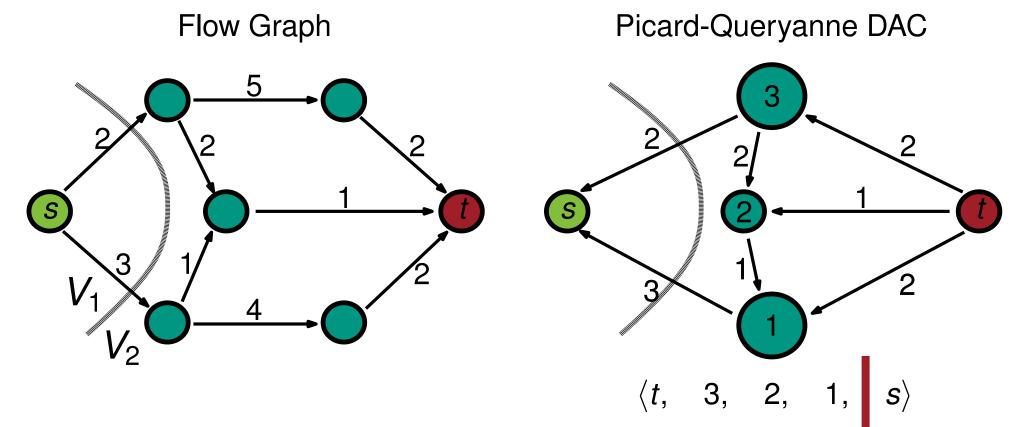Institute of Theoretical Informatics
Algorithmics Group

Most Balanced Minimum Cut

One maximum flow f has enough information to enumerate all minimum (s, t)-cuts

Flow Graph

Picard-Queryanne DAC

$\langle t, \quad 3, \quad | \quad 2, \quad 1, \quad s \rangle$

Sweep through **reverse** topological order

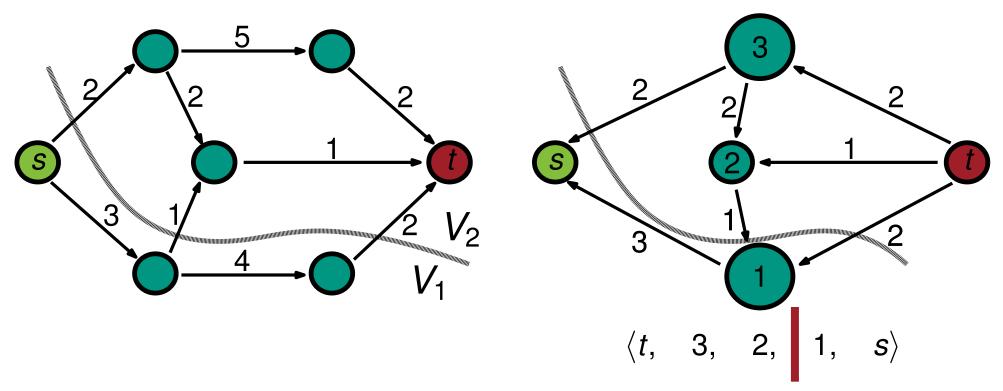Institute of Theoretical Informatics
Algorithmics Group

Most Balanced Minimum Cut

One maximum flow f has enough information to enumerate all minimum (s, t)-cuts
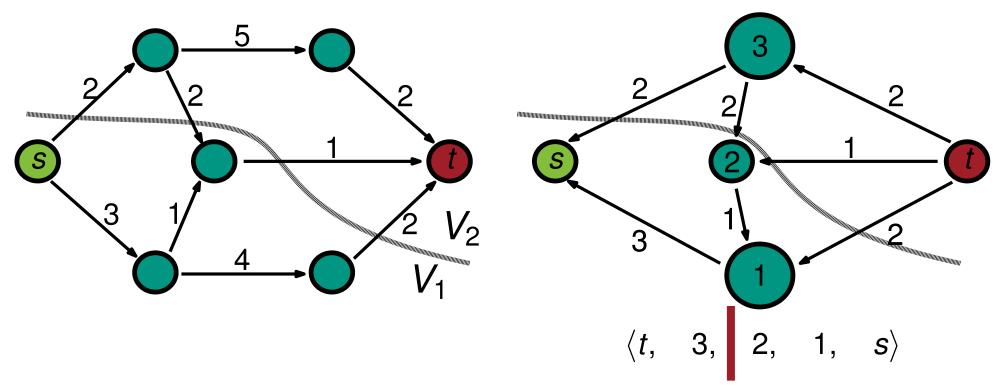
Flow Graph

Picard-Queryanne DAC

$\langle t, \;|\; 3, \; 2, \; 1, \; s \rangle$

Sweep through **reverse** topological order

Institute of Theoretical Informatics
Algorithmics Group