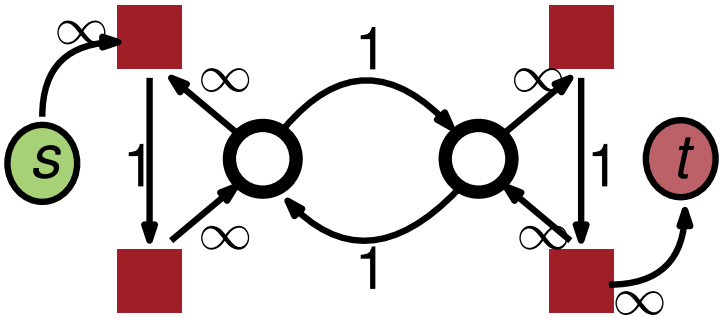
Karlsruhe Institute of Technology

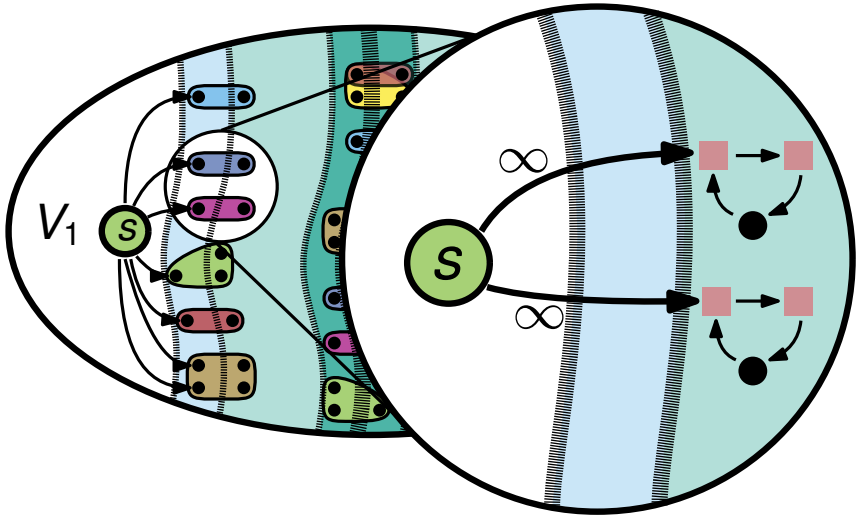
High Quality Hypergraph Partitioning via Max-Flow-Min-Cut Computations

Master Thesis · February 16, 2018

Tobias Heuer

INSTITUTE OF THEORETICAL INFORMATICS · ALGORITHMICS GROUP



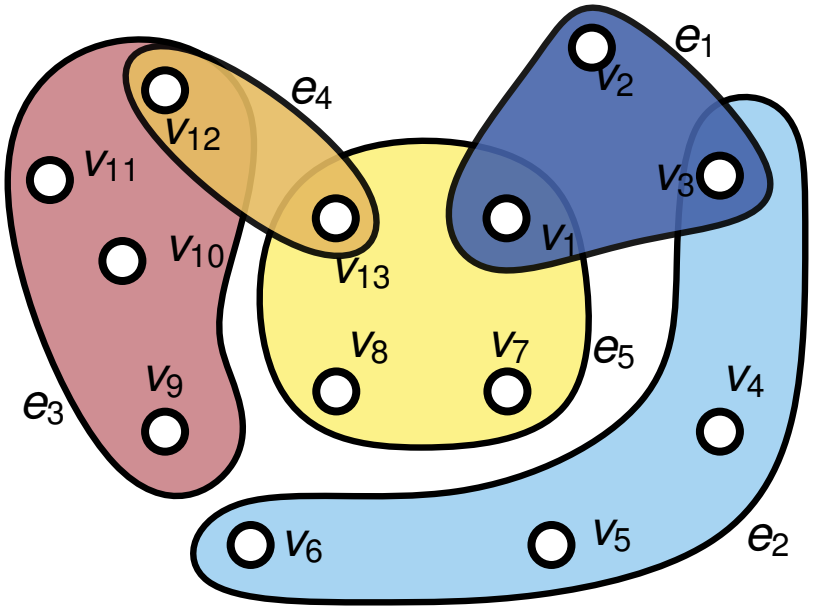


Hypergraphs
[from SEA'17]

- 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 $\omega : E \to \mathbb{R}_{\geq 1}$



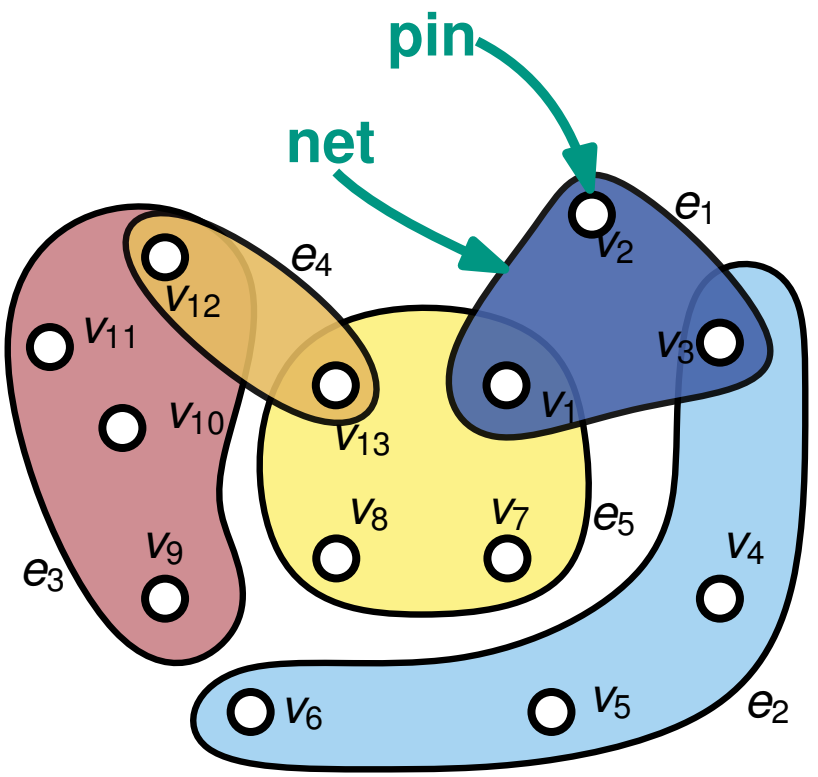
Institute of Theoretical Informatics
Algorithmics Group

Hypergraphs
[from SEA'17]

- 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 $\omega : E \to \mathbb{R}_{\geq 1}$



Institute of Theoretical Informatics
Algorithmics Group

Hypergraph Partitioning Problem

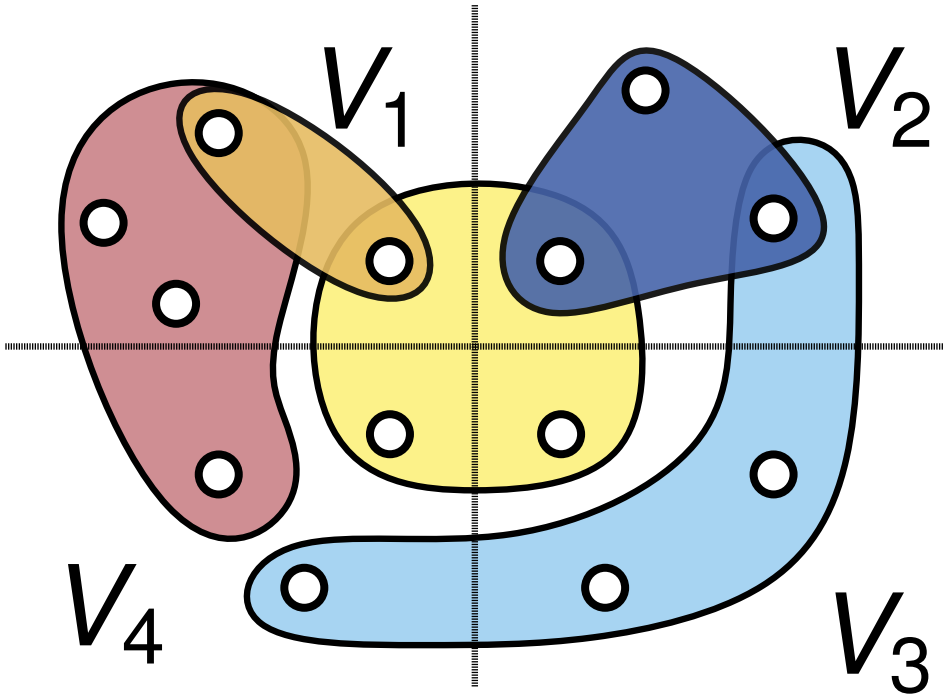
[from SEA'17]

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

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



Hypergraph Partitioning Problem

[from SEA'17]

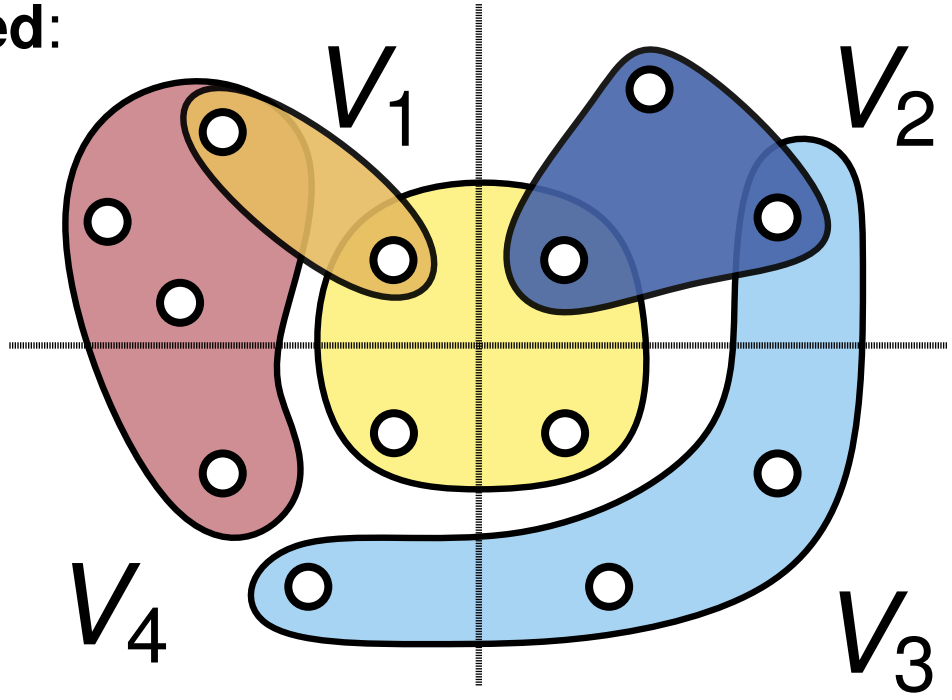
Partition hypergraph $H = (V, E, c, \omega)$ into $\mathbf{k}$ non-empty disjoint blocks $\Pi = \{V_1, \ldots, V_k\}$ such that:

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

imbalance parameter

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

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



Hypergraph Partitioning Problem

[from SEA'17]

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**:

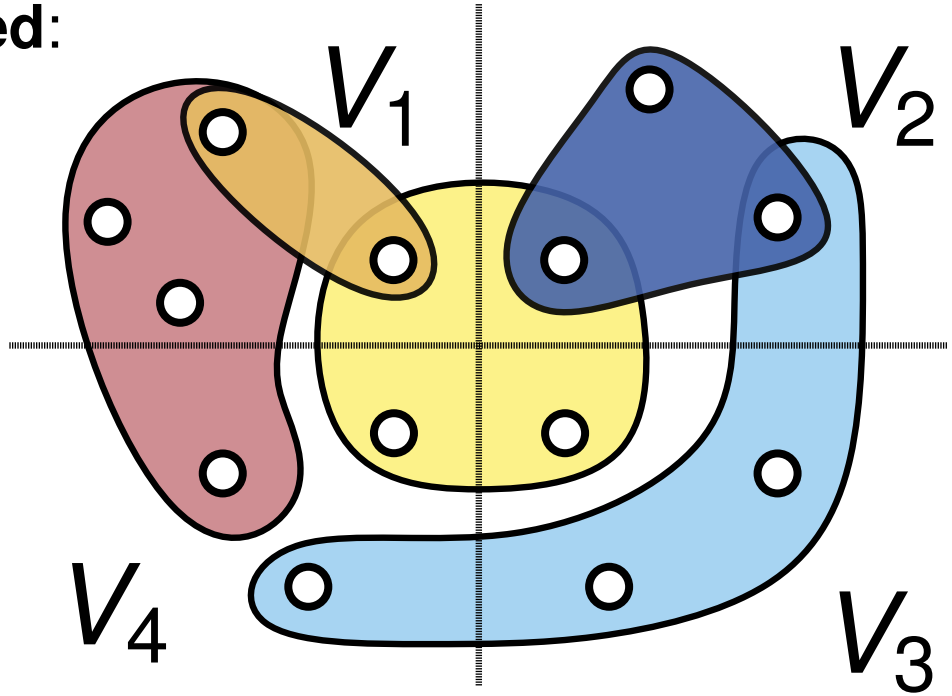
imbalance parameter

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

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

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

connectivity:
 # blocks connected by net e

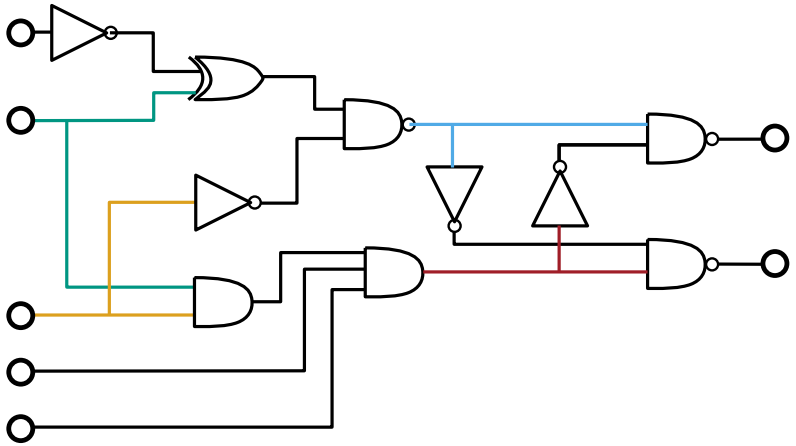


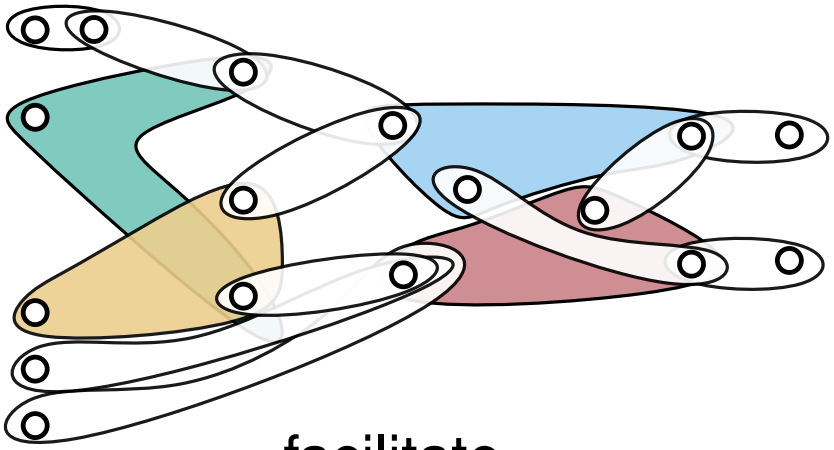
Applications

[from SEA'17]

Karlsruhe Institute of Technology

VLSI Design





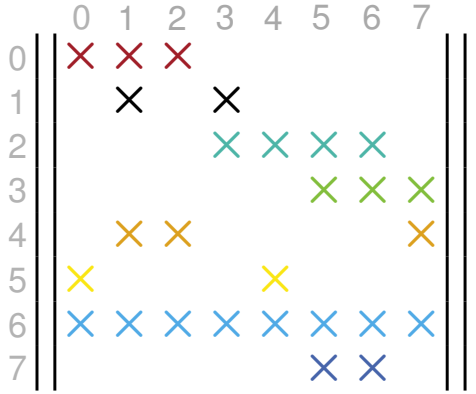
facilitate
floorplanning & placement

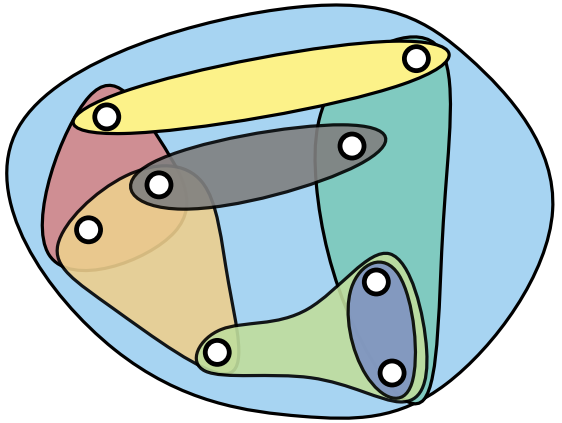
Application Domain

Hypergraph Model

Goal

Scientific Computing



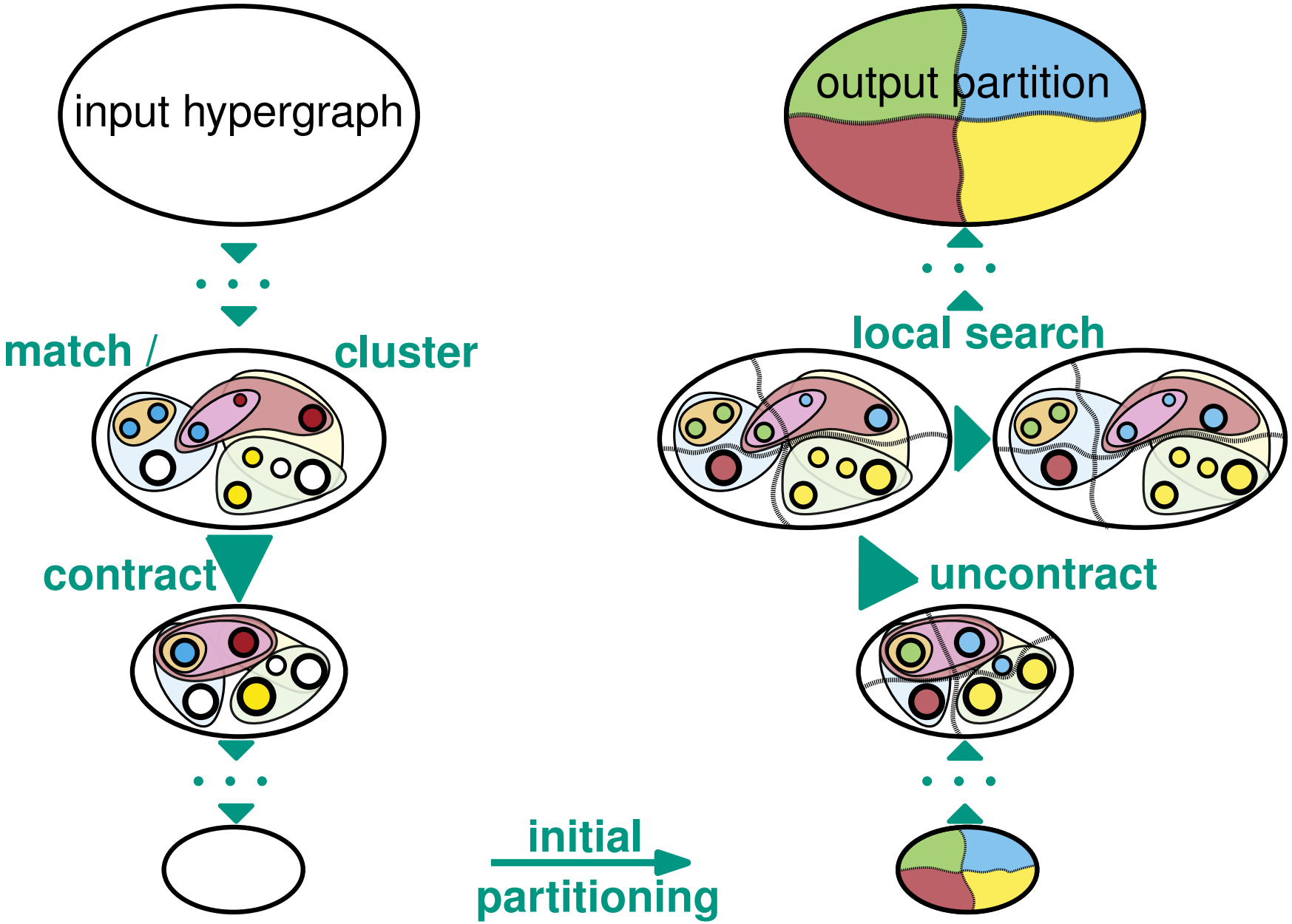


minimize
communication

Institute of Theoretical Informatics
Algorithmics Group

The Multilevel Framework

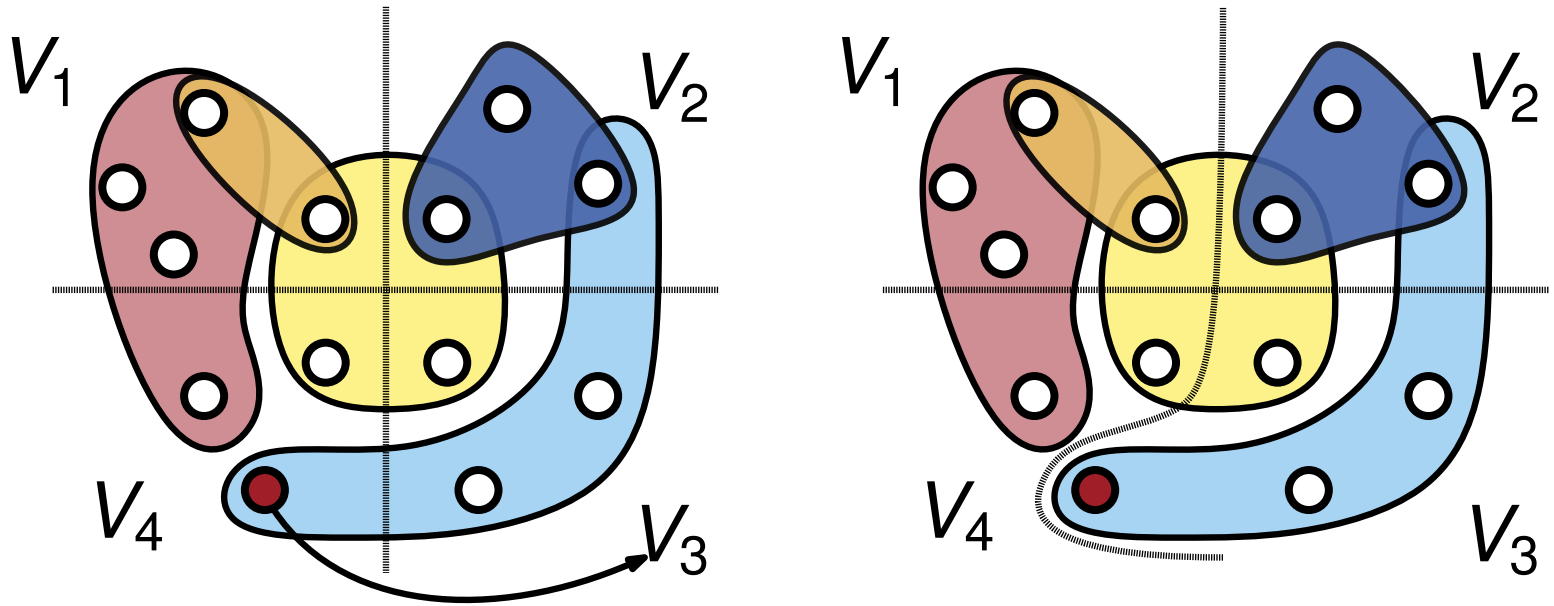
[from SEA'17]



Institute of Theoretical Informatics
Algorithmics Group

FM Algorithm

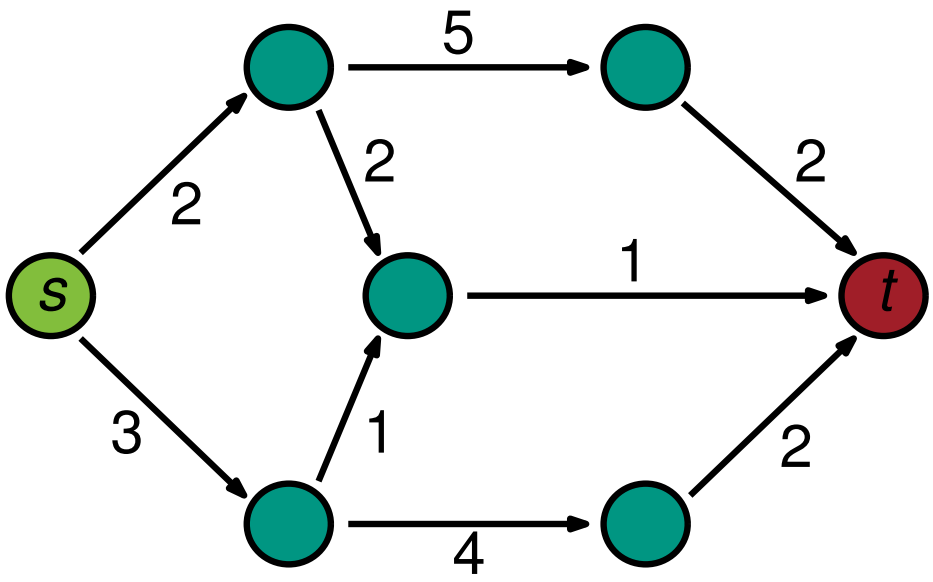
[Fiduccia, Mattheyses 88]



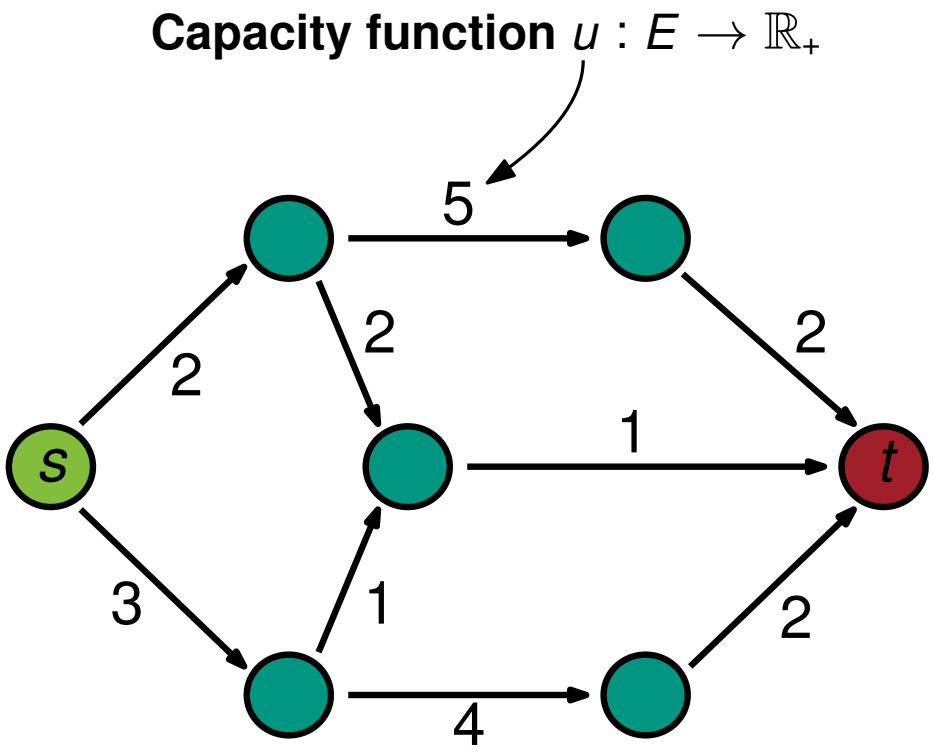
Moving ● from V_4 to V_3 reduces cut by 1

gain

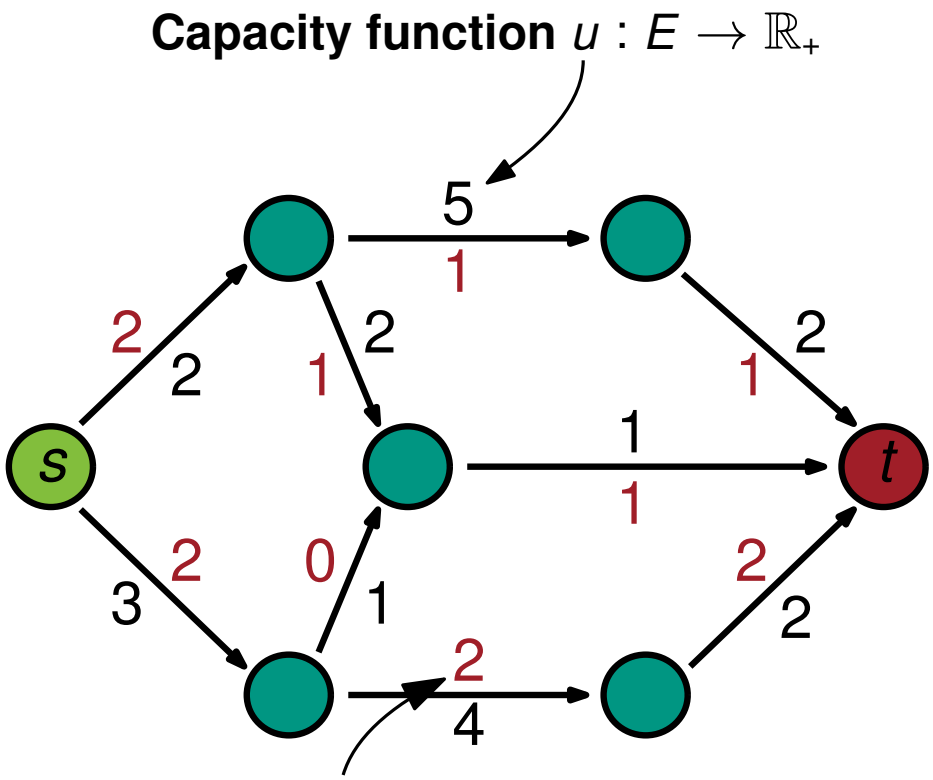
Flows



Flows

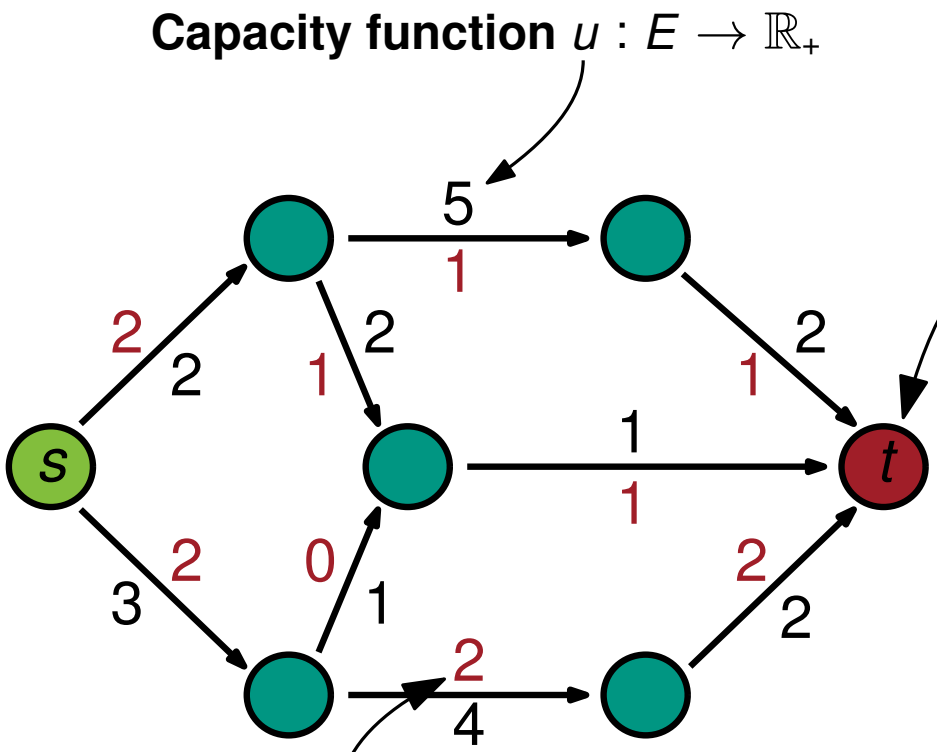


Flows



- $\forall (v, w) \in E : f(v, w) \leq u(v, w)$
- $\forall v \in V \setminus \{s, t\} : \sum_{(w,v) \in E} f(w, v) = \sum_{(v,w) \in E} f(v, w)$

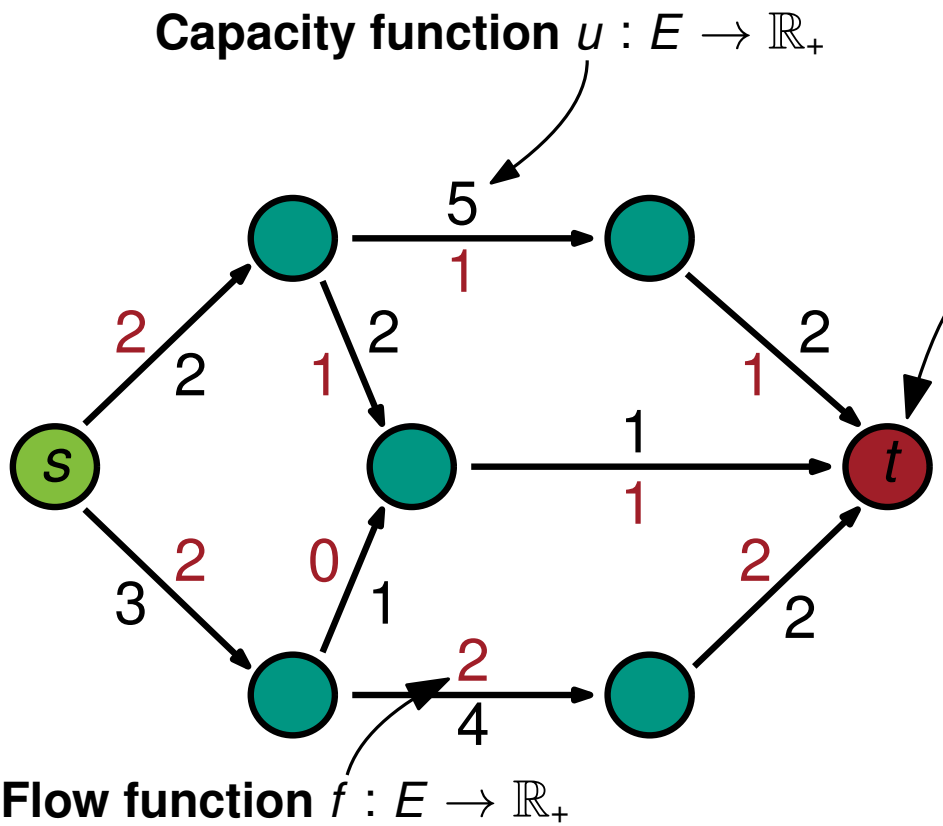
Flows



- $\forall (v,w) \in E : f(v,w) \leq u(v,w)$
- $\forall v \in V \setminus \{s,t\} : \sum_{(w,v)\in E} f(w,v) = \sum_{(v,w)\in E} f(v,w)$

Institute of Theoretical Informatics
Algorithmics Group

Flows



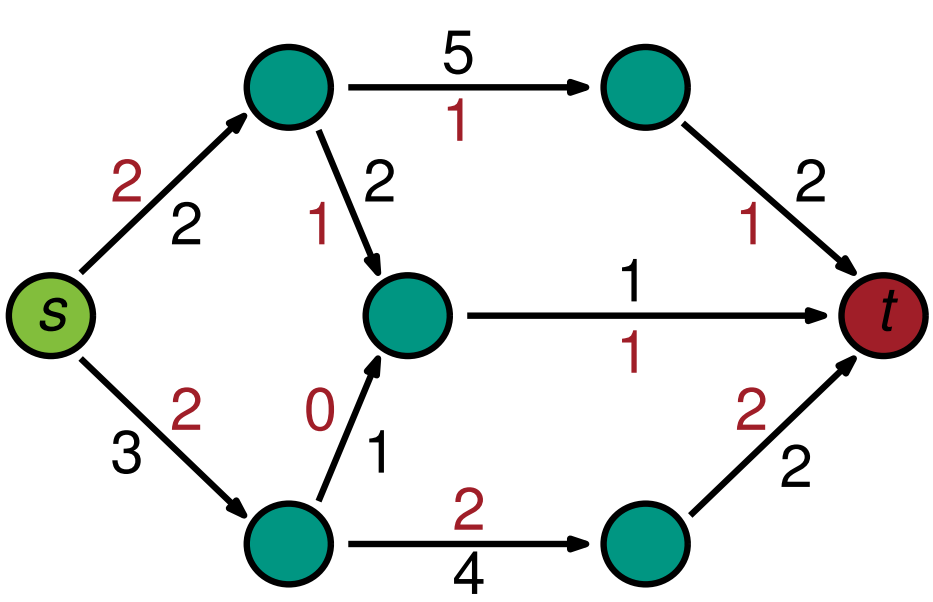
Value of flow $|f| = \sum_{(v,t)\in E} f(v,t)$

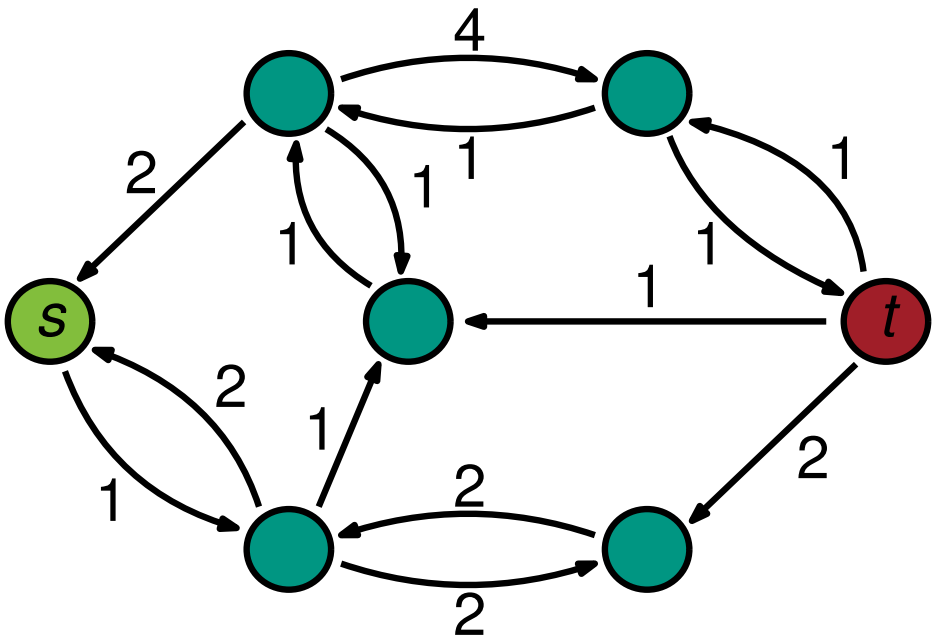
Maximum Flow Problem

Find a maximum flow f from s to t such that $\forall f' : |f'| \leq |f|$

- $\forall (v,w) \in E : f(v,w) \leq u(v,w)$
- $\forall v \in V \setminus \{s,t\} : \sum_{(w,v)\in E} f(w,v) = \sum_{(v,w)\in E} f(v,w)$

Flows

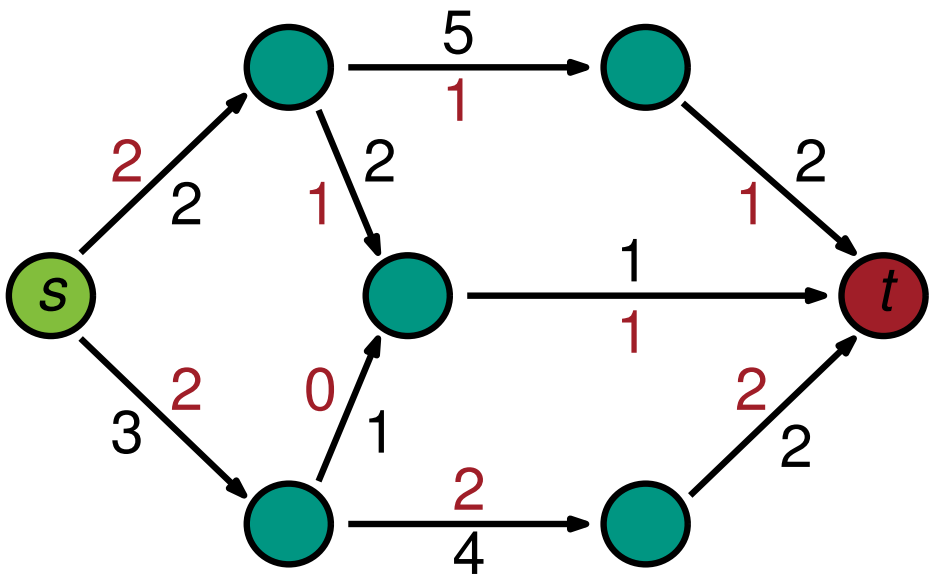




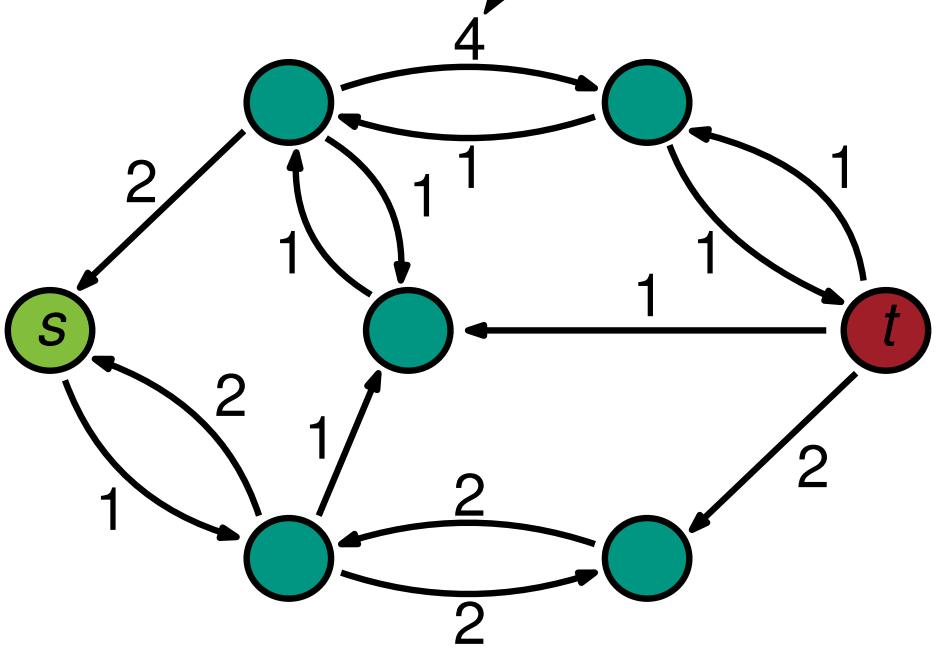
Residual Graph G_f

Institute of Theoretical Informatics
Algorithmics Group

Flows

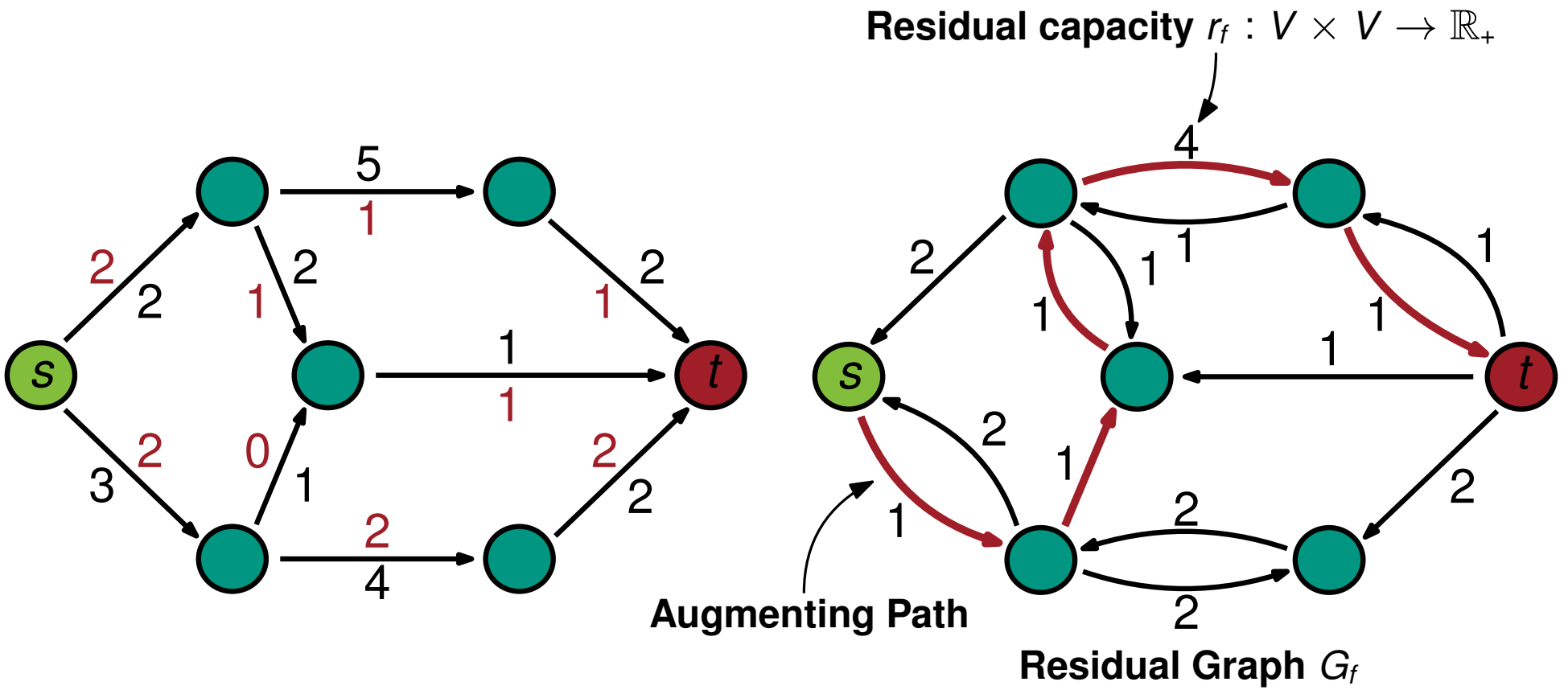


Residual capacity $r_f : V \times V \to \mathbb{R}_+$



Residual Graph G_f

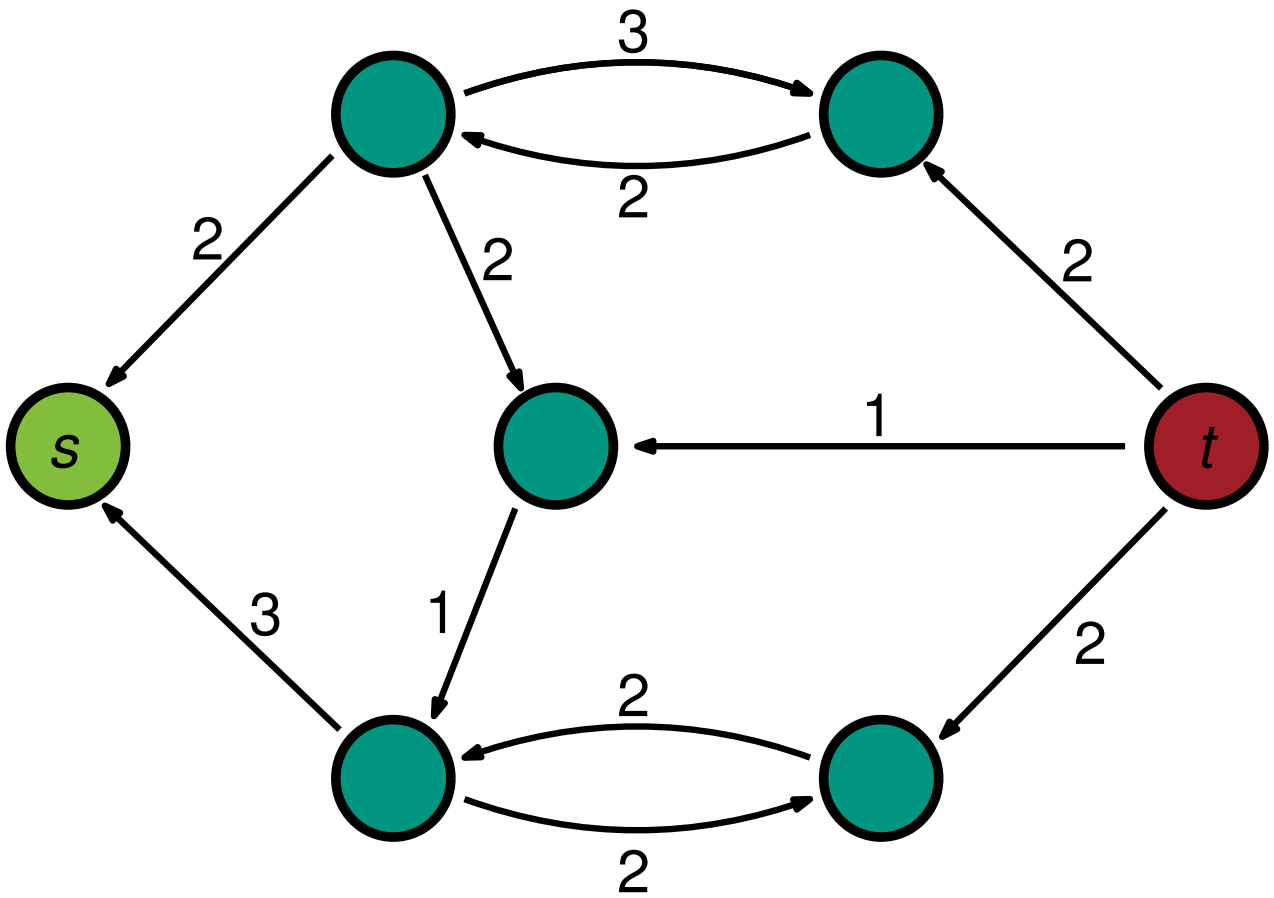
Flows



Institute of Theoretical Informatics
Algorithmics Group

Minimum (s, t)-Bipartition

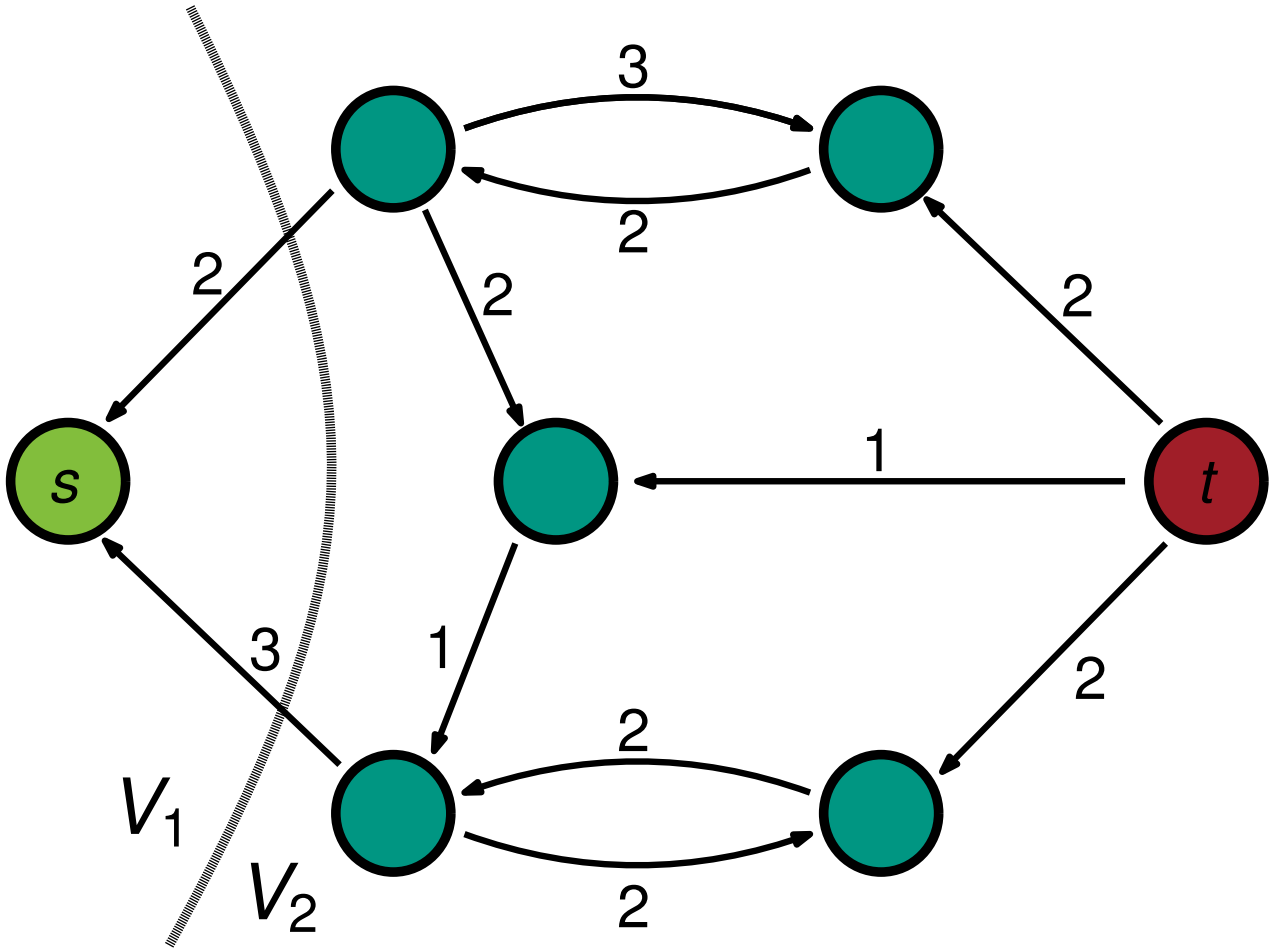
All nodes *reachable* from s are part of V_1 and $V_2 = V \setminus V_1$



Residual Graph G_f of a maximum flow f

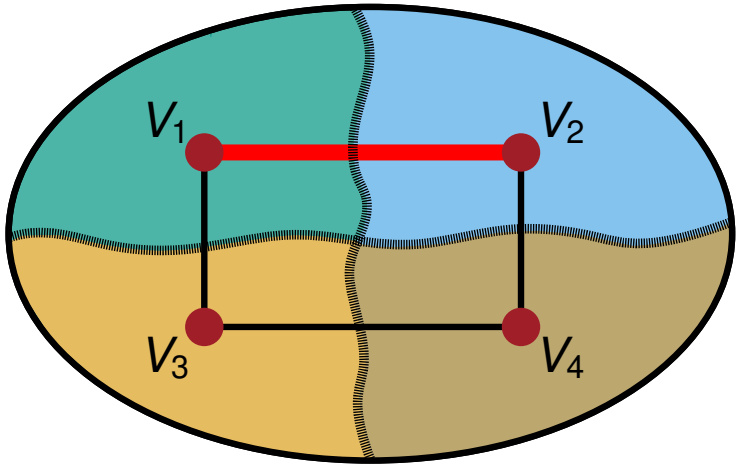
Minimum (s, t)-Bipartition

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

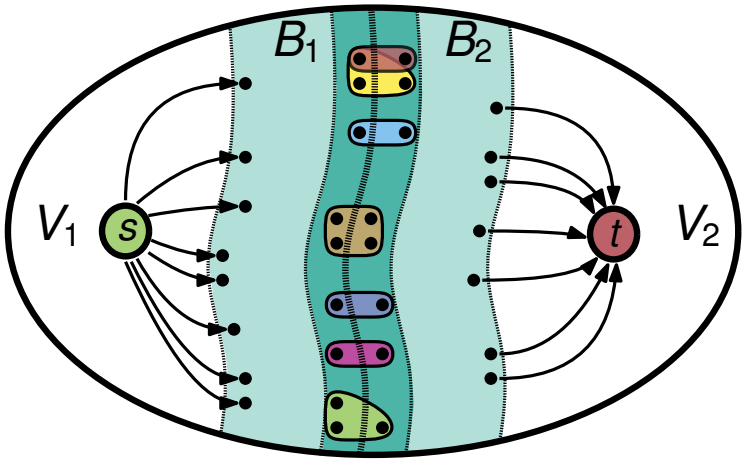


Residual Graph G_f of a maximum flow f

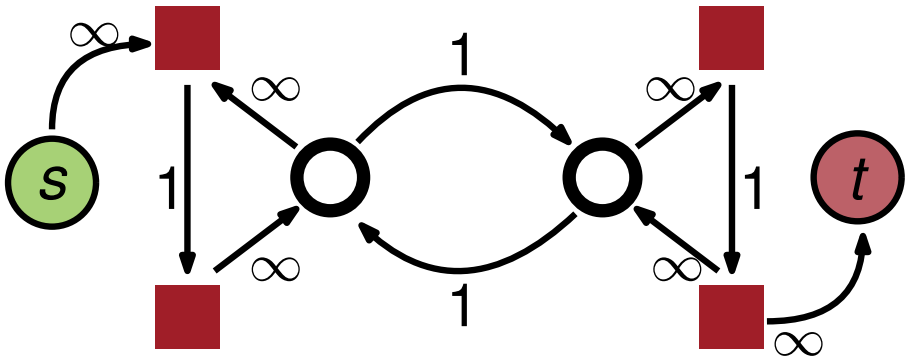
Our Flow-Based Refinement Framework



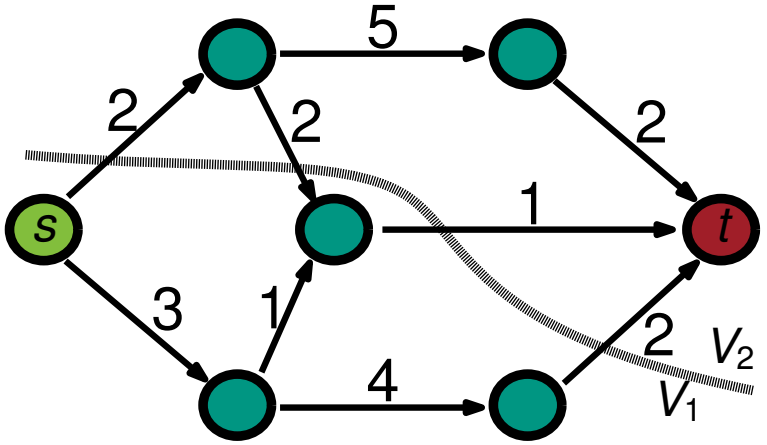
Select two adjacent blocks for refinement



Build Flow Problem



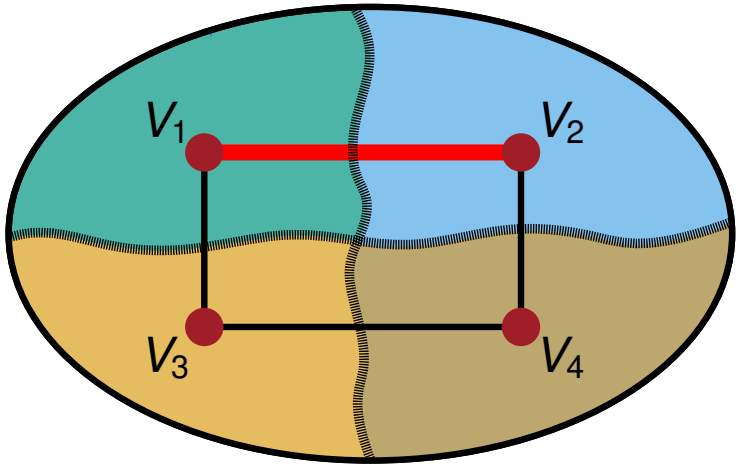
Solve Flow Problem



Find feasible minimum cut

Institute of Theoretical Informatics
Algorithmics Group

Our Flow-Based Refinement Framework



Select two adjacent blocks for refinement

Build Flow Problem

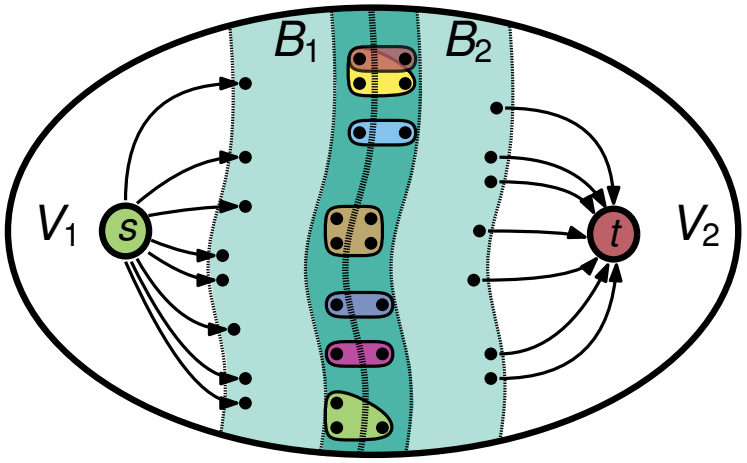
Solve Flow Problem

Find feasible minimum cut

Institute of Theoretical Informatics
Algorithmics Group

Our Flow-Based Refinement Framework

Select two adjacent blocks for refinement



Build Flow Problem

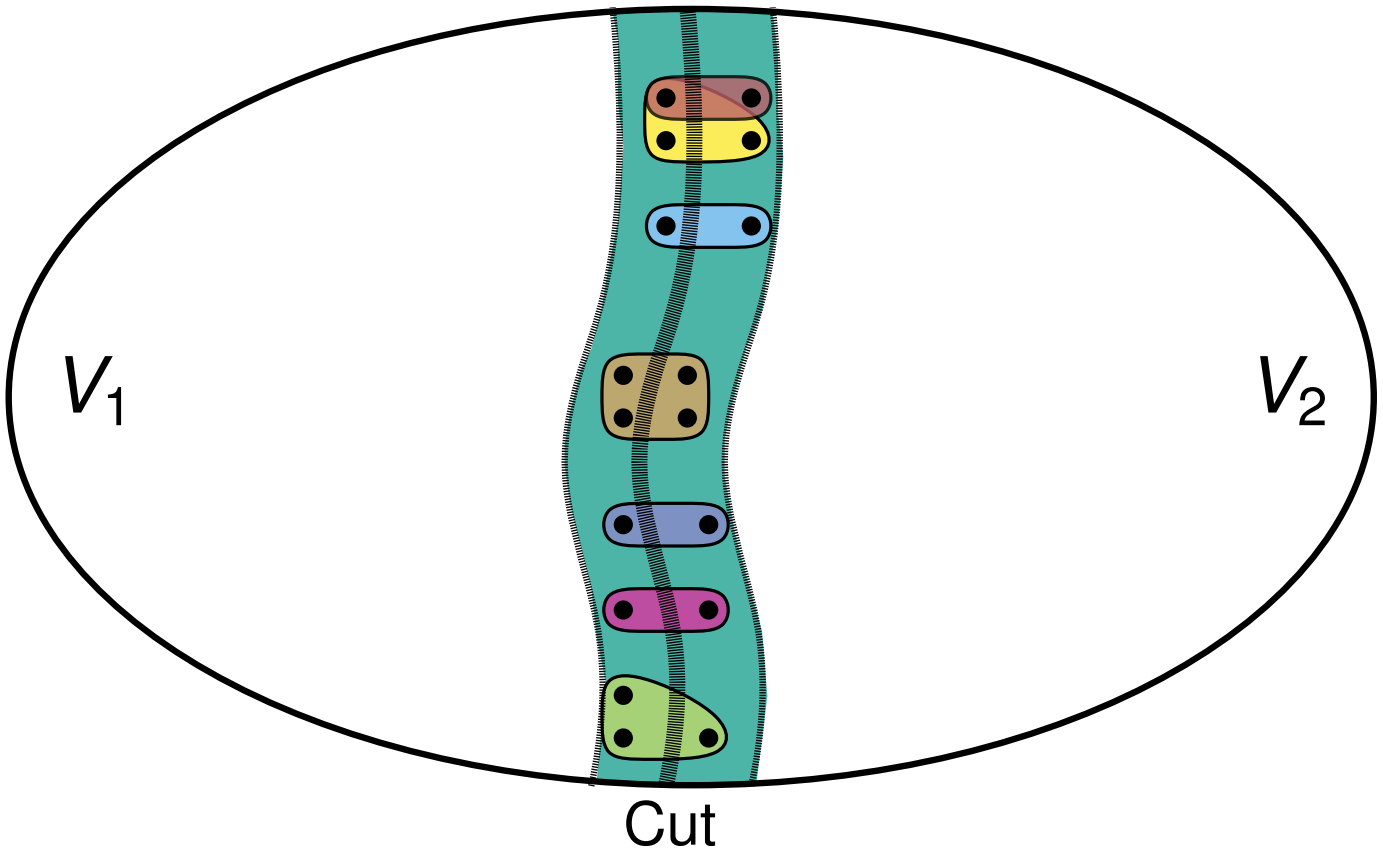
Solve Flow Problem

Find feasible minimum cut

Institute of Theoretical Informatics
Algorithmics Group

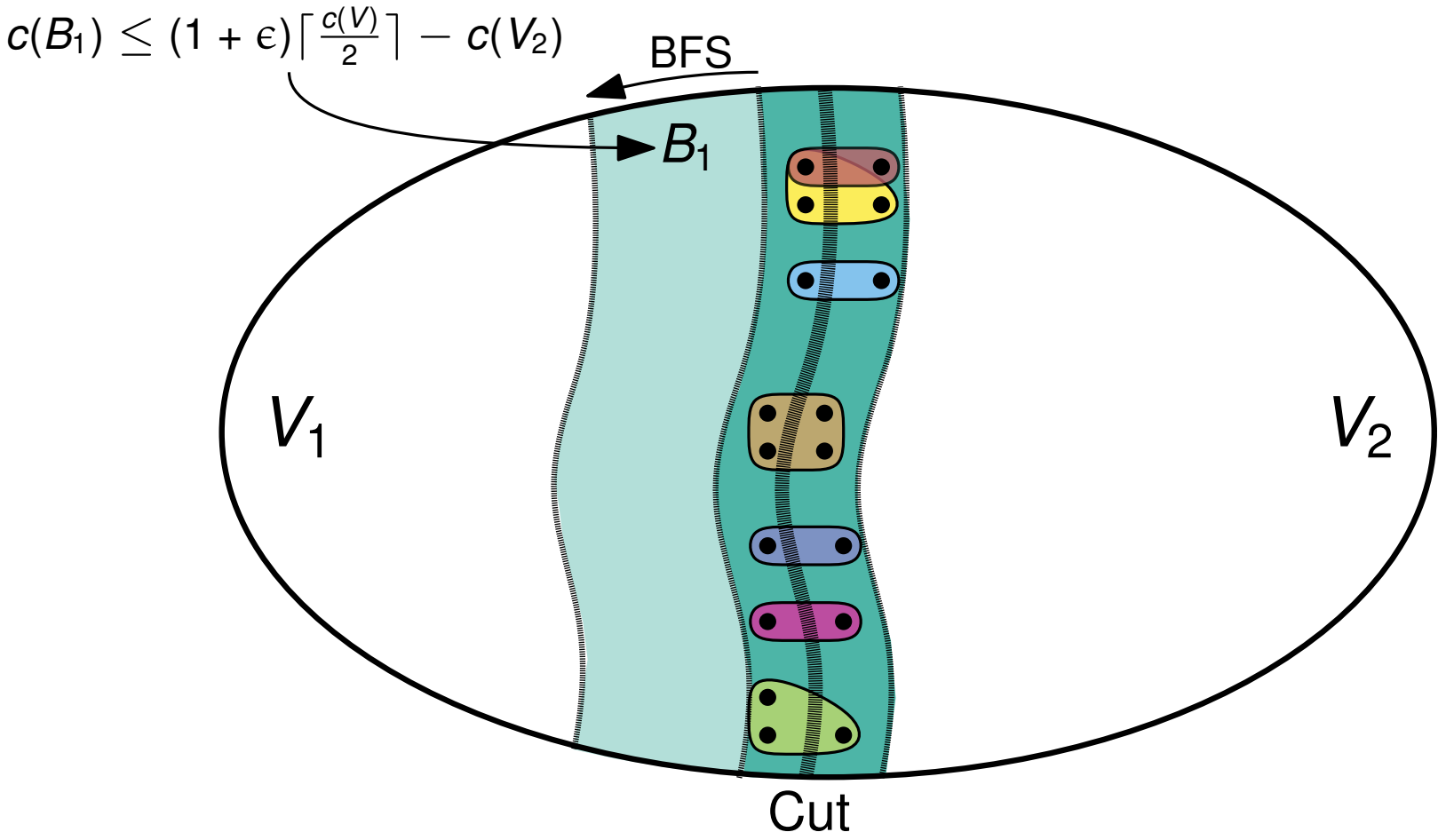
Adaptive Flow Iterations

KaFFPa [Sanders, Schulz 11]



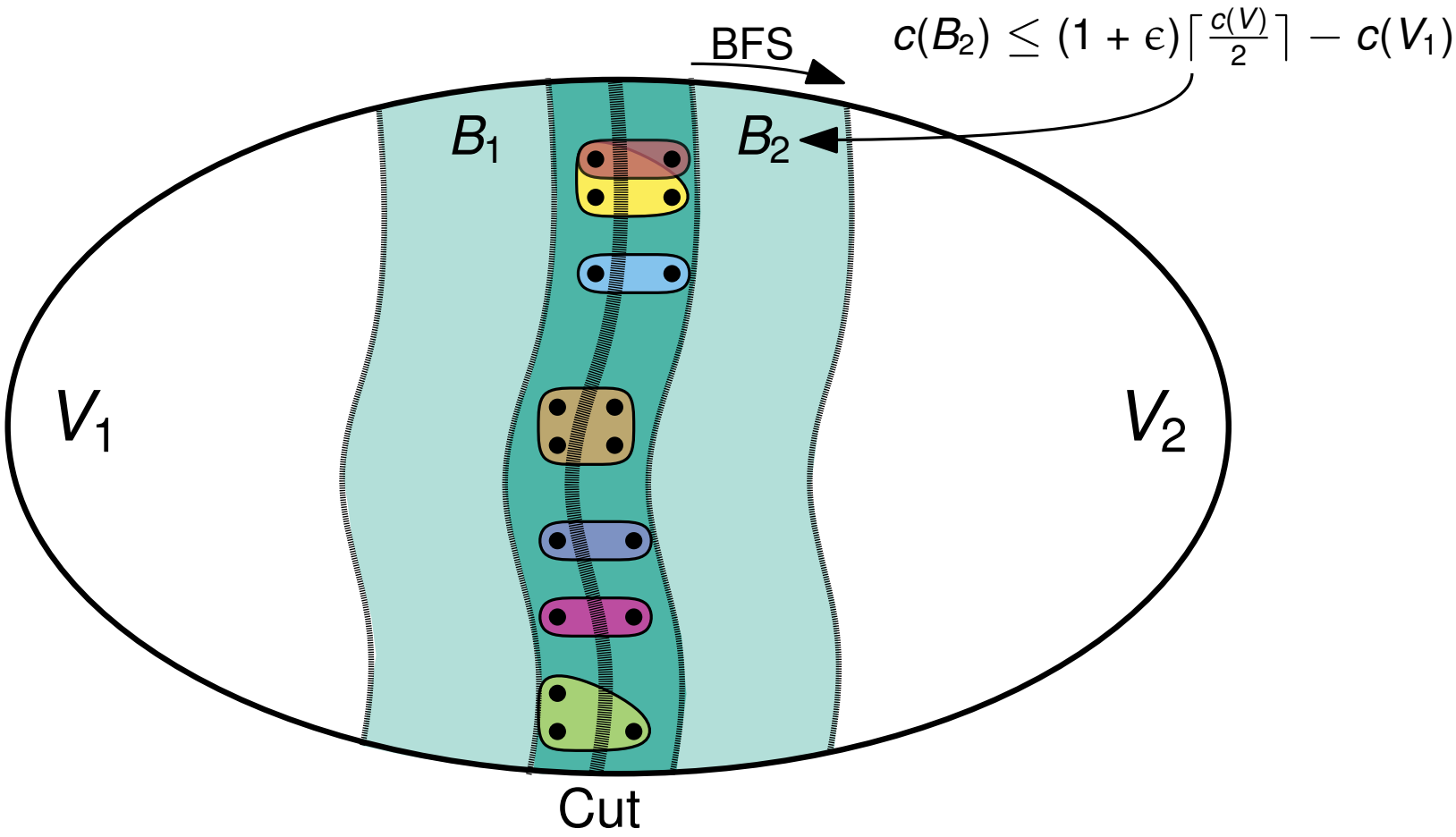
Adaptive Flow Iterations

KaFFPa [Sanders, Schulz 11]



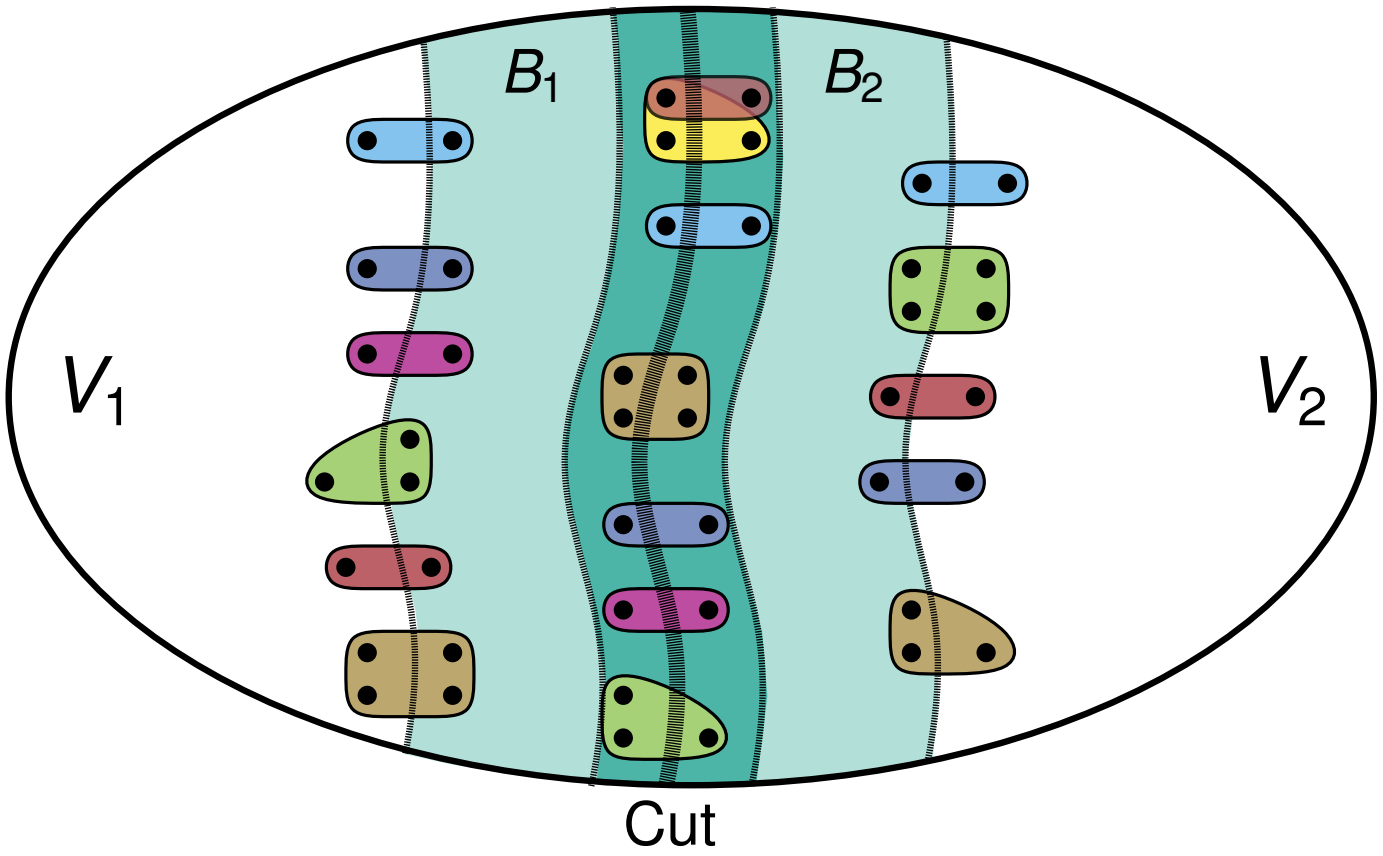
Adaptive Flow Iterations

KaFFPa [Sanders, Schulz 11]



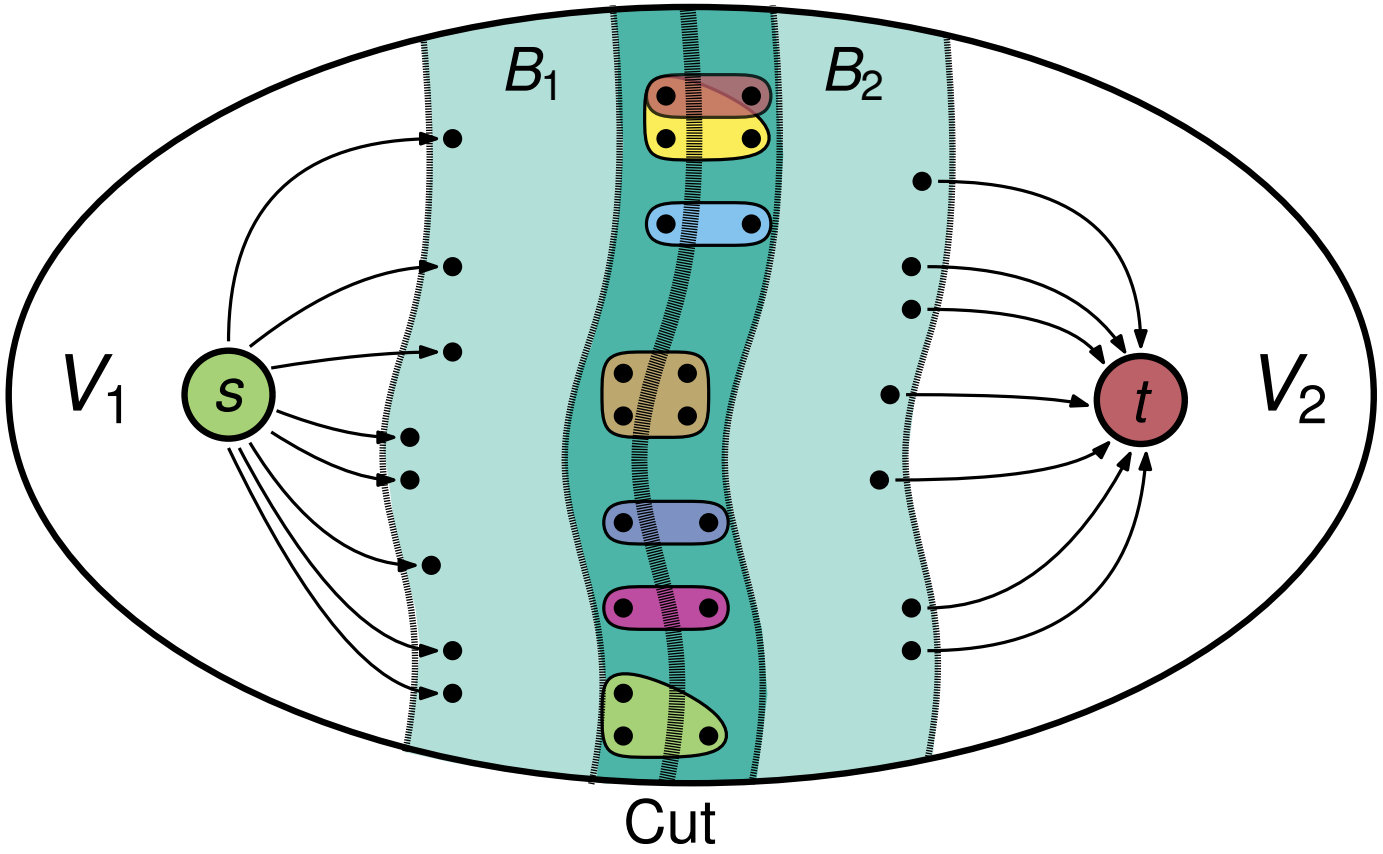
Adaptive Flow Iterations

KaFFPa [Sanders, Schulz 11]



Adaptive Flow Iterations

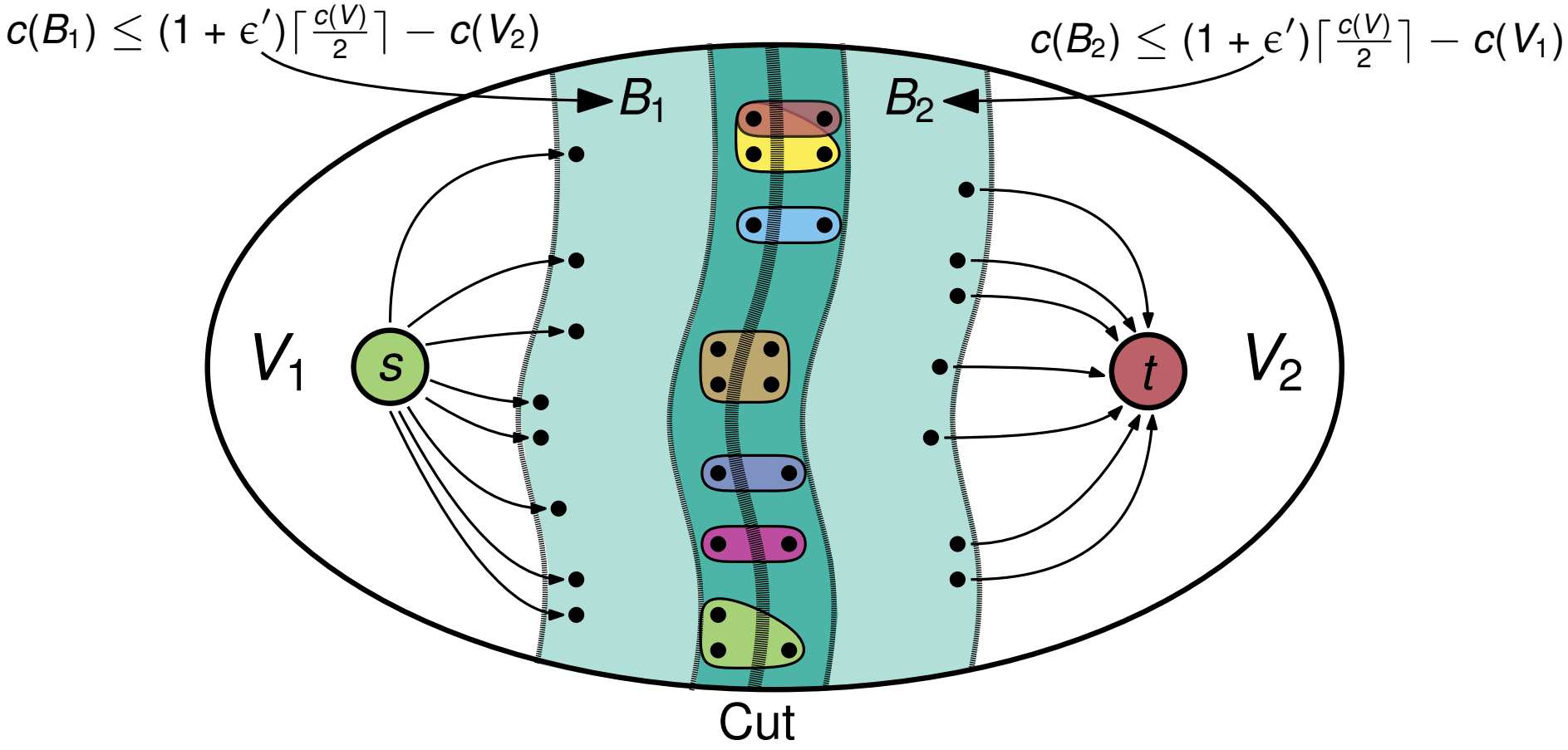
KaFFPa [Sanders, Schulz 11]



Adaptive Flow Iterations

KaFFPa [Sanders, Schulz 11]

Use $\epsilon' = \alpha\epsilon$ instead of ϵ



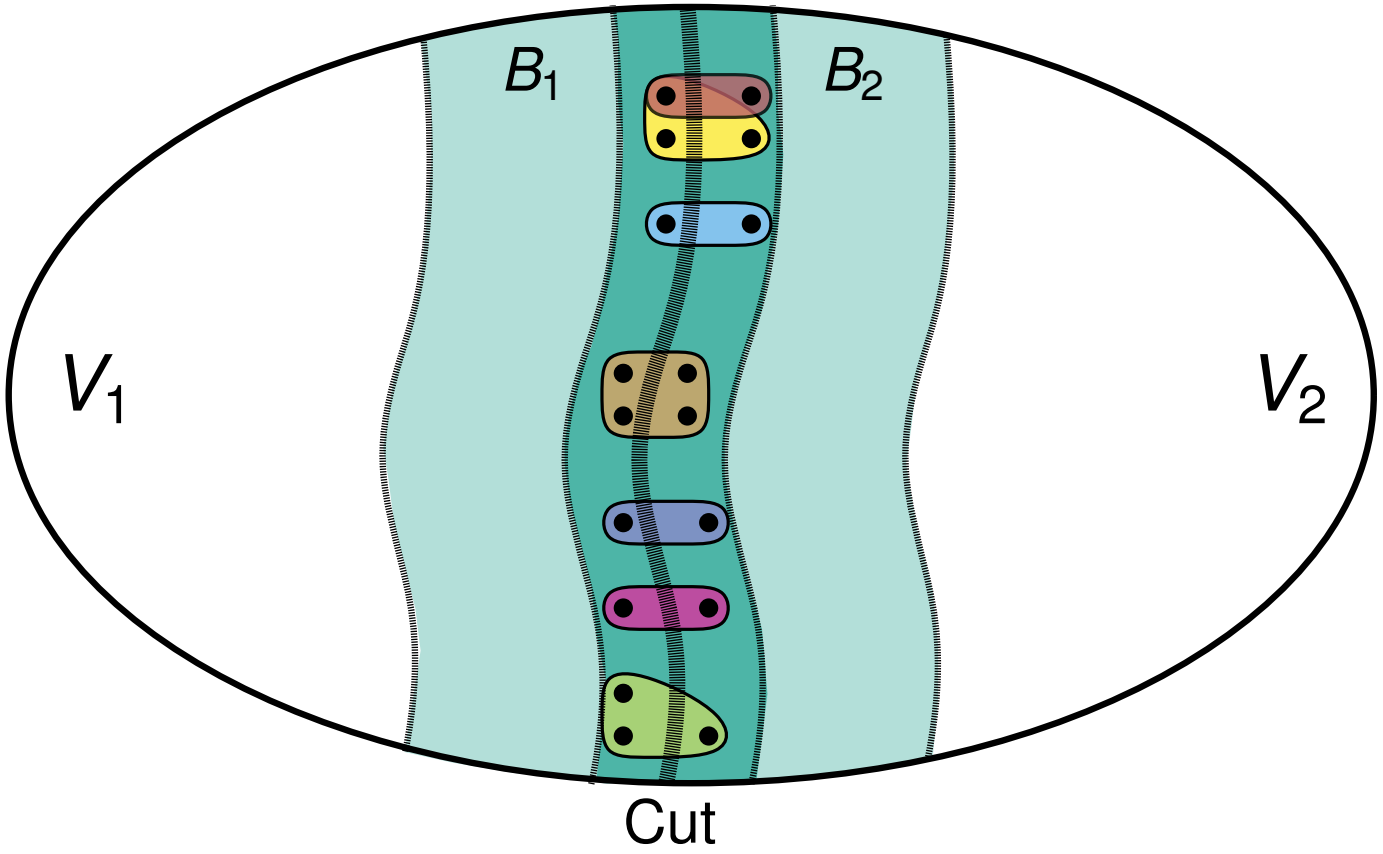
Institute of Theoretical Informatics
Algorithmics Group

Adaptive Flow Iterations

KaFFPa [Sanders, Schulz 11]

Use $\epsilon' = \alpha\epsilon$ instead of ϵ

$\alpha = 1 \Rightarrow$ **Improvement Found** $\Rightarrow \alpha = \min(2\alpha, \alpha') = 2$



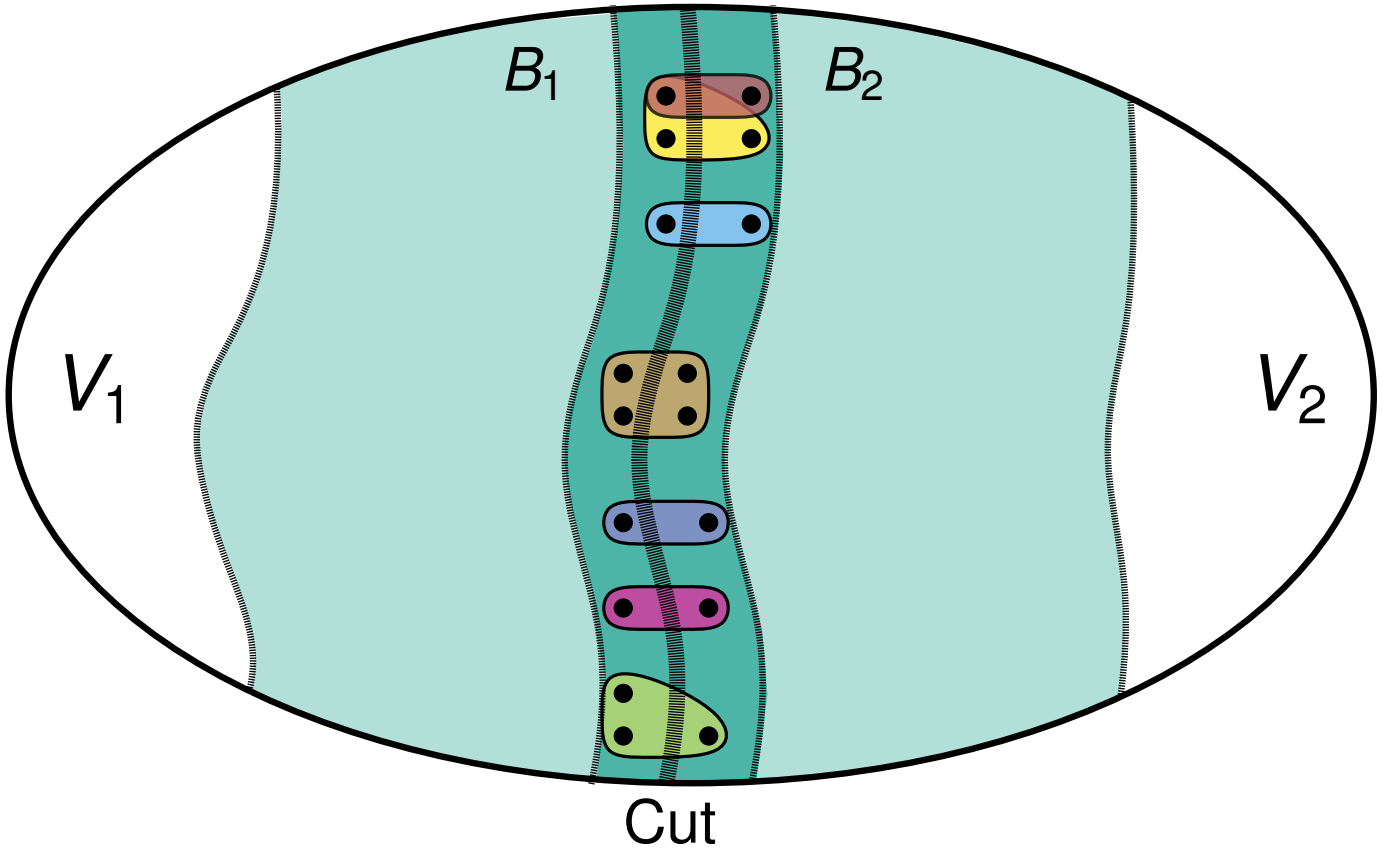
Institute of Theoretical Informatics
Algorithmics Group

Adaptive Flow Iterations

KaFFPa [Sanders, Schulz 11]

Use $\epsilon' = \alpha\epsilon$ instead of ϵ

$\alpha = 2 \Rightarrow$ **No Improvement** $\Rightarrow \alpha = \max(\frac{\alpha}{2}, 1) = 1$



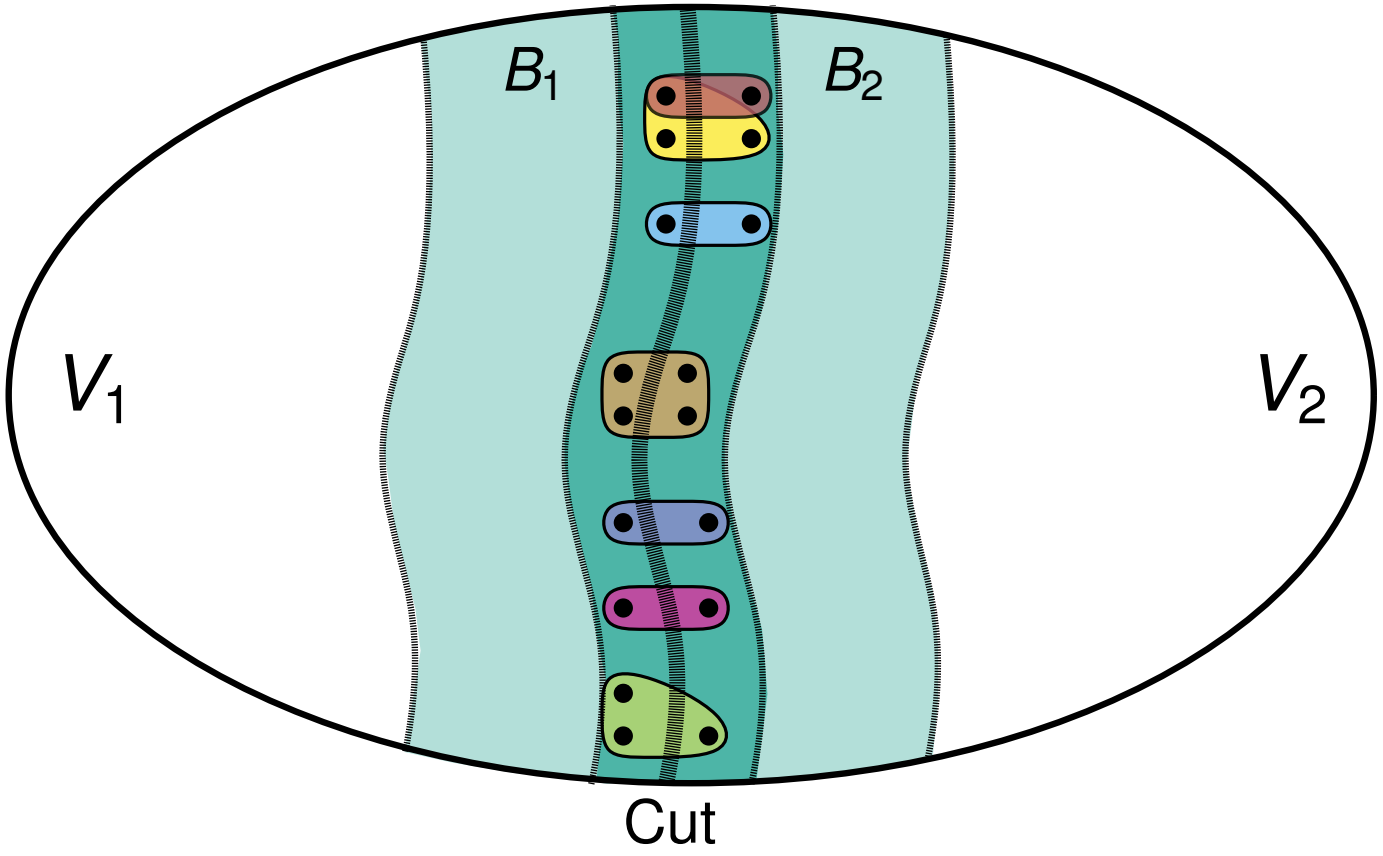
Institute of Theoretical Informatics
Algorithmics Group

Adaptive Flow Iterations

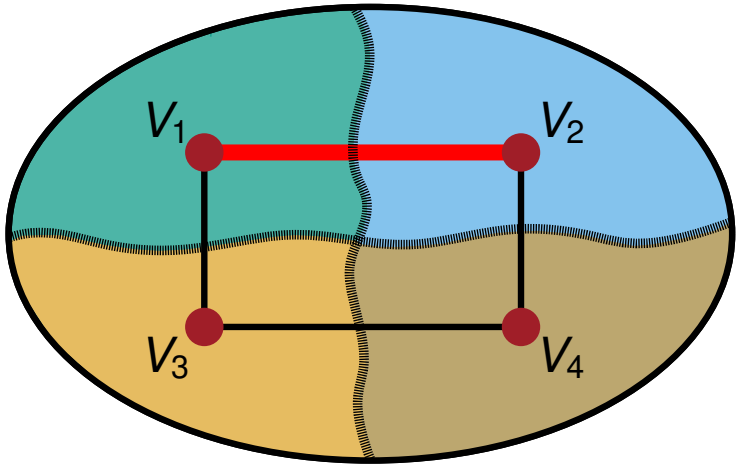
KaFFPa [Sanders, Schulz 11]

Use $\epsilon' = \alpha\epsilon$ instead of ϵ

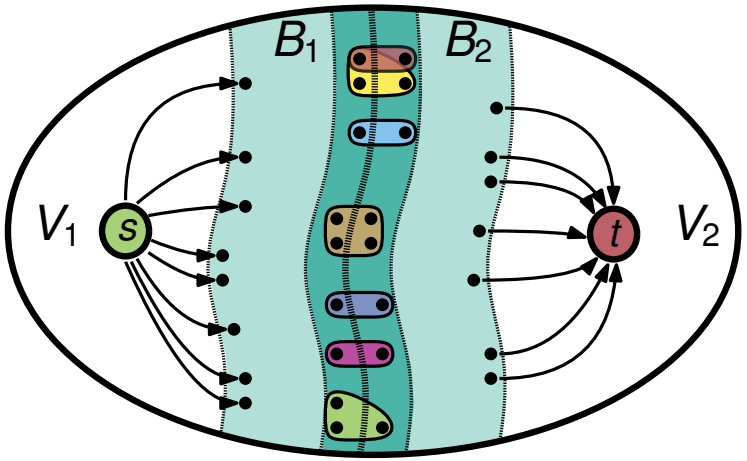
$\alpha = 1 \Rightarrow$ **No Improvement** $\Rightarrow$ **Terminate**



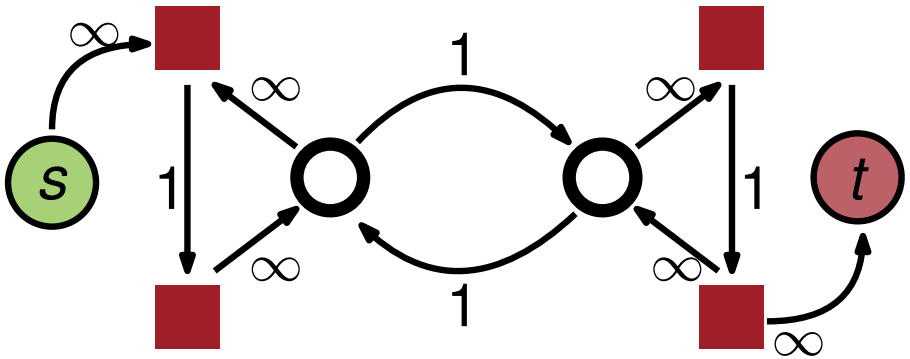
Our Flow-Based Refinement Framework



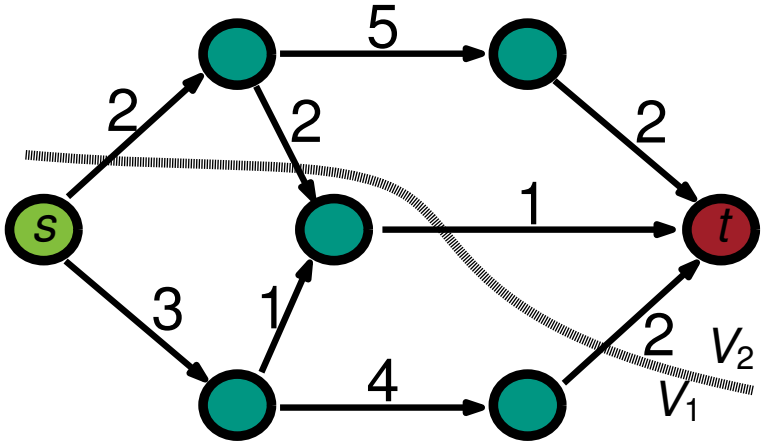
Select two adjacent blocks for refinement



Build Flow Problem



Solve Flow Problem



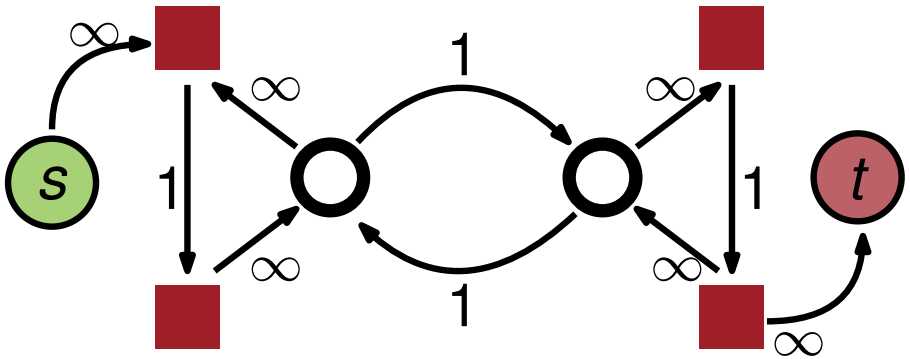
Find feasible minimum cut

Institute of Theoretical Informatics
Algorithmics Group

Our Flow-Based Refinement Framework

Select two adjacent blocks for refinement

Build Flow Problem

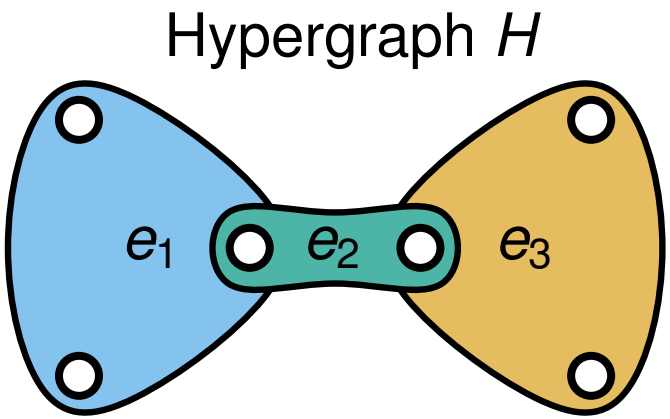


Solve Flow Problem

Find feasible minimum cut

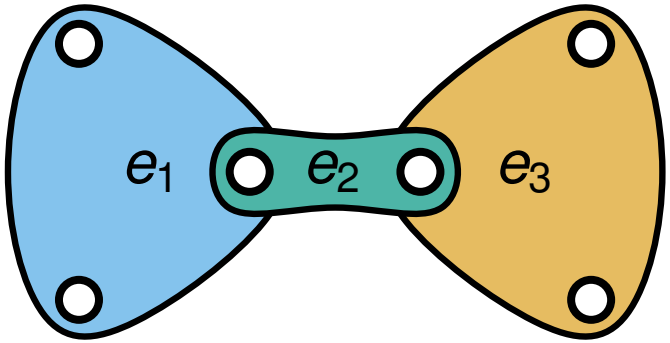
Institute of Theoretical Informatics
Algorithmics Group

Hypergraph Flow Network

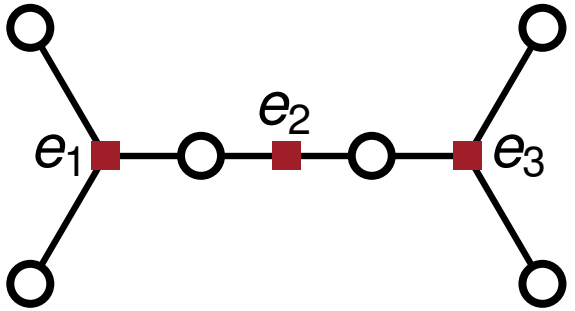


Hypergraph Flow Network

Hypergraph H

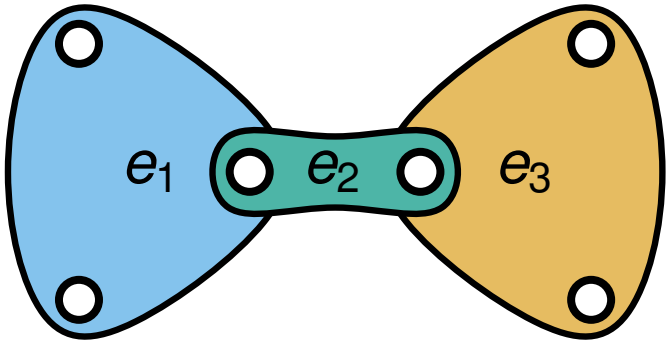


Bipartite Graph $G_*(H)$

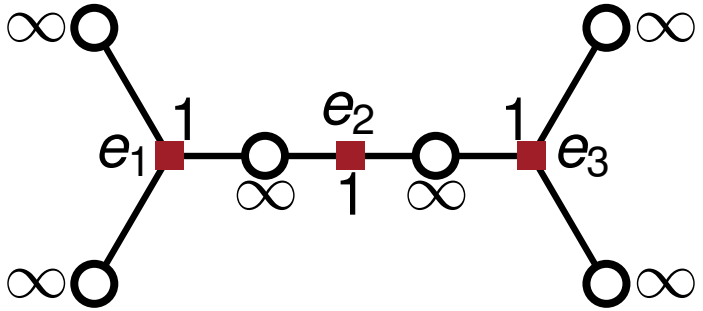


Hypergraph Flow Network

Hypergraph H



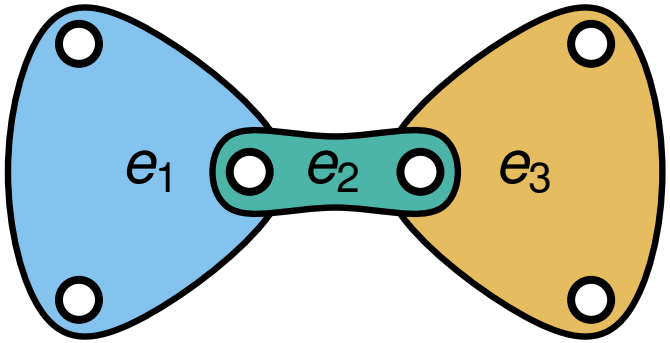
Bipartite Graph $G_*(H)$



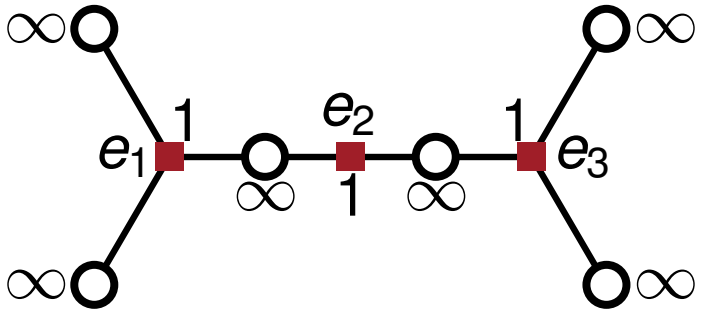
Vertex Separator Problem [Hu, Moerder 85]

Hypergraph Flow Network

Hypergraph H

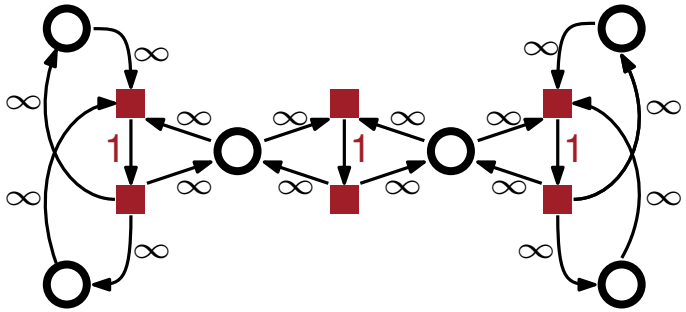


Bipartite Graph $G_*(H)$



Vertex Separator Problem [Hu, Moerder 85]

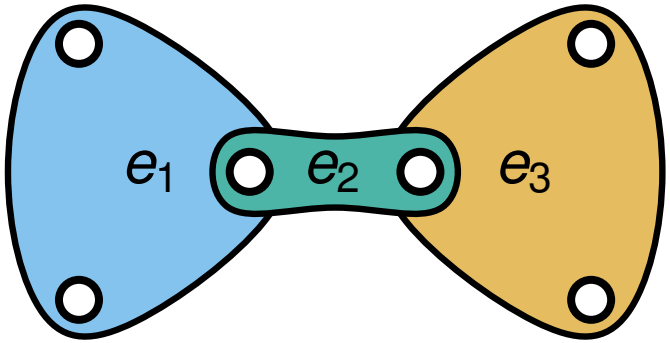
Lawler Network [Lawler 73]



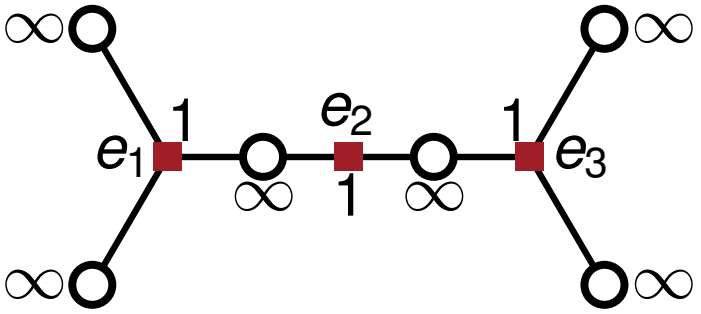
Institute of Theoretical Informatics
Algorithmics Group

Hypergraph Flow Network

Hypergraph H

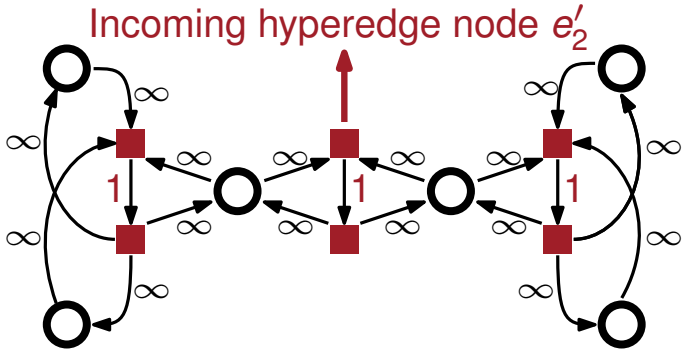


Bipartite Graph $G_*(H)$



Vertex Separator Problem [Hu, Moerder 85]

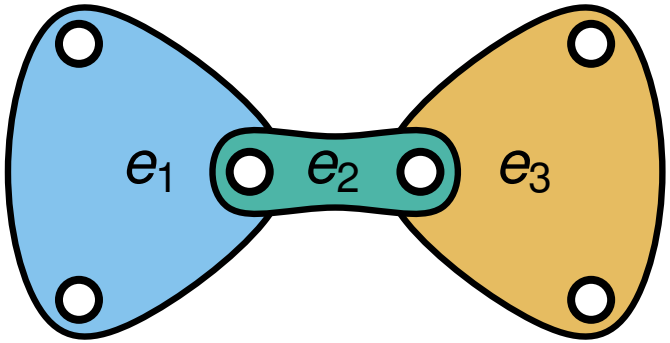
Lawler Network [Lawler 73]



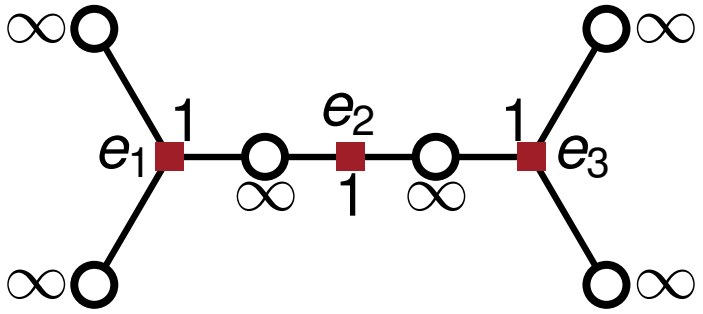
Institute of Theoretical Informatics
Algorithmics Group

Hypergraph Flow Network

Hypergraph H

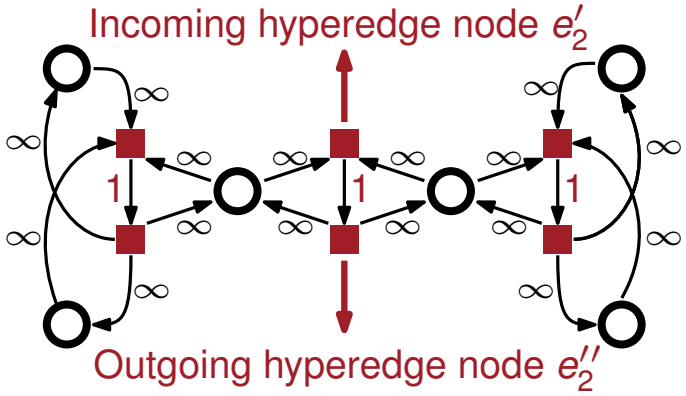


Bipartite Graph $G_*(H)$



Vertex Separator Problem [Hu, Moerder 85]

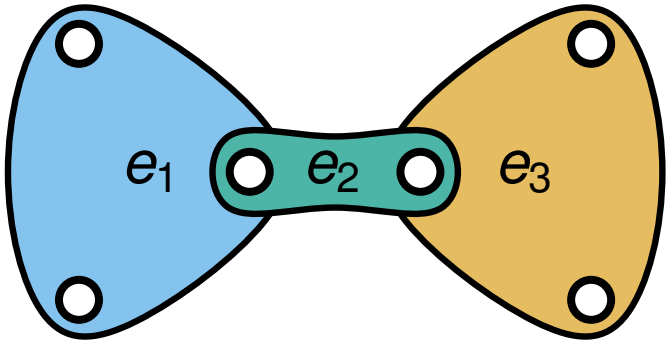
Lawler Network [Lawler 73]



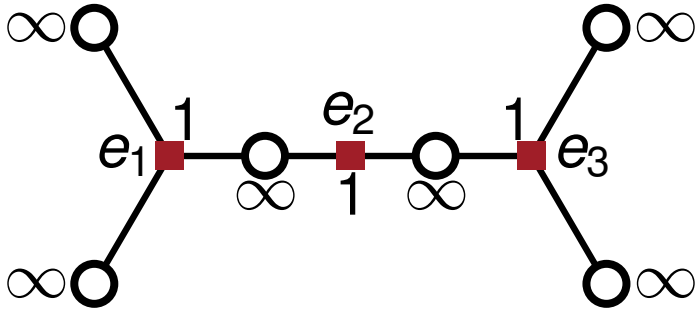
Institute of Theoretical Informatics
Algorithmics Group

Hypergraph Flow Network - Low Degree Vertices

Hypergraph H



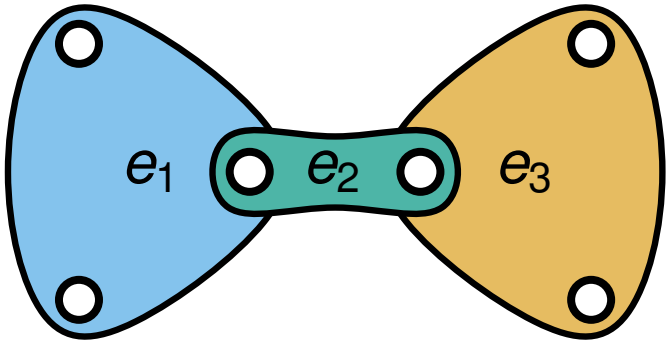
Bipartite Graph $G_*(H)$



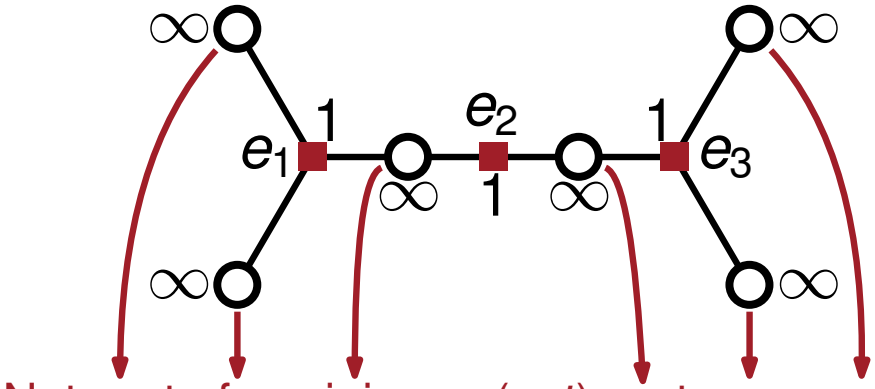
Institute of Theoretical Informatics
Algorithmics Group

Hypergraph Flow Network - Low Degree Vertices

Hypergraph H



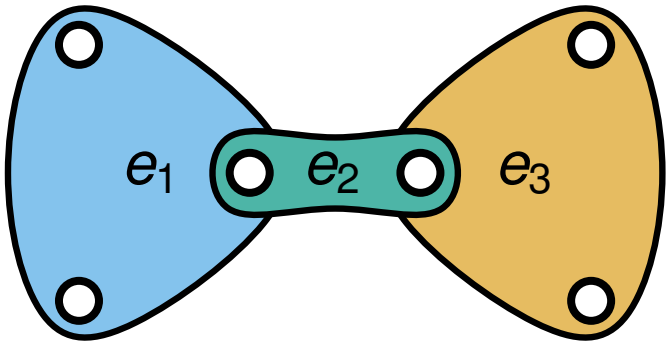
Bipartite Graph $G_*(H)$

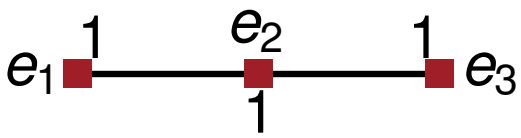


Not part of a minimum (s, t)-vertex separator

Hypergraph Flow Network - Low Degree Vertices

Hypergraph H

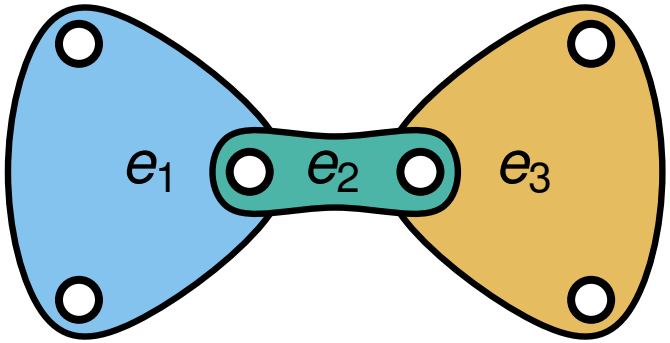


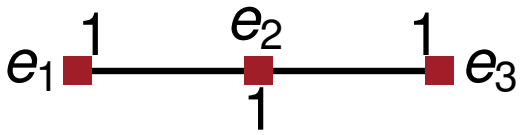


Remove all vertices by adding a clique

Hypergraph Flow Network - Low Degree Vertices

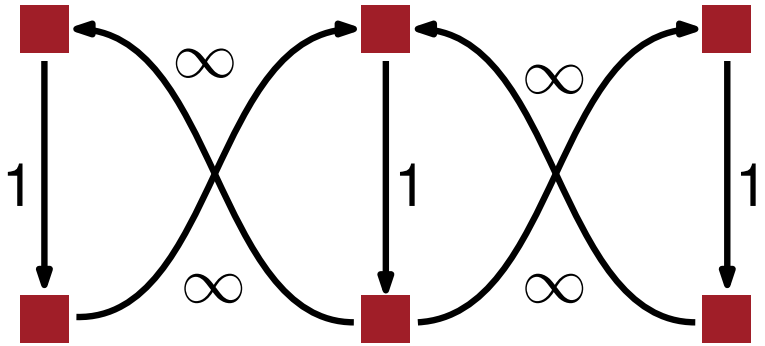
Hypergraph H





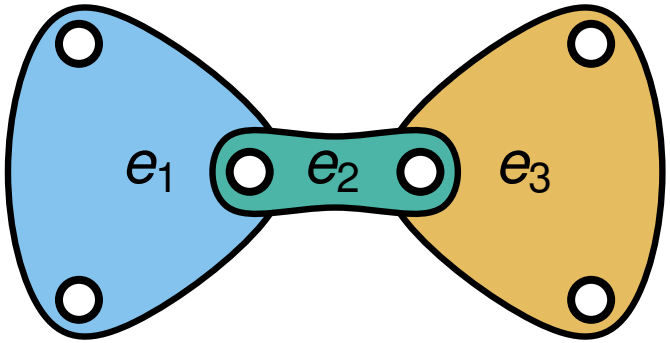
Remove all vertices by adding a clique

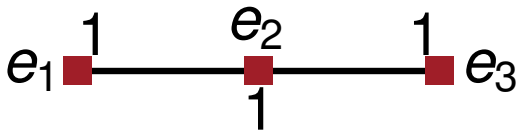
Our Network



Hypergraph Flow Network - Low Degree Vertices

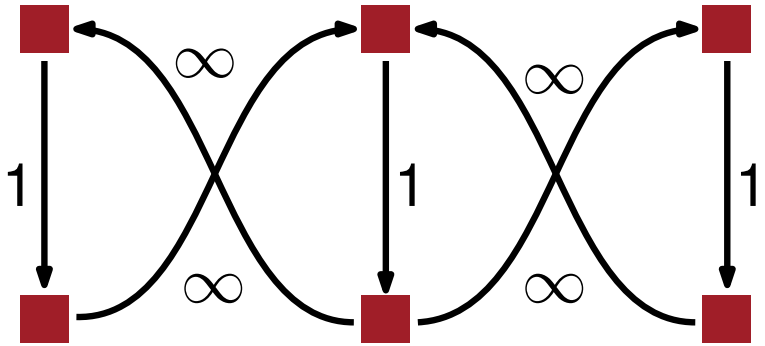
Hypergraph H





Remove all vertices by adding a clique

Our Network



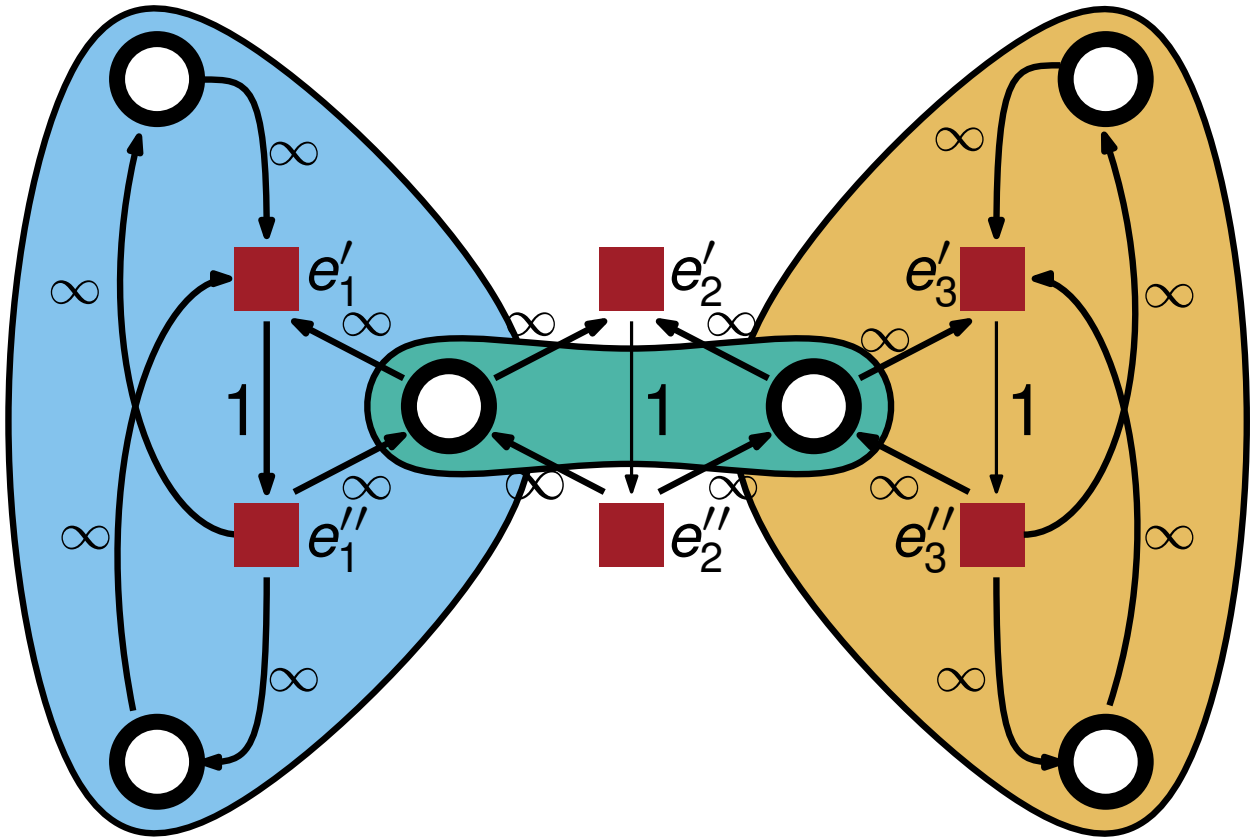
A hypernode v induces . . .

- . . . $2d(v)$ 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

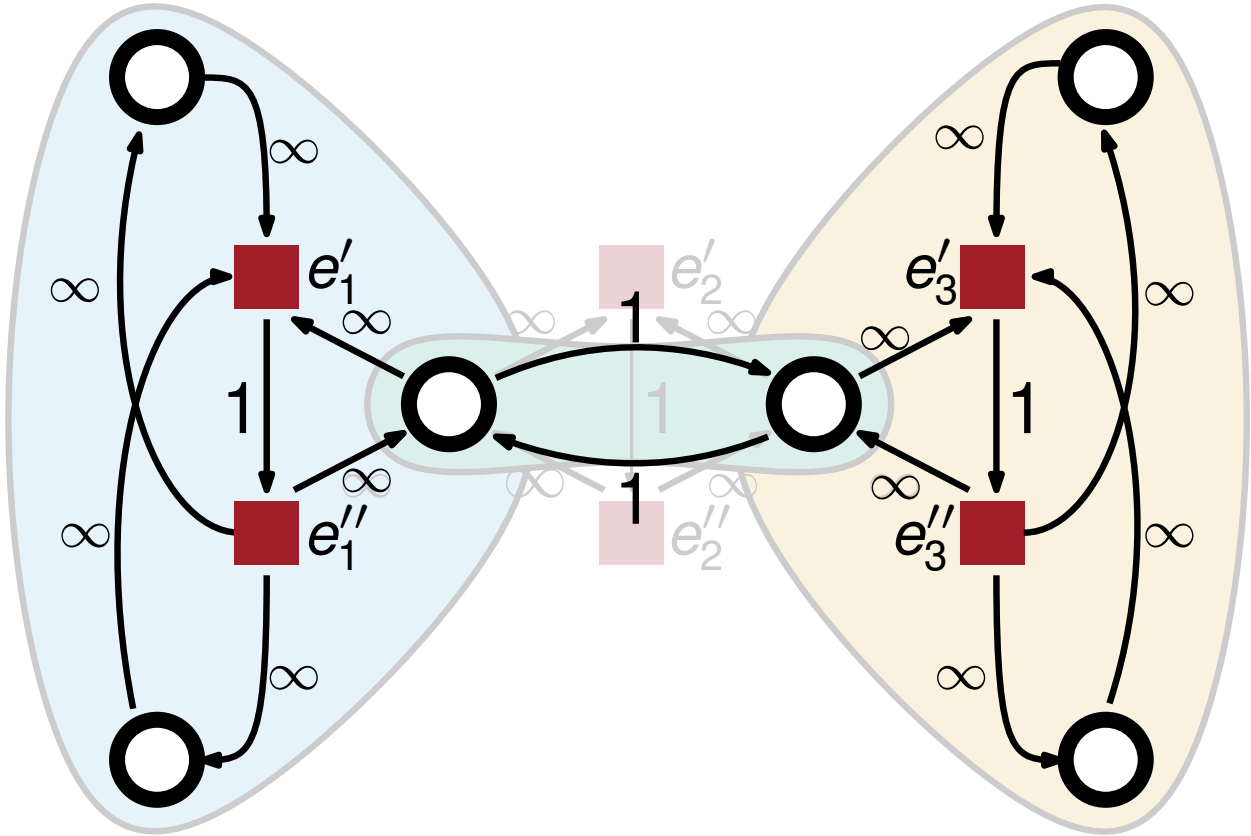
Hypergraph Flow Network - Summary



Lawler Network [Lawler 73]

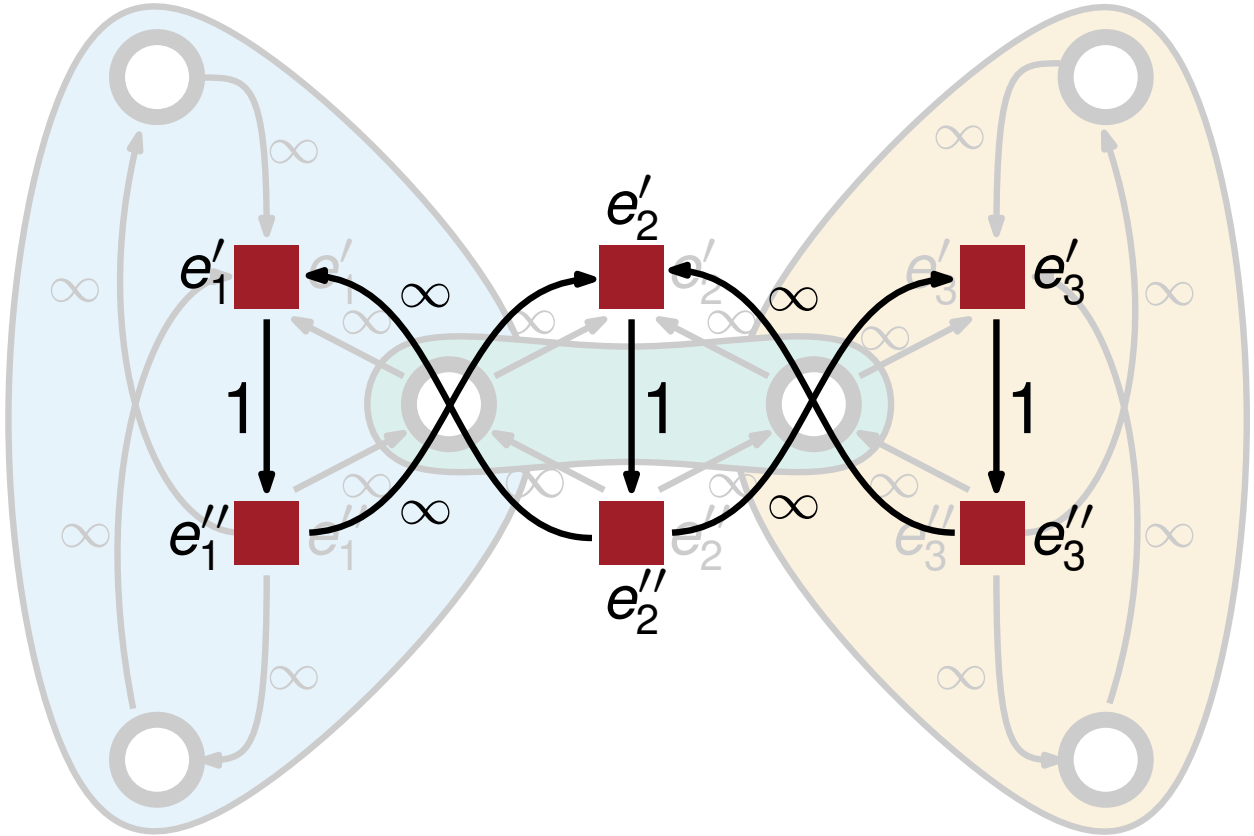
Institute of Theoretical Informatics
Algorithmics Group

Hypergraph Flow Network - Summary



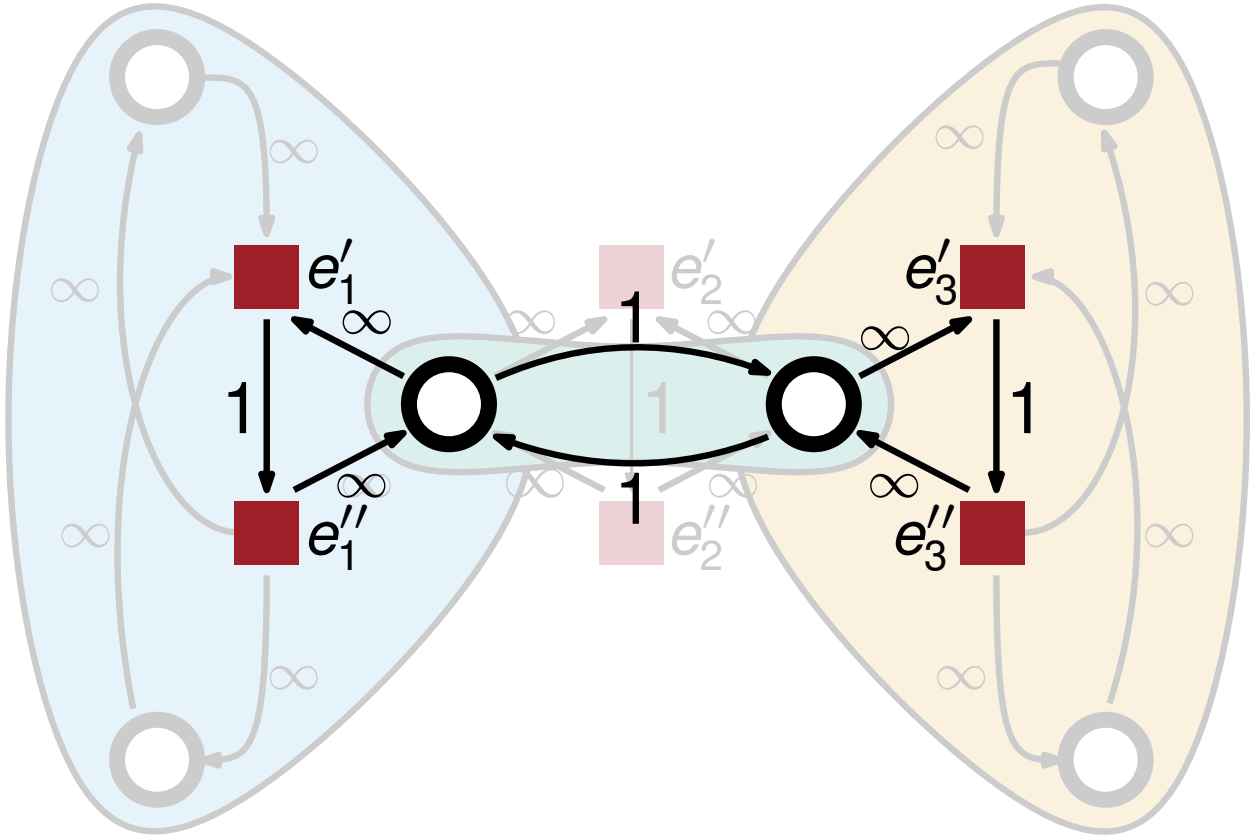
Wong Network [Liu, Wong 98]

Hypergraph Flow Network - Summary



Our Network

Hypergraph Flow Network - Summary

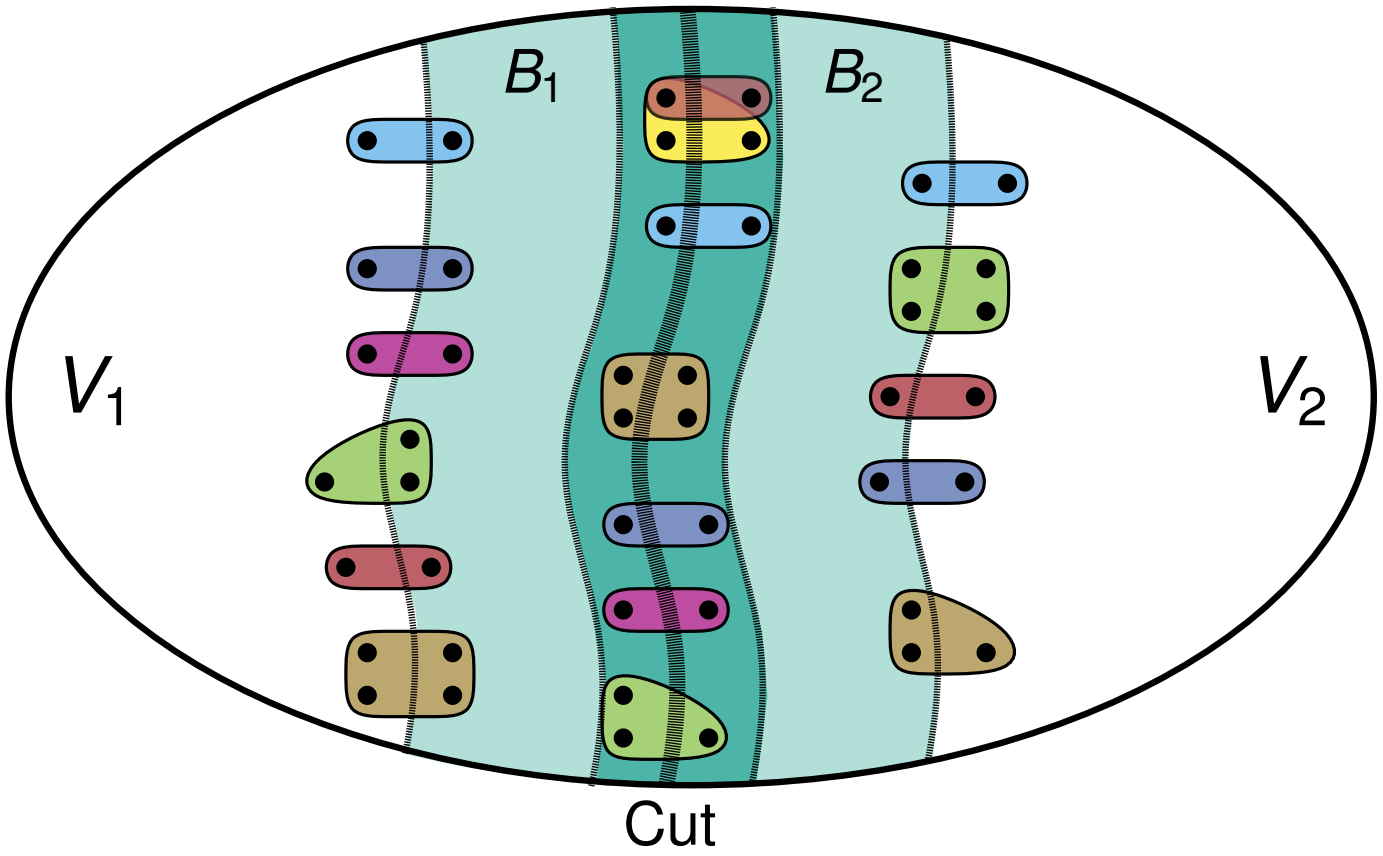


Hybrid Network

Institute of Theoretical Informatics
Algorithmics Group

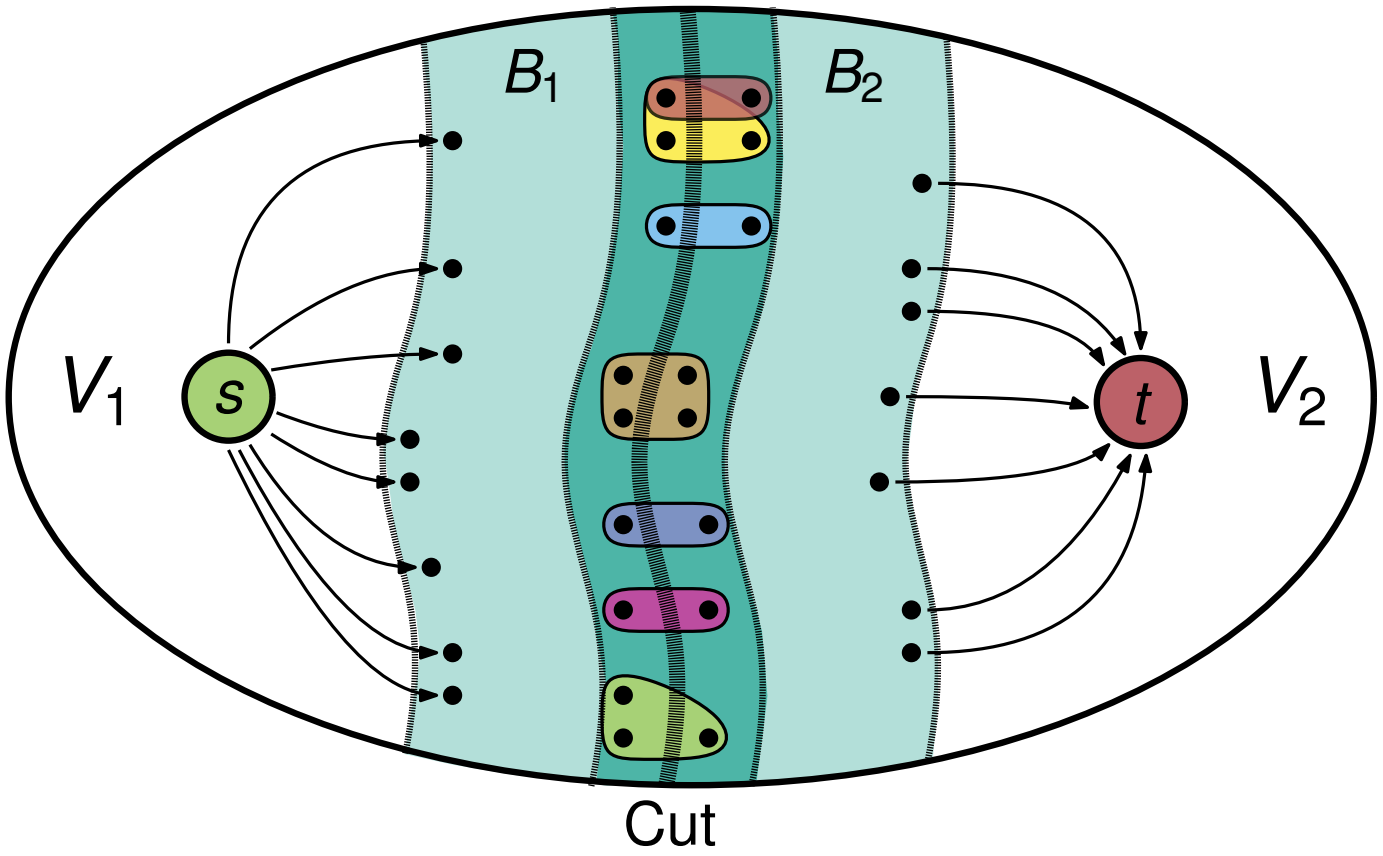
Optimized Flow Problem Modeling Approach

Modeling Approach in *KaFFPa*



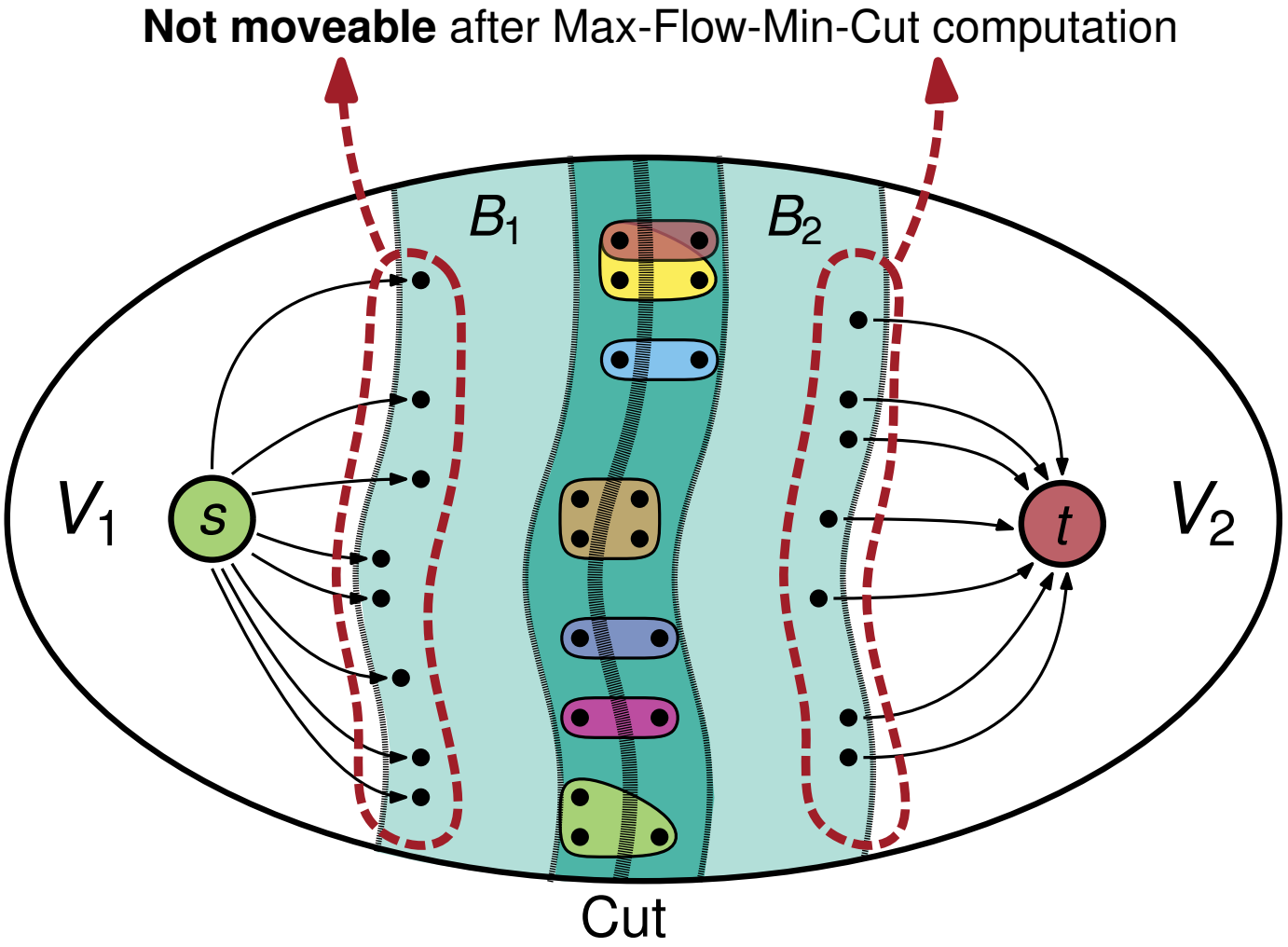
Optimized Flow Problem Modeling Approach

Modeling Approach in *KaFFPa*



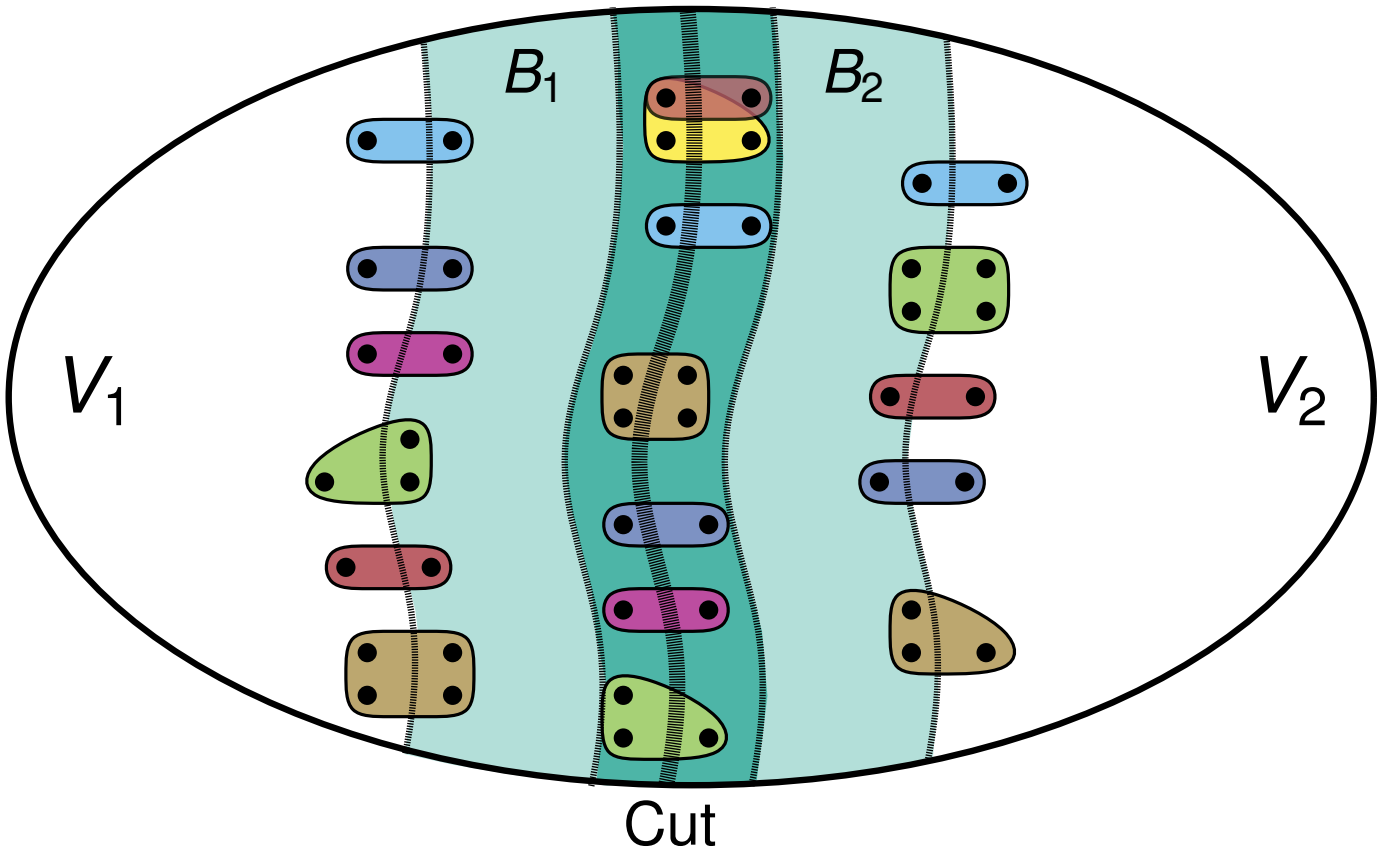
Optimized Flow Problem Modeling Approach

Modeling Approach in *KaFFPa*



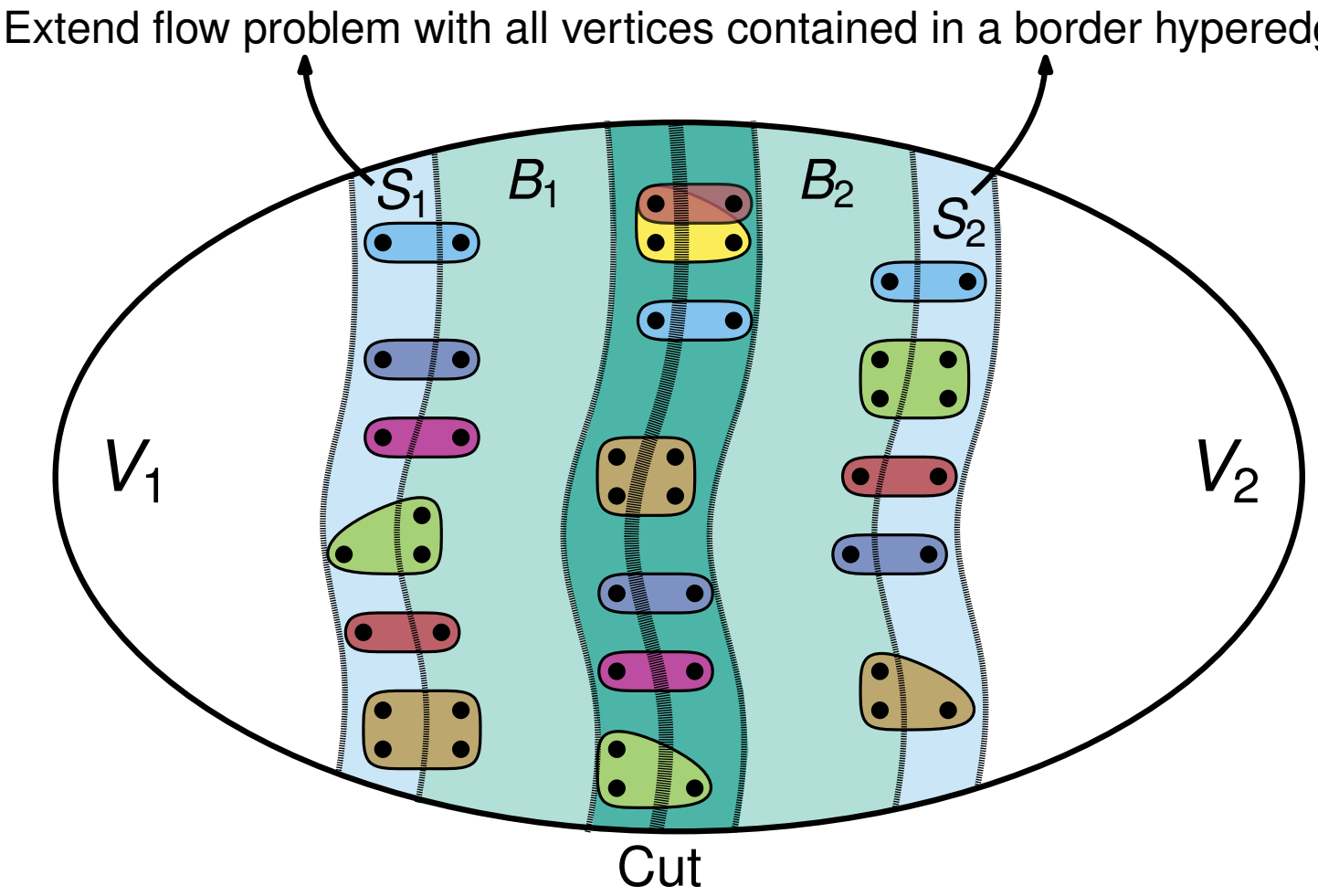
Optimized Flow Problem Modeling Approach

Modeling Approach in *KaHyPar*



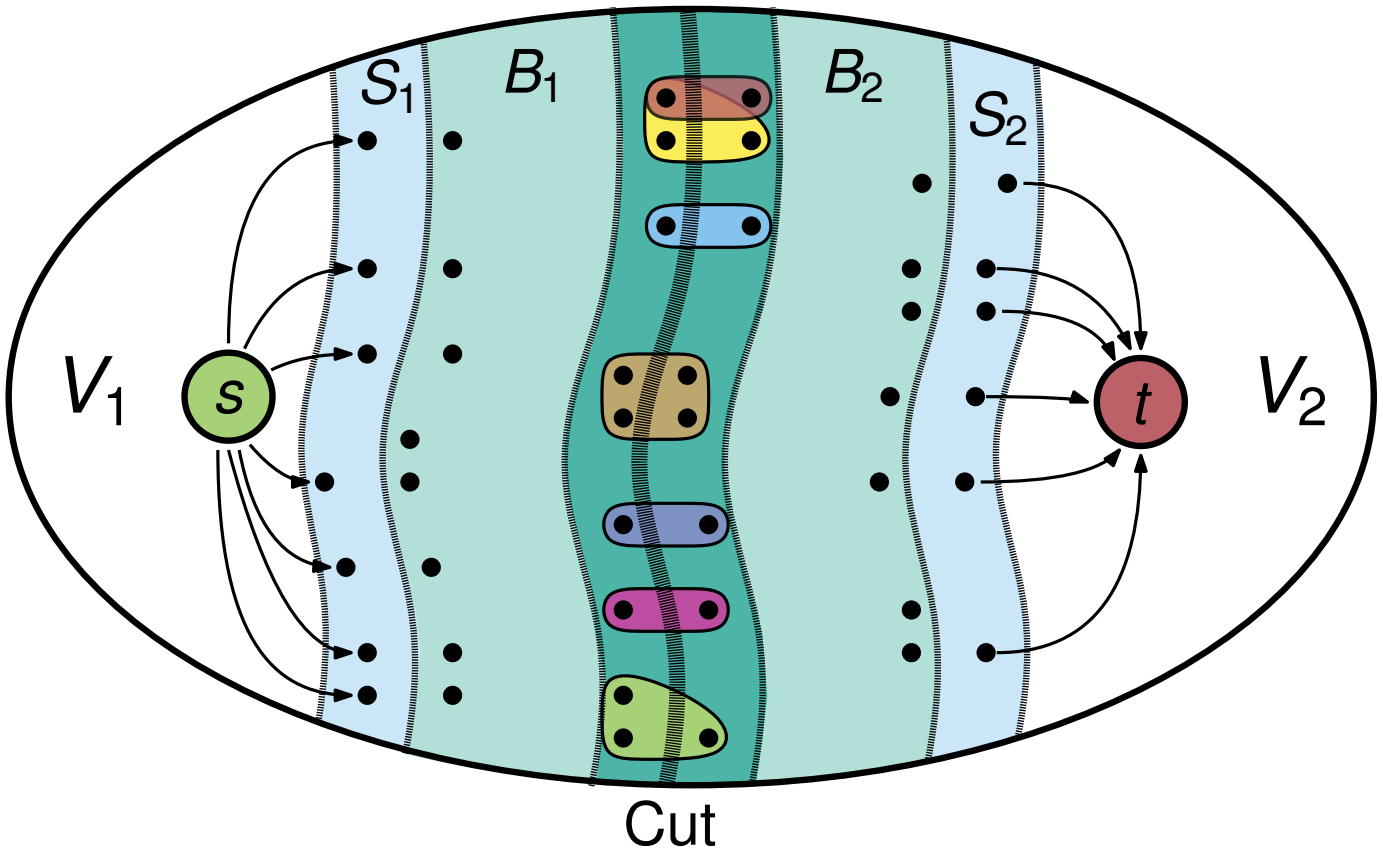
Optimized Flow Problem Modeling Approach

Modeling Approach in *KaHyPar*



Optimized Flow Problem Modeling Approach

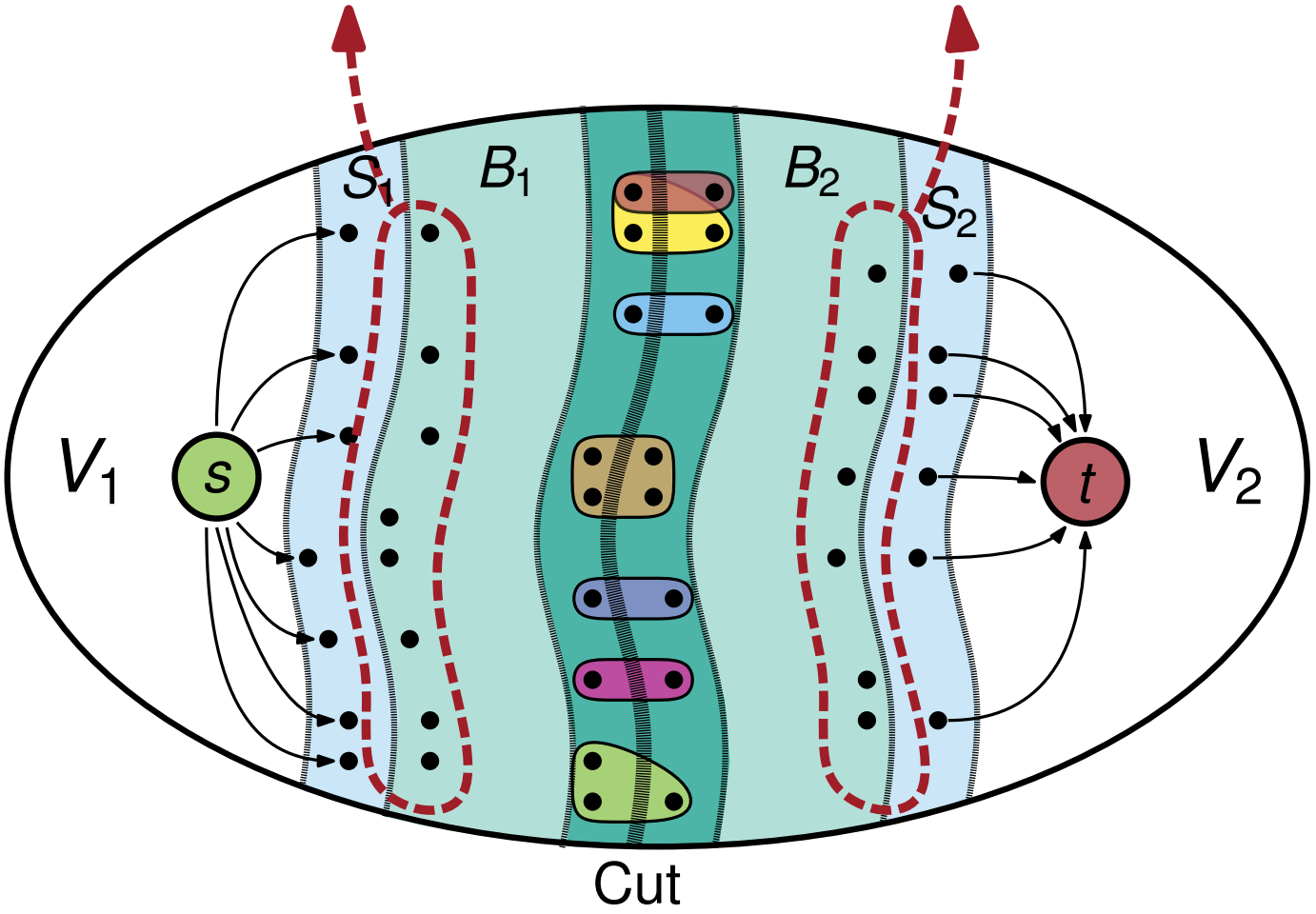
Modeling Approach in *KaHyPar*



Optimized Flow Problem Modeling Approach

Modeling Approach in *KaHyPar*

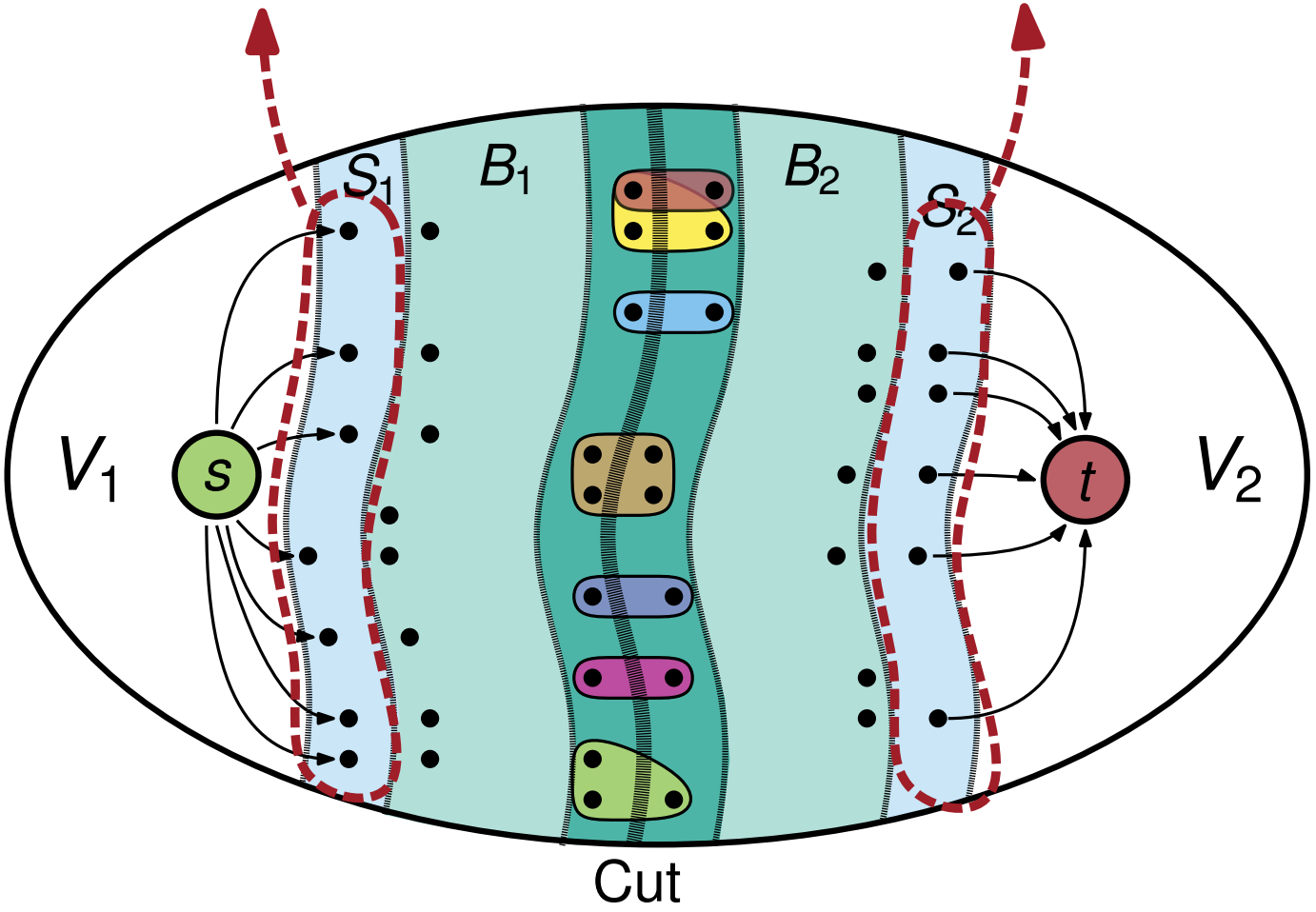
Moveable after Max-Flow-Min-Cut computation, but . . .



Optimized Flow Problem Modeling Approach

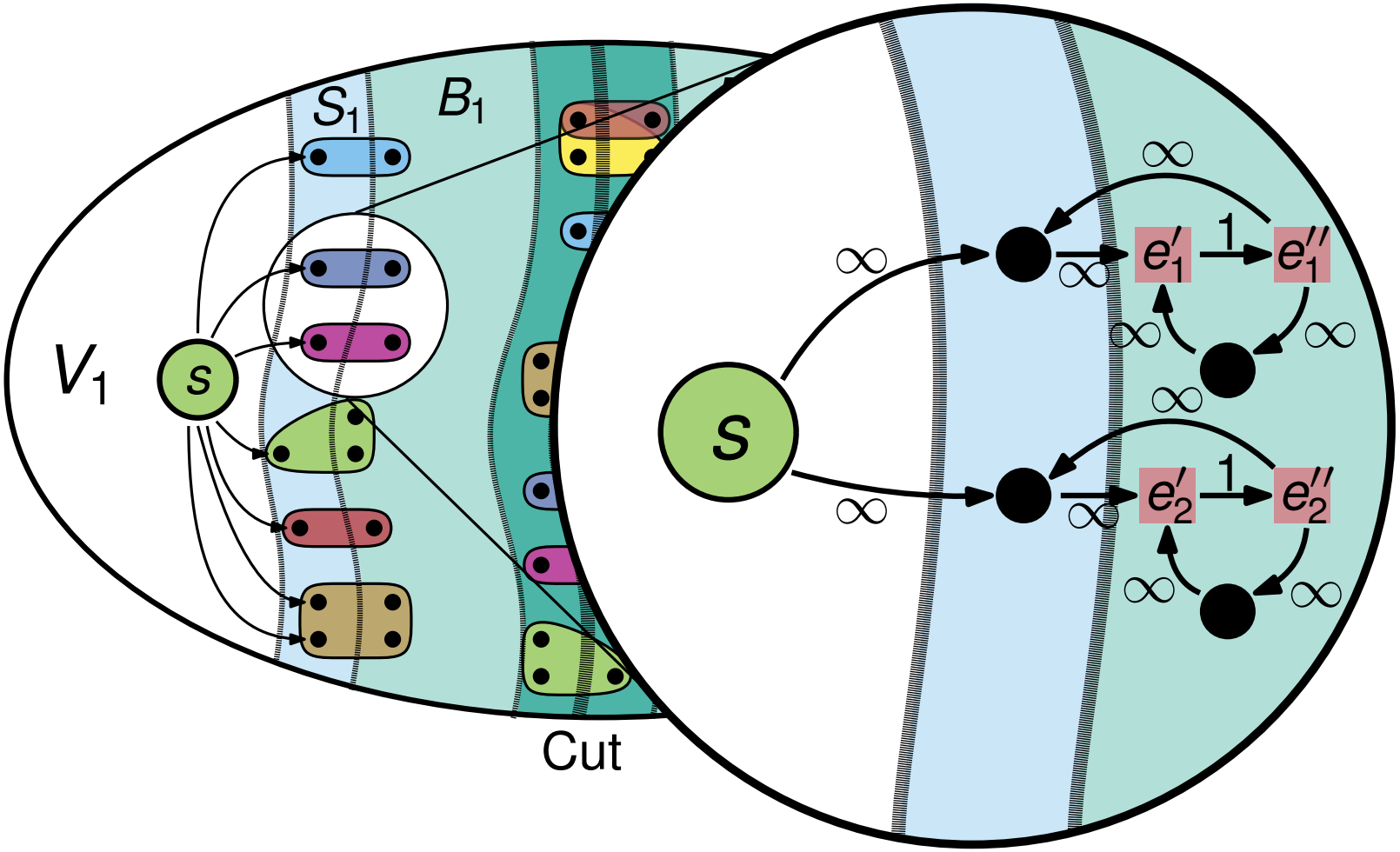
Modeling Approach in *KaHyPar*

. . . flow problem has significantly more nodes and edges.



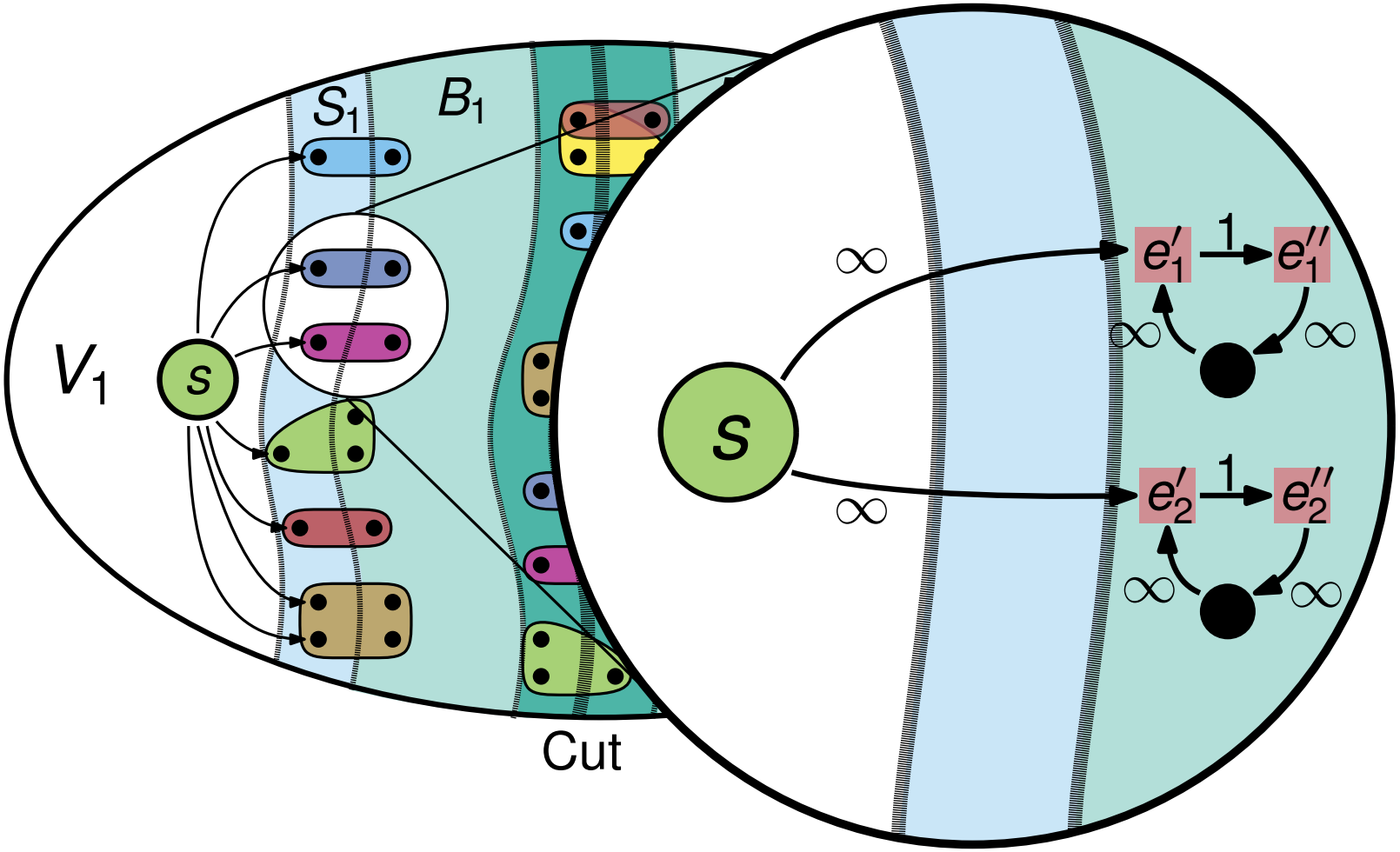
Optimized Flow Problem Modeling Approach

Modeling Approach in *KaHyPar*



Optimized Flow Problem Modeling Approach

Modeling Approach in *KaHyPar*

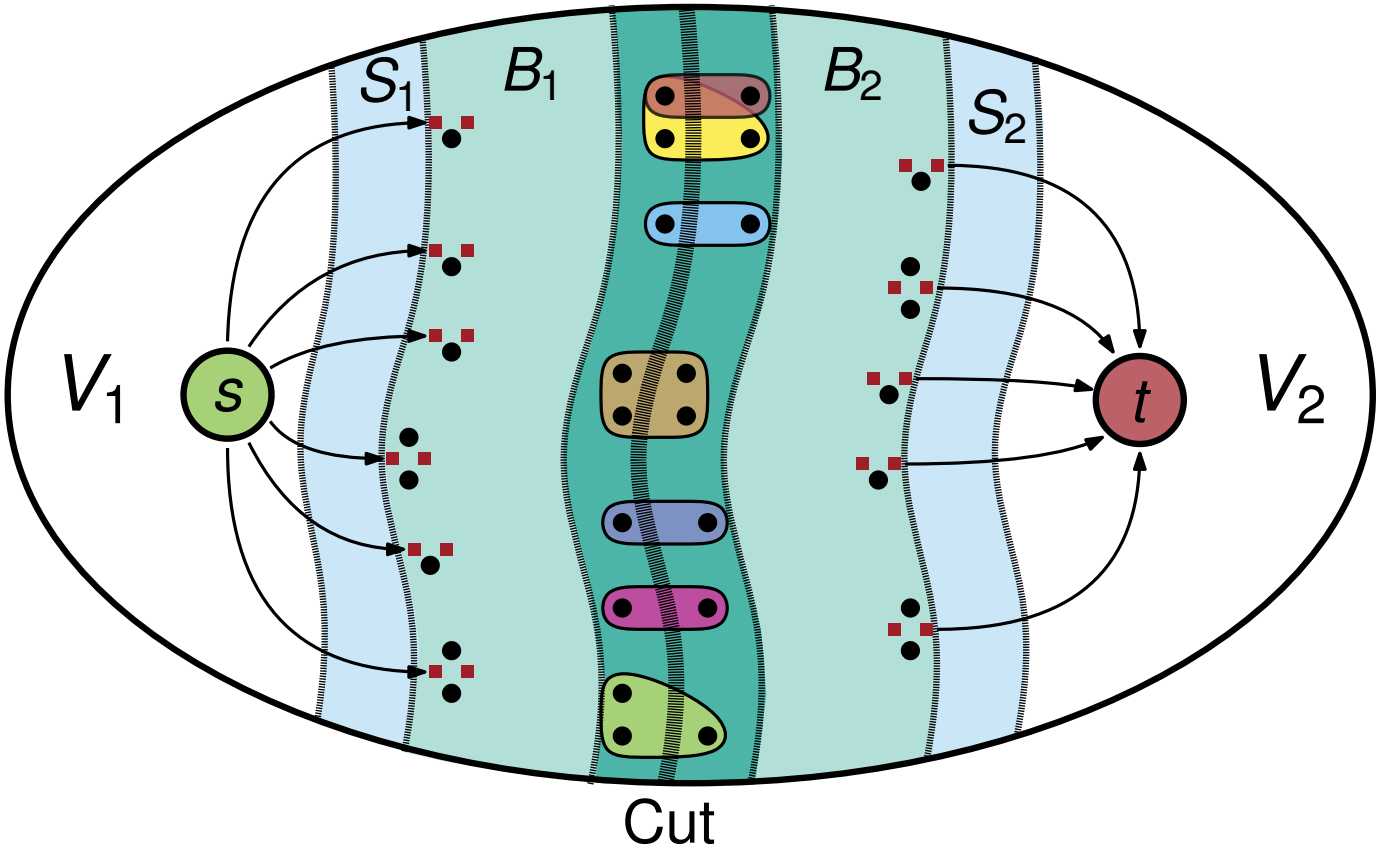


Optimized Flow Problem Modeling Approach

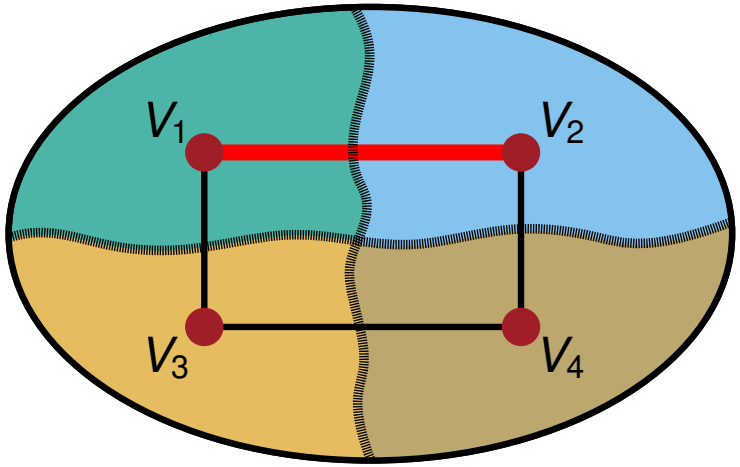
Modeling Approach in *KaHyPar*

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

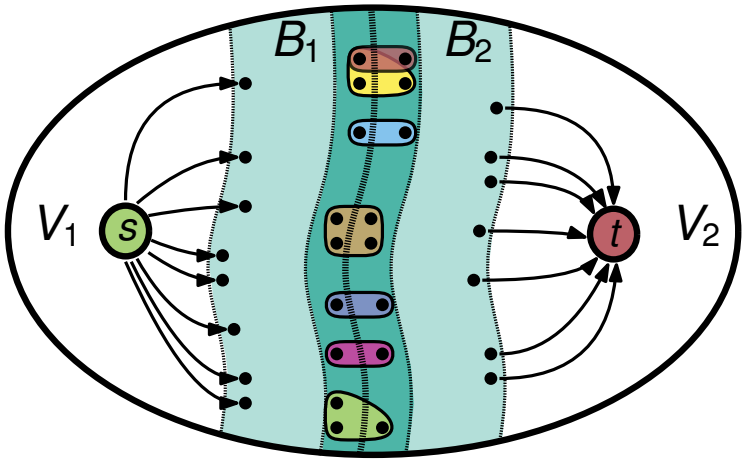
$$T = \{e'' \mid e \in I(S_2)\}$$



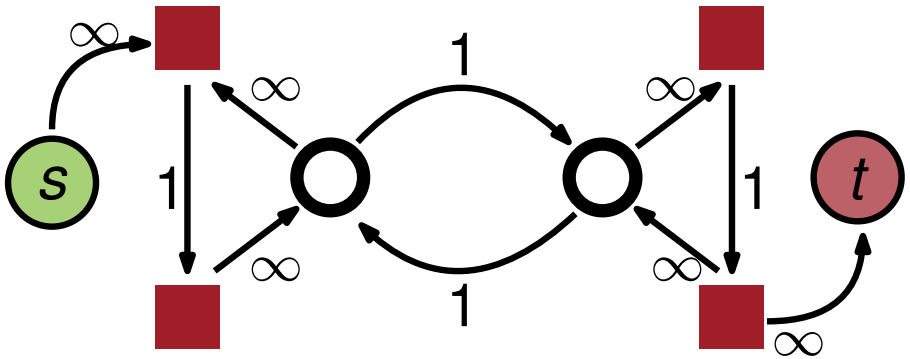
Our Flow-Based Refinement Framework



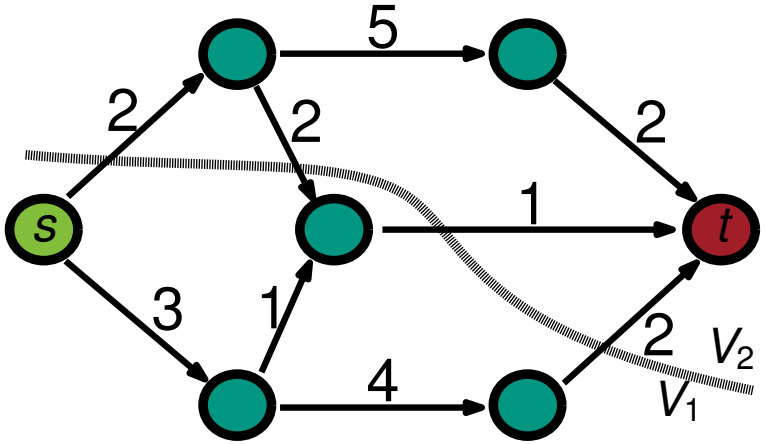
Select two adjacent blocks for refinement



Build Flow Problem



Solve Flow Problem



Find feasible minimum cut

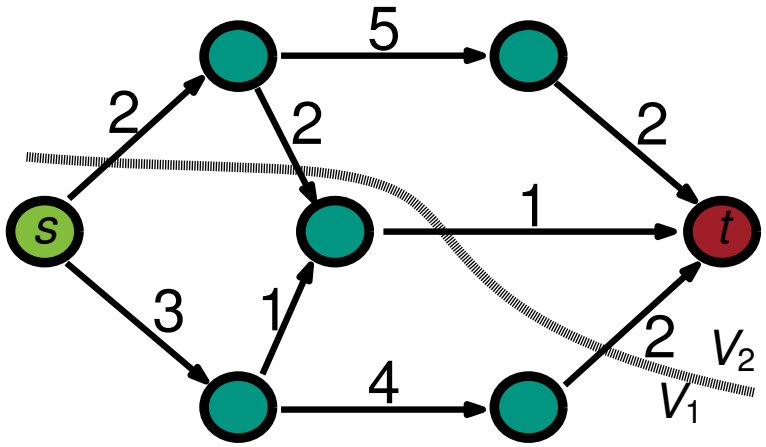
Institute of Theoretical Informatics
Algorithmics Group

Our Flow-Based Refinement Framework

Select two adjacent blocks for refinement

Build Flow Problem

Solve Flow Problem



Find feasible minimum cut

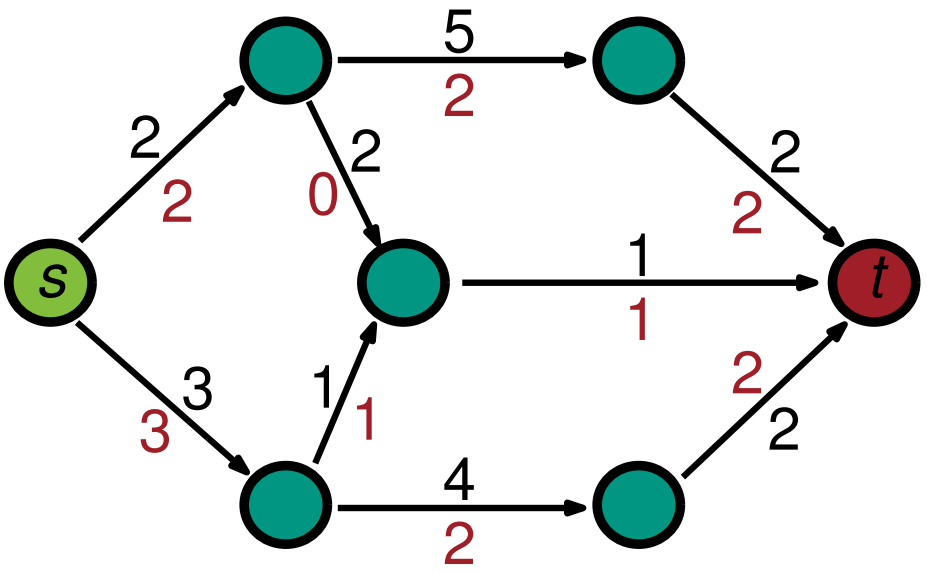
Institute of Theoretical Informatics
Algorithmics Group

Most Balanced Minimum Cut

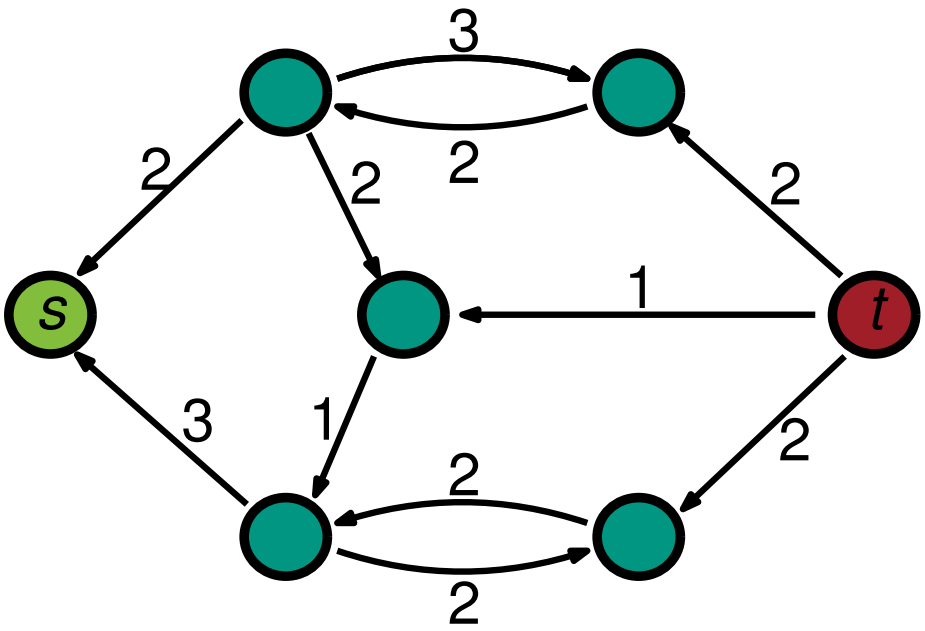
KaFFPa [Sanders, Schulz 11]

One maximum flow f has enough information to enumerate all minimum (s, t)-cuts [Picard, Queyranne 80]

Flow Graph



Residual Graph

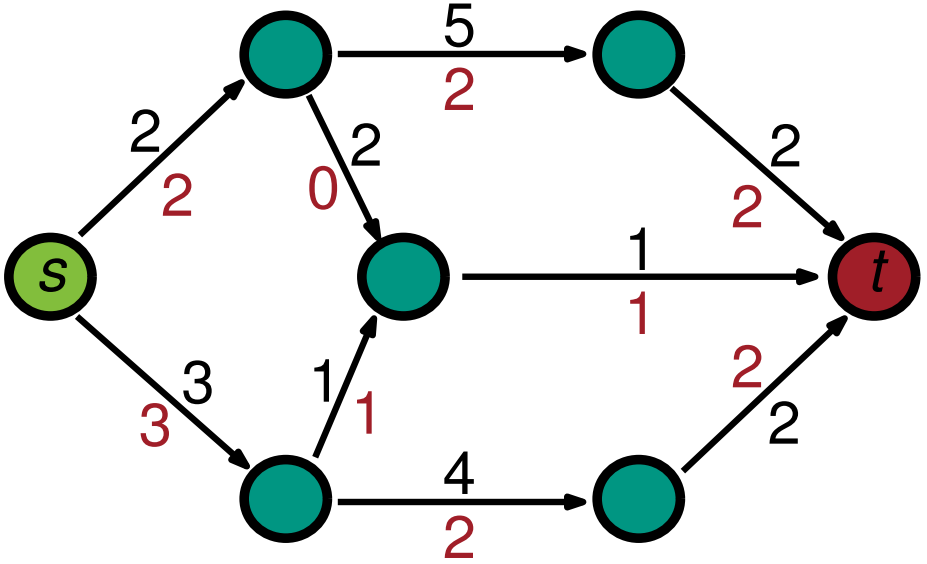


Most Balanced Minimum Cut

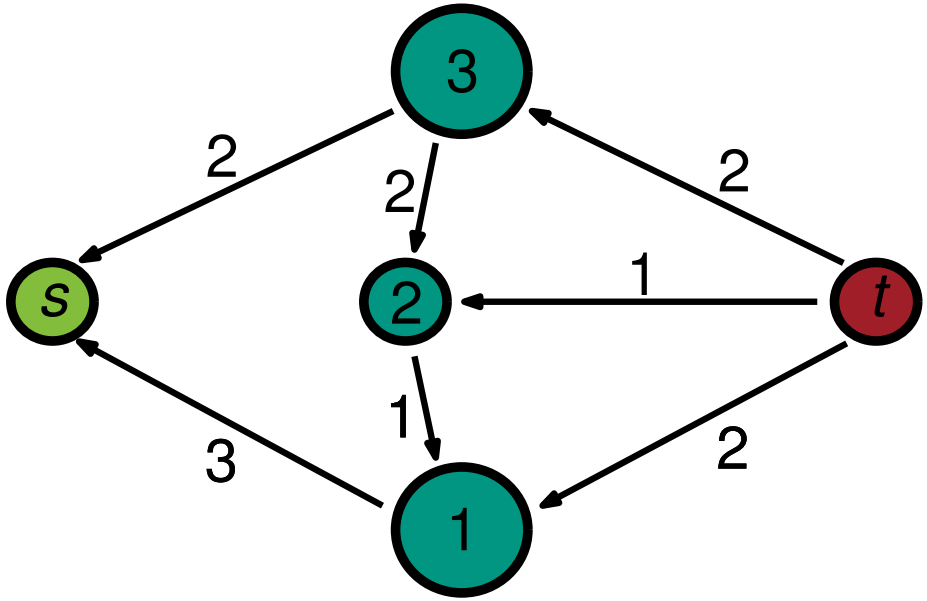
KaFFPa [Sanders, Schulz 11]

One maximum flow f has enough information to enumerate all minimum (s, t)-cuts [Picard, Queyranne 80]

Flow Graph



Picard-Queryanne DAC



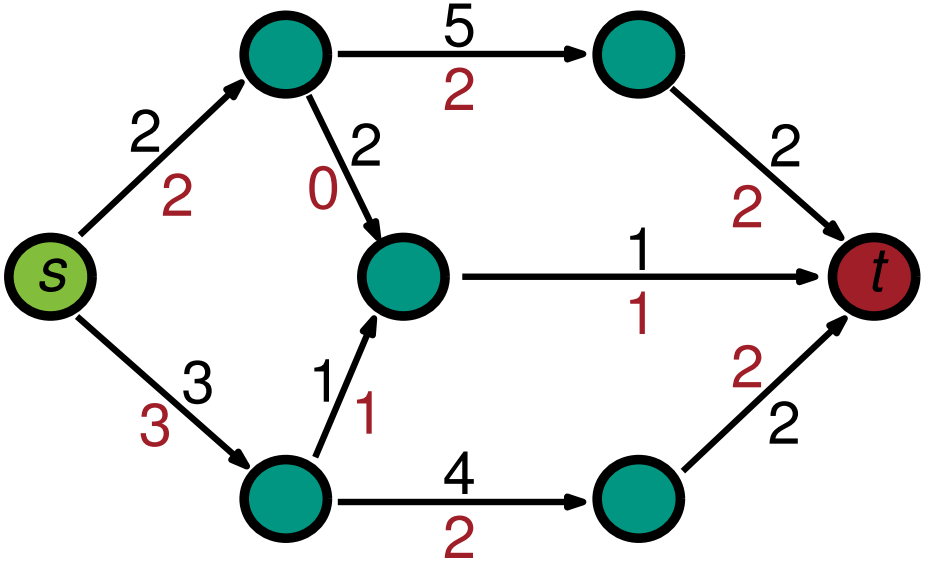
Contract all *strongly connected components* in the residual graph

Most Balanced Minimum Cut

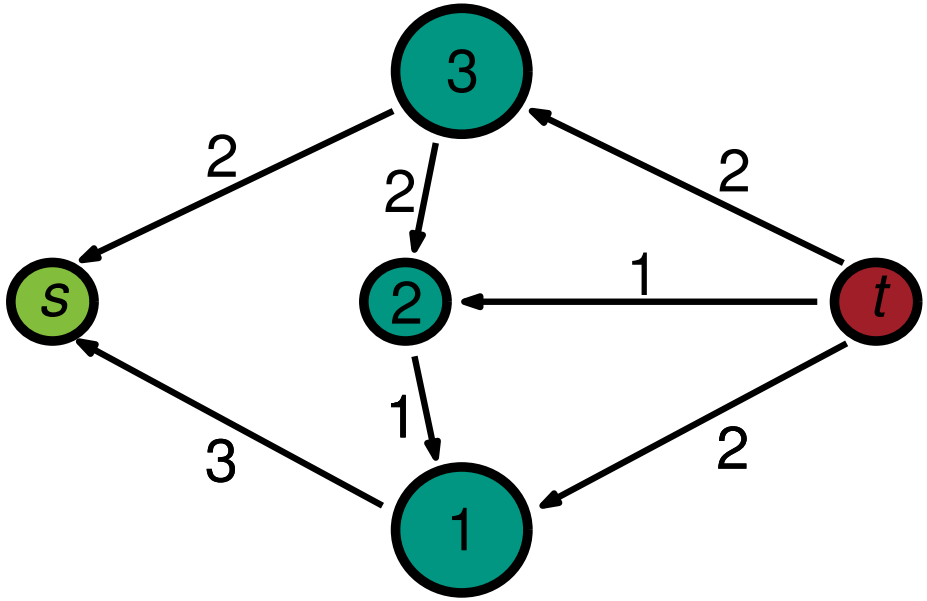
KaFFPa [Sanders, Schulz 11]

One maximum flow f has enough information to enumerate all minimum (s, t)-cuts [Picard, Queyranne 80]

Flow Graph



Picard-Queryanne DAC



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

Find *topological order*

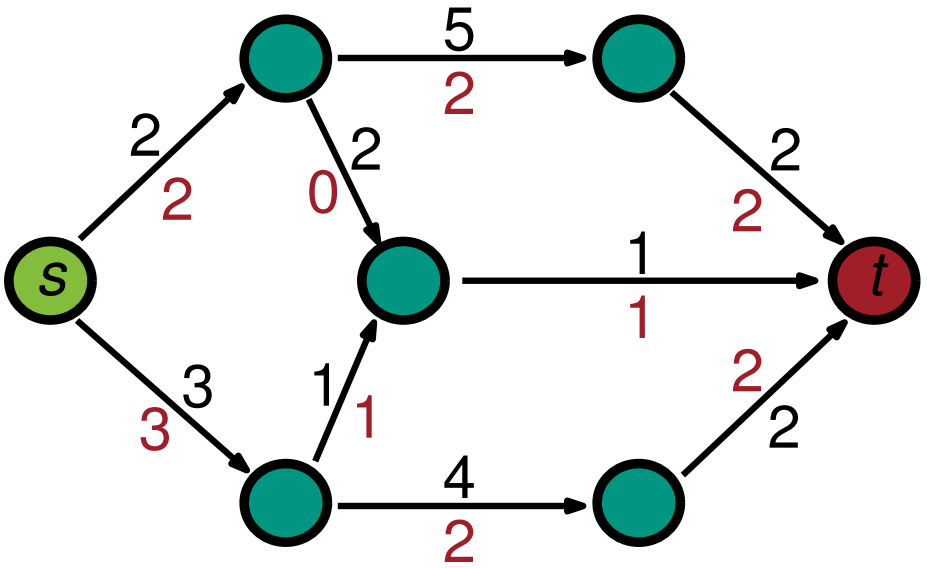
Institute of Theoretical Informatics
Algorithmics Group

Most Balanced Minimum Cut

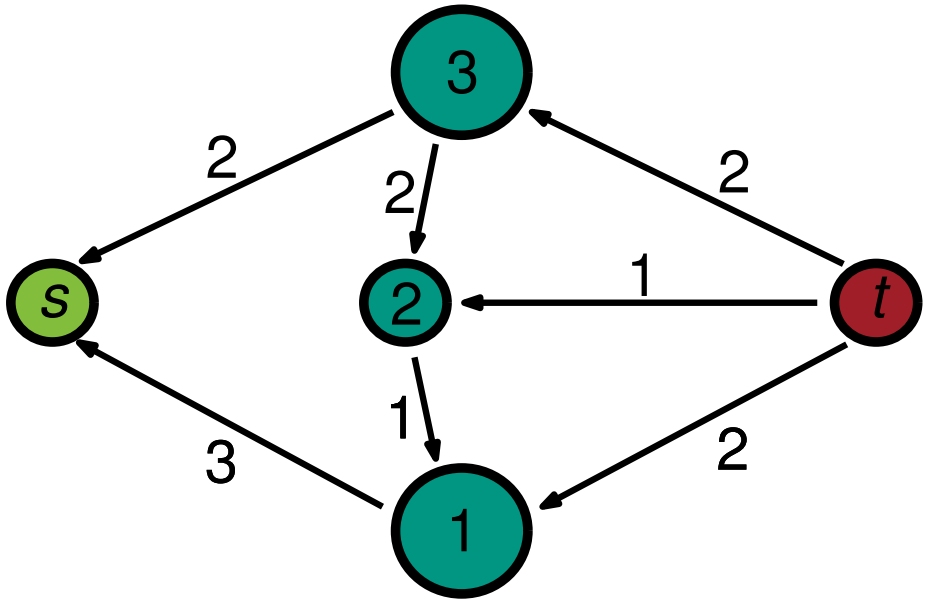
KaFFPa [Sanders, Schulz 11]

One maximum flow f has enough information to enumerate all minimum (s, t)-cuts [Picard, Queyranne 80]

Flow Graph



Picard-Queryanne DAC



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

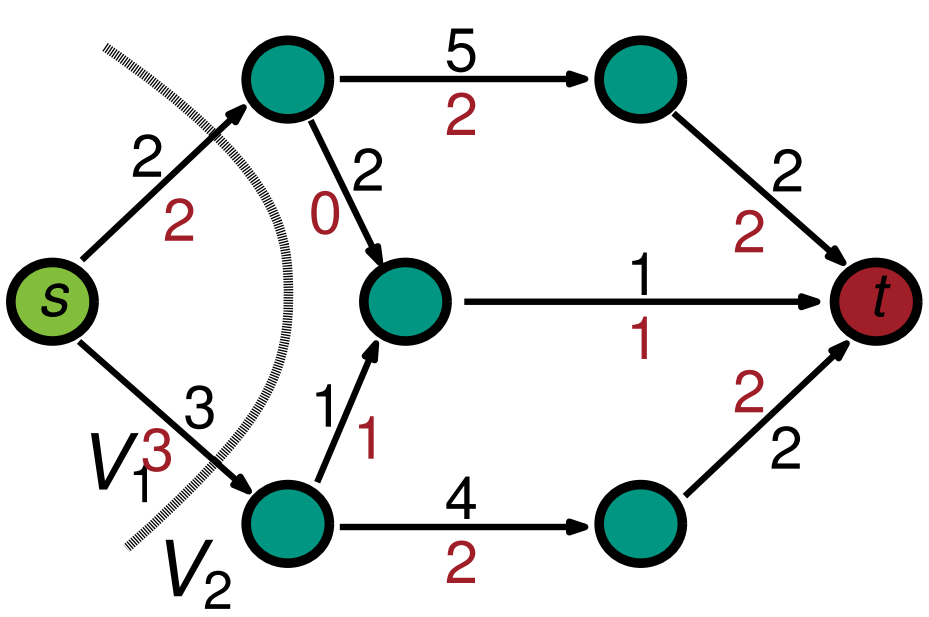
Sweep through **reverse** topological order

Most Balanced Minimum Cut

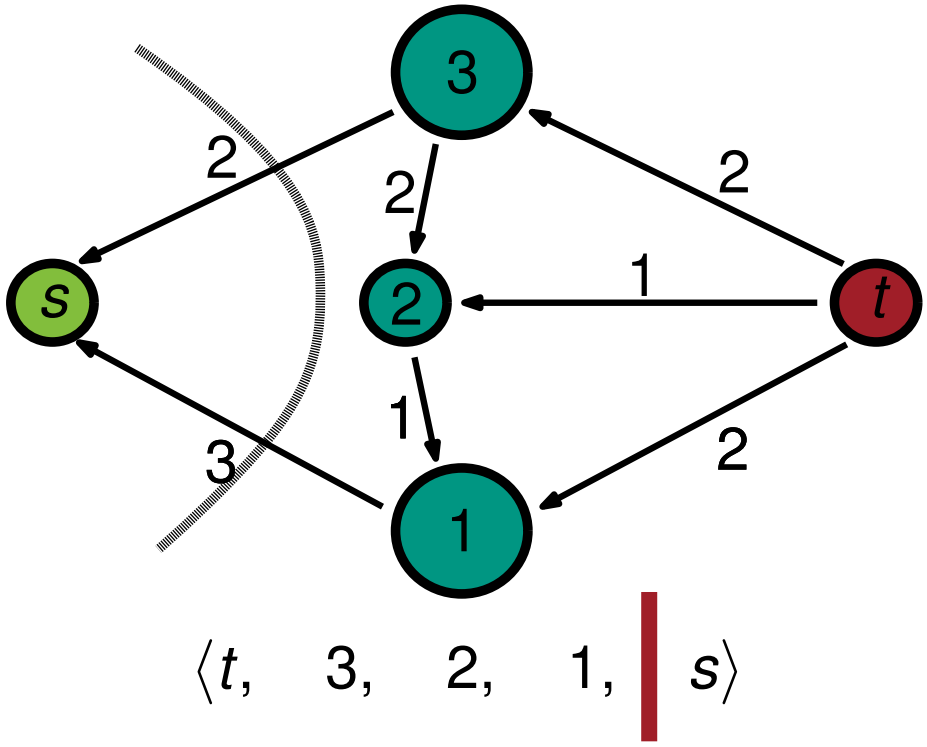
KaFFPa [Sanders, Schulz 11]

One maximum flow f has enough information to enumerate all minimum (s, t)-cuts [Picard, Queyranne 80]

Flow Graph



Picard-Queryanne DAC



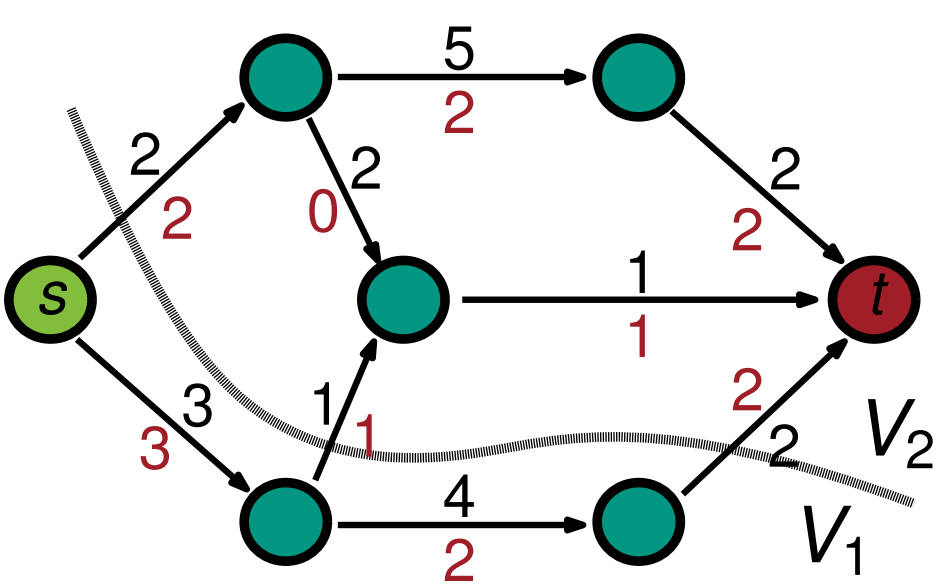
Sweep through **reverse** topological order

Most Balanced Minimum Cut

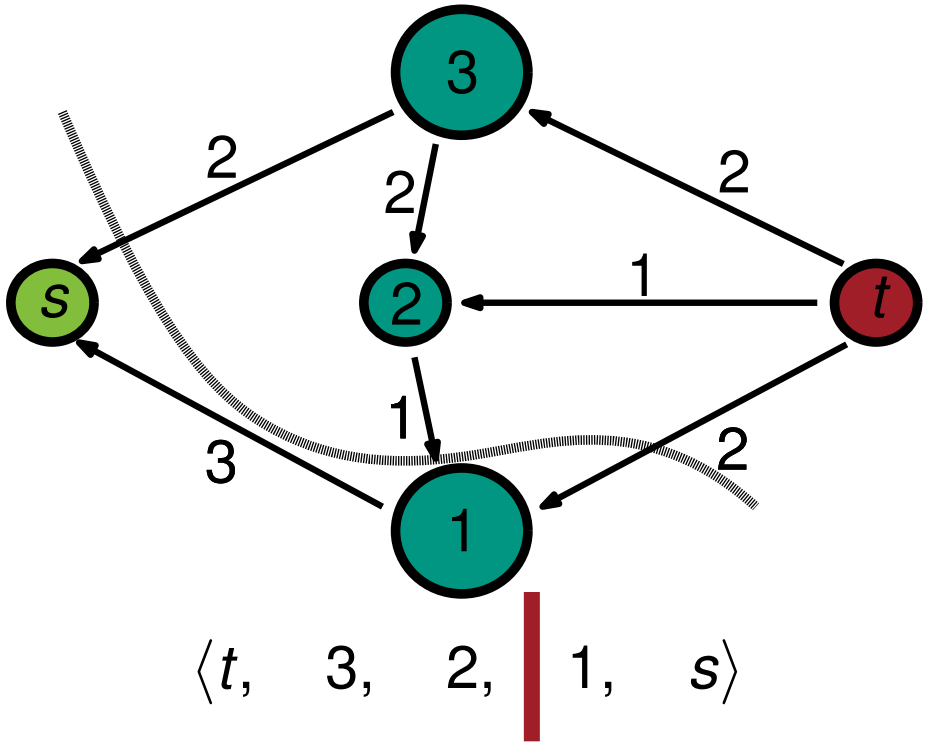
KaFFPa [Sanders, Schulz 11]

One maximum flow f has enough information to enumerate all minimum (s, t)-cuts [Picard, Queyranne 80]

Flow Graph



Picard-Queryanne DAC



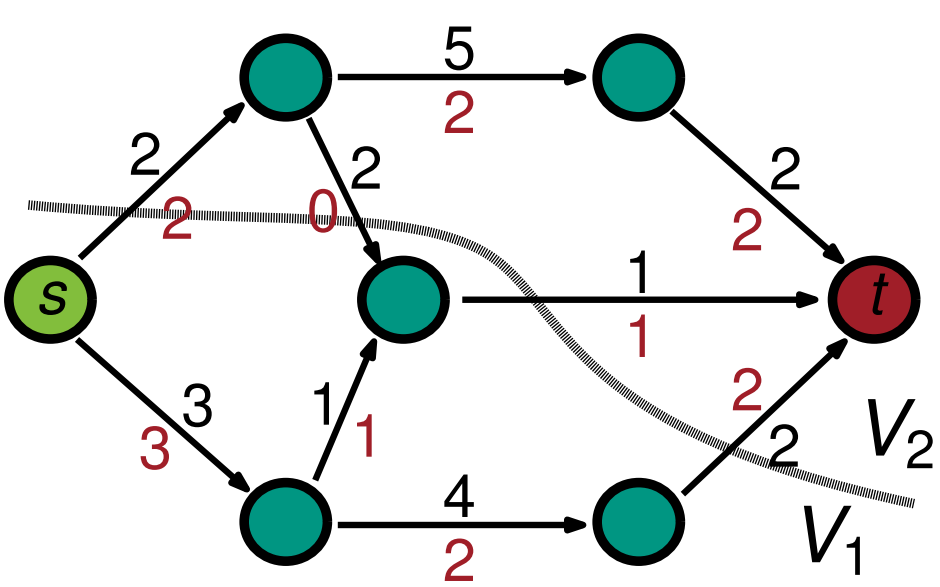
Sweep through **reverse** topological order

Most Balanced Minimum Cut

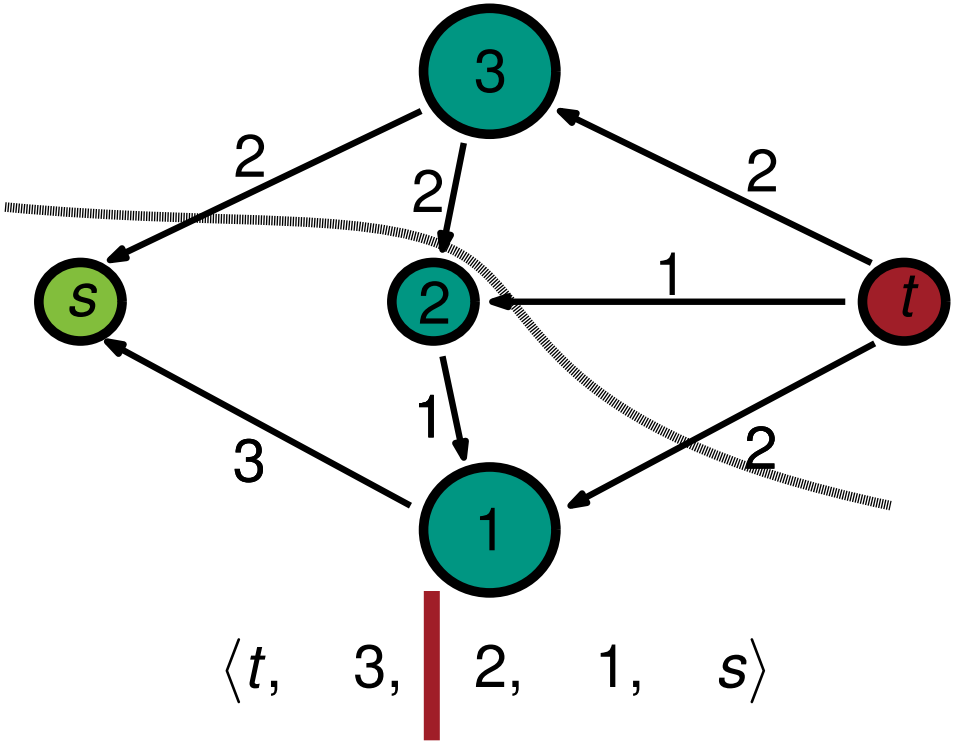
KaFFPa [Sanders, Schulz 11]

One maximum flow f has enough information to enumerate all minimum (s, t)-cuts [Picard, Queyranne 80]

Flow Graph



Picard-Queryanne DAC



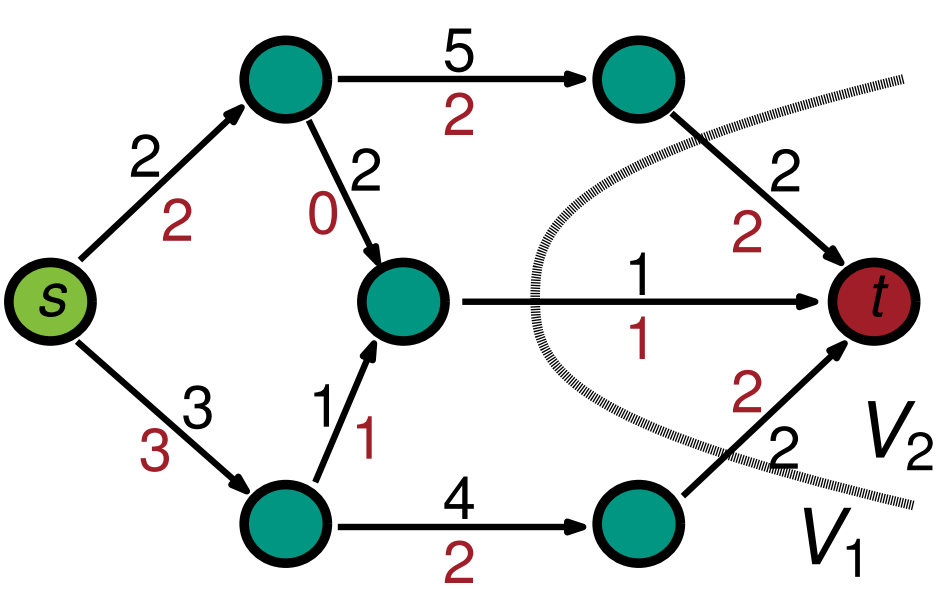
Sweep through **reverse** topological order

Most Balanced Minimum Cut

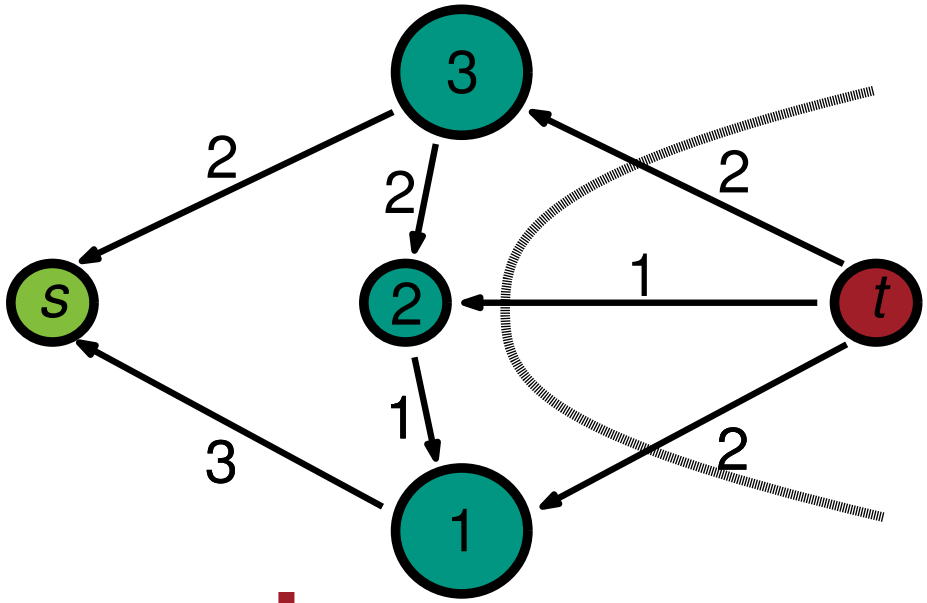
KaFFPa [Sanders, Schulz 11]

One maximum flow f has enough information to enumerate all minimum (s, t)-cuts [Picard, Queyranne 80]

Flow Graph



Picard-Queryanne DAC



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

Sweep through **reverse** topological order

Integration into KaHyPar

- KaHyPar is a *n*-level hypergraph partitioner

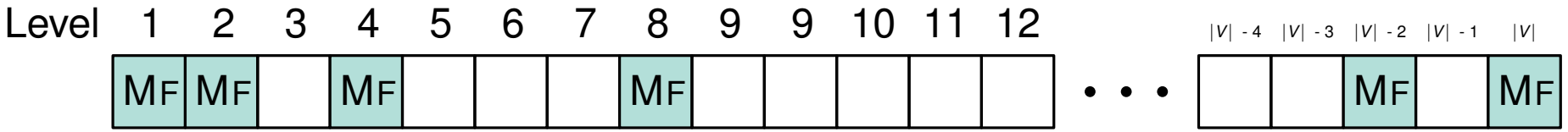
Institute of Theoretical Informatics
Algorithmics Group

Integration into KaHyPar

- KaHyPar is a *n*-level hypergraph partitioner

Flow Execution Policy

Exponential: Execute *Max-Flow-Min-Cut* computations (MF) on each level i with $i = 2^j$

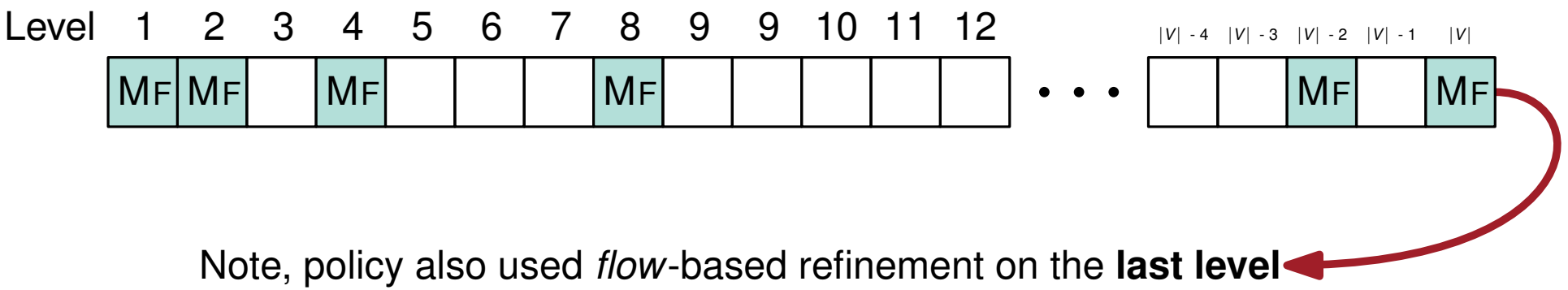


Integration into KaHyPar

- KaHyPar is a *n*-level hypergraph partitioner

Flow Execution Policy

Exponential: Execute *Max-Flow-Min-Cut* computations (MF) on each level i with $i = 2^j$



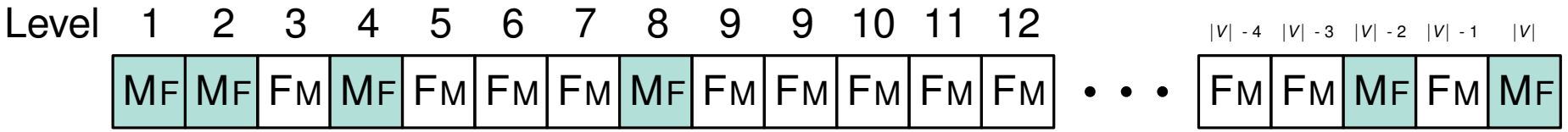
Note, policy also used *flow*-based refinement on the **last level**

Integration into KaHyPar

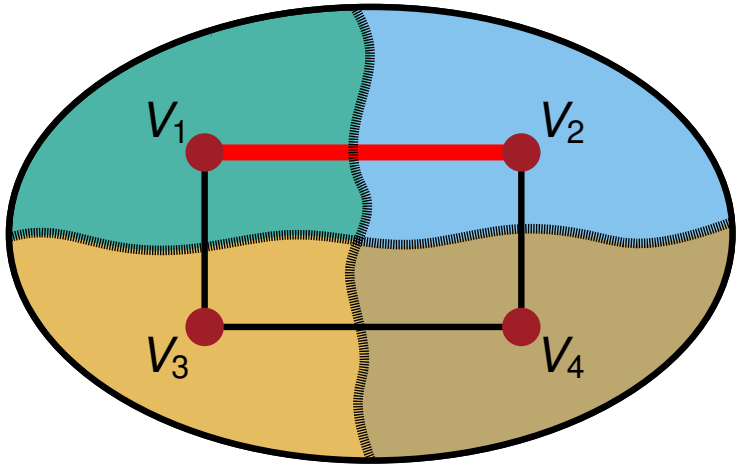
- KaHyPar is a *n*-level hypergraph partitioner

Flow Execution Policy

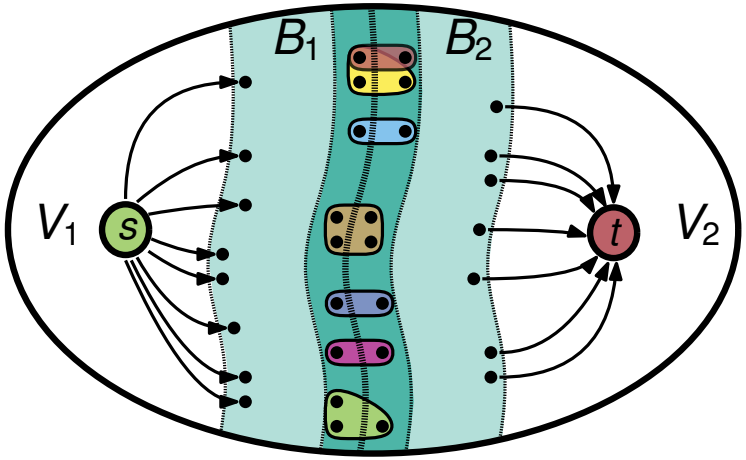
Exponential: Execute *Max-Flow-Min-Cut* computations (MF) on each level i with $i = 2^j$



Speed-Up Heuristics



Active Block Scheduling



Adaptive Flow Iterations

(R1) Skip bipartitions with small cuts (e.g. ≤ 10)

(R2) Incorporate improvement history

(R3) Break, if nothing changed

Experimental Setup

System

- Intel Xeon E5-2670 Octa-Core (2.6 GHz)
- 64 GB Main Memory
- 20 MB L3-Cache, 8 $\times$ 256 KB L2-Cache

Flow Algorithms

- EDMONDKARP
- GOLDBERGTARJAN
- BOYKOVKOLMOGOROV
- IBFS (fastest in our experiments)

Experimental Setup

System

- Intel Xeon E5-2670 Octa-Core (2.6 GHz)
- 64 GB Main Memory
- 20 MB L3-Cache, 8 $\times$ 256 KB L2-Cache

Flow Algorithms

- EDMONDKARP
- GOLDBERGTARJAN

Own Implementations

- BOYKOVKOLMOGOROV
- IBFS (fastest in our experiments)

Third-Party Implementations

Experimental Setup

Benchmarks

- Parameter Tuning Benchmark Set (25 Hypergraphs)
- Benchmark Subset (165 Hypergraphs)
- Full Benchmark Set (488 Hypergraphs)

Benchmark Type

- DAC (10 Hypergraphs)
- ISPD98 (18 Hypergraphs)
- PRIMAL (92 Hypergraphs)
- LITERAL (92 Hypergraphs)
- DUAL (92 Hypergraphs)
- SPM (184 Hypergraphs)

Experimental Setup

Benchmarks

- Parameter Tuning Benchmark Set (25 Hypergraphs)
- Benchmark Subset (165 Hypergraphs)
- Full Benchmark Set (488 Hypergraphs)

Benchmark Type

- DAC (10 Hypergraphs)
- ISPD98 (18 Hypergraphs)

VLSI Design

- PRIMAL (92 Hypergraphs)
- LITERAL (92 Hypergraphs)
- DUAL (92 Hypergraphs)

SAT Formulas

- SPM (184 Hypergraphs)

Sparse Matrices

Experimental Setup

Benchmarks

- Parameter Tuning Benchmark Set (25 Hypergraphs)
- Benchmark Subset (165 Hypergraphs)
- Full Benchmark Set (488 Hypergraphs)

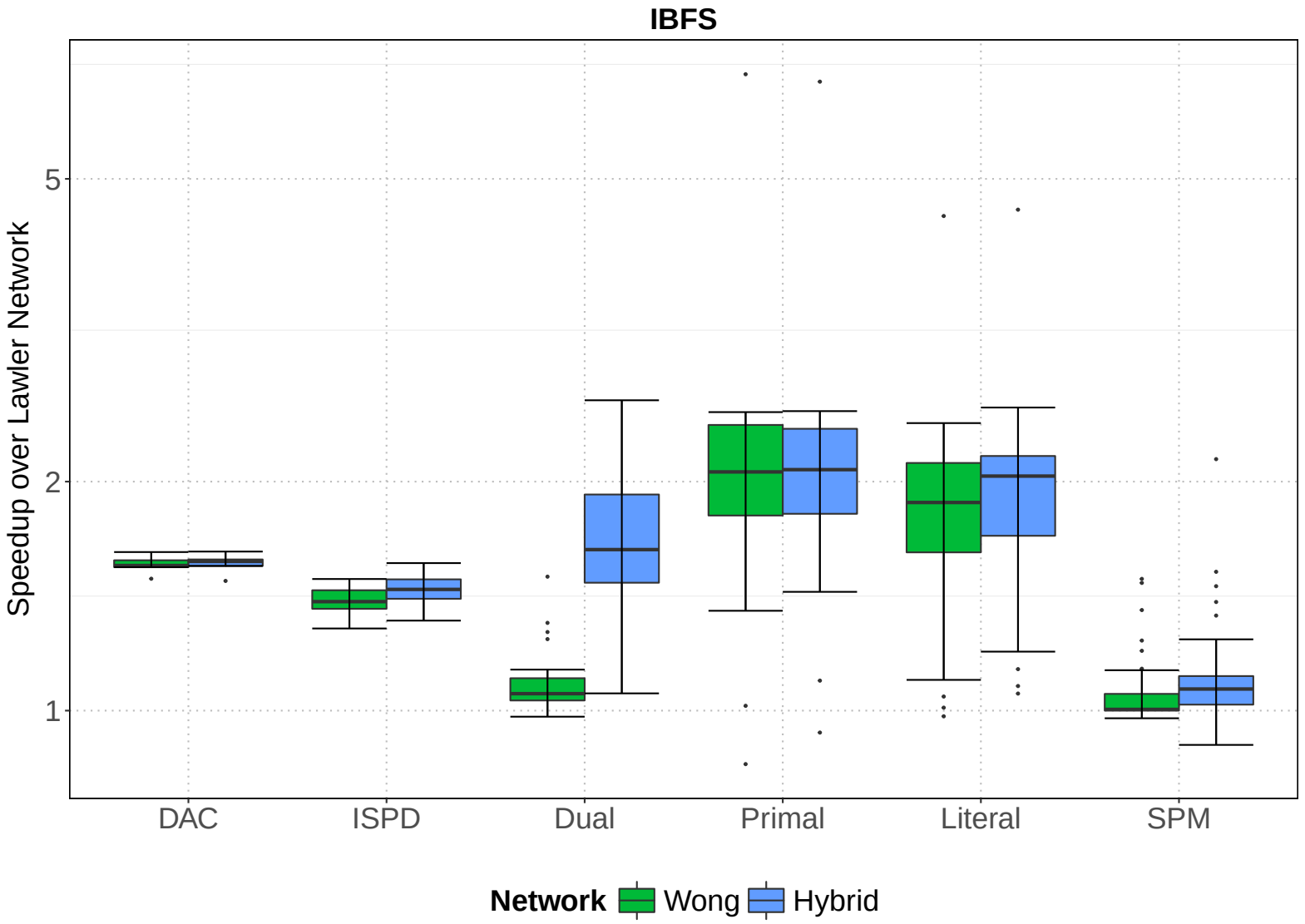
Benchmark Type

- DAC (10 Hypergraphs)
- ISPD98 (18 Hypergraphs)
- PRIMAL (92 Hypergraphs)
- LITERAL (92 Hypergraphs)
- DUAL (92 Hypergraphs)
- SPM (184 Hypergraphs)

Methodology

- $\varepsilon = 3\%$
- $k \in \{2, 4, 8, 16, 32, 64, 128\}$
- 10 seeds

Flow Networks



Flow Configuration

- +/− F = Flow-based refinement
- +/− M = Most Balanced Minimum Cut
- +/− FM = FM Heuristic

Config. α'	(+F,-M,-FM) Avg [%]	$t[s]$	(+F,+M,-FM) Avg [%]	$t[s]$	(+F,+M,+FM) Avg [%]	$t[s]$
1	−6.09	12.91	−5.60	13.40	0.23	15.16
2	−3.19	15.75	−2.07	16.74	0.73	17.51
4	−1.82	20.37	−0.19	21.88	1.21	21.53
8	−0.85	28.49	0.98	30.67	1.71	28.68
16	−0.19	43.32	1.75	46.66	2.21	41.31
Ref.	(-F,-M,+FM)		6373.88	13.73		

Flow Configuration

- +/− F = Flow-based refinement
- +/− M = Most Balanced Minimum Cut
- +/− FM = FM Heuristic

Config. α'	(+F,-M,-FM) Avg [%]	$t[s]$	(+F,+M,-FM) Avg [%]	$t[s]$	(+F,+M,+FM) Avg [%]	$t[s]$
1	−6.09	12.91	−5.60	13.40	0.23	15.16
2	−3.19	15.75	−2.07	16.74	0.73	17.51
4	−1.82	20.37	−0.19	21.88	1.21	21.53
8	−0.85	28.49	0.98	30.67	1.71	28.68
16	−0.19	43.32	1.75	46.66	2.21	41.31
Ref.	(-F,-M,+FM)		6373.88	13.73		

Flow Configuration

- +/− F = Flow-based refinement
- +/− M = Most Balanced Minimum Cut
- +/− FM = FM Heuristic

Config. α'	(+F,-M,-FM) Avg [%]	$t[s]$	(+F,+M,-FM) Avg [%]	$t[s]$	(+F,+M,+FM) Avg [%]	$t[s]$
1	−6.09	12.91	−5.60	13.40	0.23	15.16
2	−3.19	15.75	−2.07	16.74	0.73	17.51
4	−1.82	20.37	−0.19	21.88	1.21	21.53
8	−0.85	28.49	0.98	30.67	1.71	28.68
16	−0.19	43.32	1.75	46.66	2.21	41.31
Ref.	(-F,-M,+FM)		6373.88	13.73		

Institute of Theoretical Informatics
Algorithmics Group

Flow Configuration

- +/− F = Flow-based refinement
- +/− M = Most Balanced Minimum Cut
- +/− FM = FM Heuristic

Config.	(+F,-M,-FM)		(+F,+M,-FM)		(+F,+M,+FM)	
α'	Avg [%]	$t[s]$	Avg [%]	$t[s]$	Avg [%]	$t[s]$
1	−6.09	12.91	−5.60	13.40	0.23	15.16
2	−3.19	15.75	−2.07	16.74	0.73	17.51
4	−1.82	20.37	−0.19	21.88	1.21	21.53
8	−0.85	28.49	0.98	30.67	1.71	28.68
16	−0.19	43.32	1.75	46.66	2.21	41.31
Ref.	(-F,-M,+FM)		6373.88	13.73		

Speed-Up Heuristics

Algorithm	Avg [%]	Min [%]	$t_{\text{flow}}[s]$	$t[s]$
KaHyPar-CA	7077.20	6820.17	-	29.26
KaHyPar-MF	2.47	2.12	43.04	72.30
KaHyPar-MF$_{(R1)}$	2.41	2.06	33.89	63.15
KaHyPar-MF$_{(R1,R2)}$	2.40	2.05	28.52	57.78
KaHyPar-MF$_{(R1,R2,R3)}$	2.41	2.06	21.23	50.49

Speed-Up Heuristics

Algorithm	Avg [%]	Min [%]	$t_{flow}[s]$	$t[s]$
KaHyPar-CA	7077.20	6820.17	-	29.26
KaHyPar-MF	2.47	2.12	43.04	72.30
KaHyPar-MF$_{(R1)}$	2.41	2.06	33.89	63.15
KaHyPar-MF$_{(R1,R2)}$	2.40	2.05	28.52	57.78
KaHyPar-MF$_{(R1,R2,R3)}$	2.41	2.06	21.23	50.49

Comparable Quality

Institute of Theoretical Informatics
Algorithmics Group

Speed-Up Heuristics

Algorithm	Avg [%]	Min [%]	$t_{\text{flow}}[s]$	$t[s]$
KaHyPar-CA	7077.20	6820.17	-	29.26
KaHyPar-MF	2.47	2.12	43.04	72.30
KaHyPar-MF$_{(R1)}$	2.41	2.06	33.89	63.15
KaHyPar-MF$_{(R1,R2)}$	2.40	2.05	28.52	57.78
KaHyPar-MF$_{(R1,R2,R3)}$	2.41	2.06	21.23	50.49

Speed-up by a factor of 2

Institute of Theoretical Informatics
Algorithmics Group

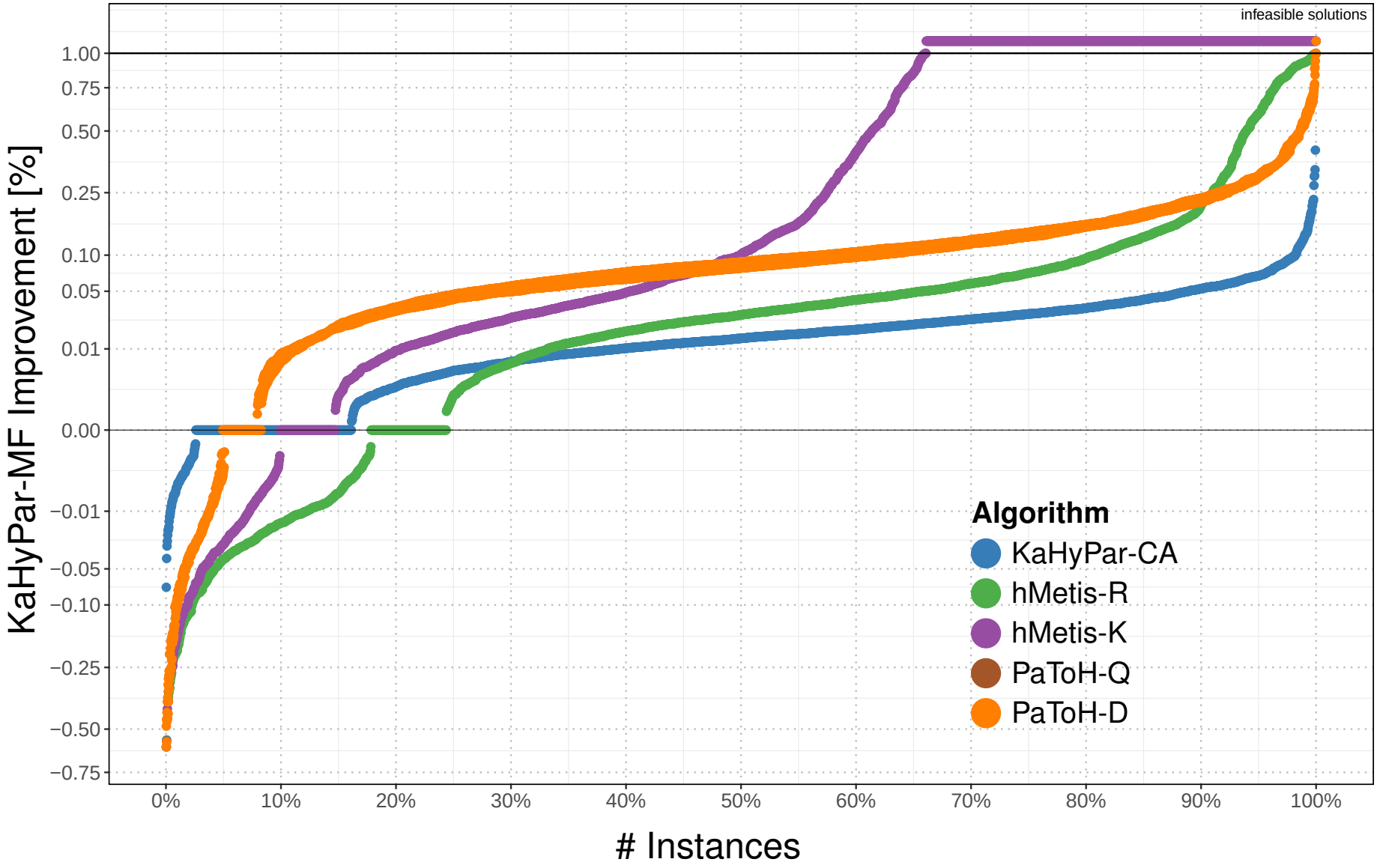
Speed-Up Heuristics

Algorithm	Avg [%]	Min [%]	$t_{\text{flow}}[s]$	$t[s]$
KaHyPar-CA	7077.20	6820.17	-	29.26
KaHyPar-MF	2.47	2.12	43.04	72.30
KaHyPar-MF$_{(R1)}$	2.41	2.06	33.89	63.15
KaHyPar-MF$_{(R1,R2)}$	2.40	2.05	28.52	57.78
KaHyPar-MF$_{(R1,R2,R3)}$	2.41	2.06	21.23	50.49

Slow-down compared to KaHyPar-CA by factor of 1.72

Institute of Theoretical Informatics
Algorithmics Group

Quality - Full Benchmark Set

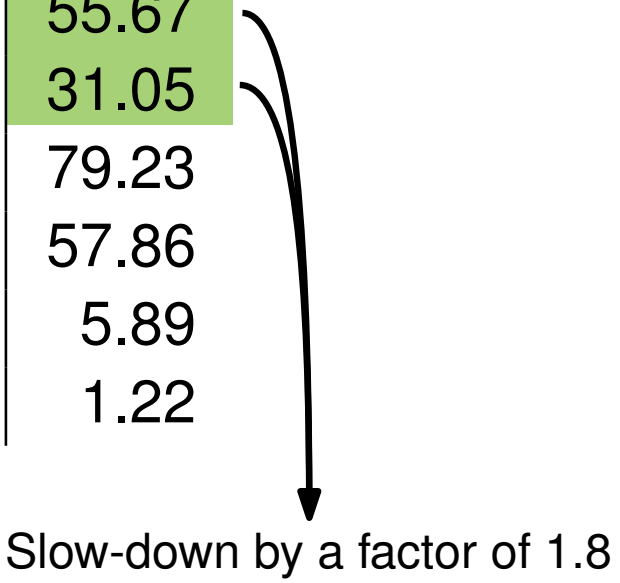


Running Time - Full Benchmark Set

Algorithm	$t[s]$
KaHyPar-MF	55.67
KaHyPar-CA	31.05
hMetis-R	79.23
hMetis-K	57.86
PaToH-Q	5.89
PaToH-D	1.22

Running Time - Full Benchmark Set

Algorithm	$t[s]$
KaHyPar-MF	55.67
KaHyPar-CA	31.05
hMetis-R	79.23
hMetis-K	57.86
PaToH-Q	5.89
PaToH-D	1.22



Running Time - Full Benchmark Set

Algorithm	$t[s]$
KaHyPar-MF	55.67
KaHyPar-CA	31.05
hMetis-R	79.23
hMetis-K	57.86
PaToH-Q	5.89
PaToH-D	1.22

Comparable running time to hMetis-K

Conclusion

Contributions

- **Generalize framework** of *KaFFPa* [Sanders, Schulz 11]
- **Sparsification techniques** for hypergraph flow networks
- Optimized **flow problem modeling** approach
- Integration of **speedup heuristics**

Conclusion

Contributions

- **Generalize framework** of *KaFFPa* [Sanders, Schulz 11]
- **Sparsification techniques** for hypergraph flow networks
- Optimized **flow problem modeling** approach
- Integration of **speedup heuristics**

Experimental Results

- Best partitions on 73% of 3222 benchmark instances
- Improve quality of *KaHyPar* by 2.5%
- Slowdown by factor of 1.8
- Comparable running time to *hMetis*

References

[Fiduccia, Mattheyses 88]
C. M. Fiduccia and R. M. Mattheyses. A Linear-Time Heuristic for improving Network Partitions. In *Papers on 25 Years of Electronic Design Automation 1988*, ACM, pp.241-247.

[Hu, Moerder 85]
T.C. Hu and K. Multiterminal Flows in a Hypergraph. In VLSI Circuit Layout: Theory and Design. In *IEEE Press 1985*, pp.1-12.

[Lawler 73]
E. Lawler. Cutsets and Partitions of Hypergraphs. *Networks 3 1973*, pp.275-285.

[Liu, Wong 98]
H. Liu and D. Wong. Network Flow Based Multi-way Partitioning with Area and Pin Constraints. In *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems 1998*, pp.50-59.

[Picard, Queyranne 80]
J.-C. Picard and M. Queyranne. On the Structure of all Minimum Cuts in a Network and Applications. In *Combinatorial Optimization II 1980*, pp.8-16.

[Sanders, Schulz 11]
P. Sanders and C. Schulz. Engineering Multilevel Graph Partitioning Algorithms. ESA 2011, vol. 6942, Springer, pp.469-480.

Appendix

Institute of Theoretical Informatics
Algorithmics Group

Flow Problem Modeling

KaFFPa vs. *KaHyPar*

	Hypergraphs			Graphs		
α	$\varepsilon = 1\%$	$\varepsilon = 3\%$	$\varepsilon = 5\%$	$\varepsilon = 1\%$	$\varepsilon = 3\%$	$\varepsilon = 5\%$
1	8.3	8.9	8.2	13.3	12.7	11.8
2	8.5	7.1	5.1	12.4	10.1	8.5
4	7.4	4.1	2.8	11.0	7.9	5.7
8	5.4	2.4	1.6	9.4	5.6	4.0
16	3.5	1.3	1.2	7.5	4.2	3.6

Flow Problem Modeling

KaFFPa vs. *KaHyPar*

Test Configuration: (+F,-M,-FM)

25 Hypergraph Instances

15 Graph Instances

	HYPERGRAPHS			GRAPHS		
α	$\varepsilon = 1\%$	$\varepsilon = 3\%$	$\varepsilon = 5\%$	$\varepsilon = 1\%$	$\varepsilon = 3\%$	$\varepsilon = 5\%$
1	8.3	8.9	8.2	13.3	12.7	11.8
2	8.5	7.1	5.1	12.4	10.1	8.5
4	7.4	4.1	2.8	11.0	7.9	5.7
8	5.4	2.4	1.6	9.4	5.6	4.0
16	3.5	1.3	1.2	7.5	4.2	3.6

Flow Problem Modeling

KaFFPa vs. *KaHyPar*

Graphs profit more than hypergraphs from new modeling approach

	HYPERGRAPHS			GRAPHS		
α	$\varepsilon = 1\%$	$\varepsilon = 3\%$	$\varepsilon = 5\%$	$\varepsilon = 1\%$	$\varepsilon = 3\%$	$\varepsilon = 5\%$
1	8.3	8.9	8.2	13.3	12.7	11.8
2	8.5	7.1	5.1	12.4	10.1	8.5
4	7.4	4.1	2.8	11.0	7.9	5.7
8	5.4	2.4	1.6	9.4	5.6	4.0
16	3.5	1.3	1.2	7.5	4.2	3.6

Flow Problem Modeling

KaFFPa vs. *KaHyPar*

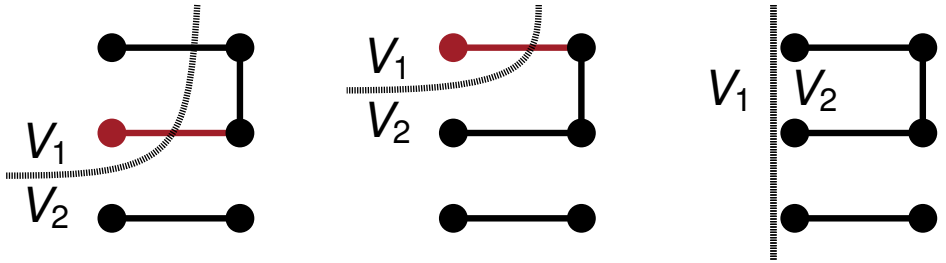
The smaller the flow problem the bigger the improvement

	HYPERGRAPHS			GRAPHS		
α	$\varepsilon = 1\%$	$\varepsilon = 3\%$	$\varepsilon = 5\%$	$\varepsilon = 1\%$	$\varepsilon = 3\%$	$\varepsilon = 5\%$
1	8.3	8.9	8.2	13.3	12.7	11.8
2	8.5	7.1	5.1	12.4	10.1	8.5
4	7.4	4.1	2.8	11.0	7.9	5.7
8	5.4	2.4	1.6	9.4	5.6	4.0
16	3.5	1.3	1.2	7.5	4.2	3.6

FM Algorithm - Disadvantages

- Incorporate only **local** informations about problem structure

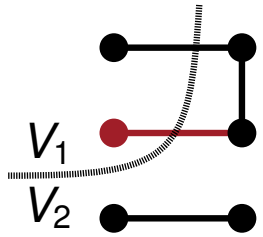
FM Algorithm

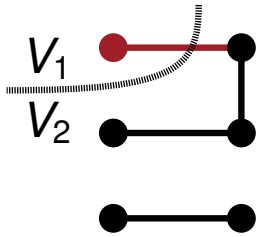


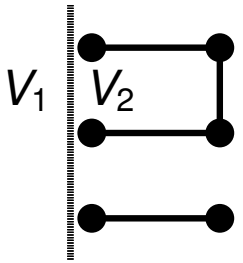
FM Algorithm - Disadvantages

- Incorporate only **local** informations about problem structure

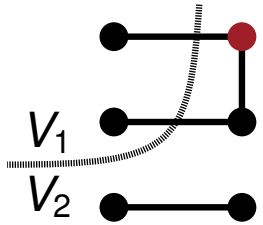
FM Algorithm

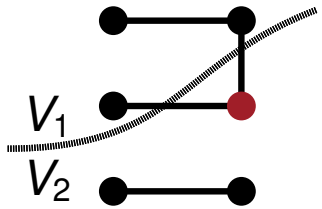


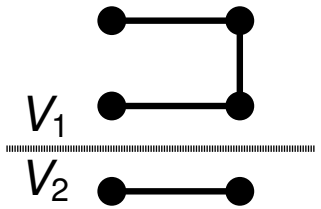




Optimal

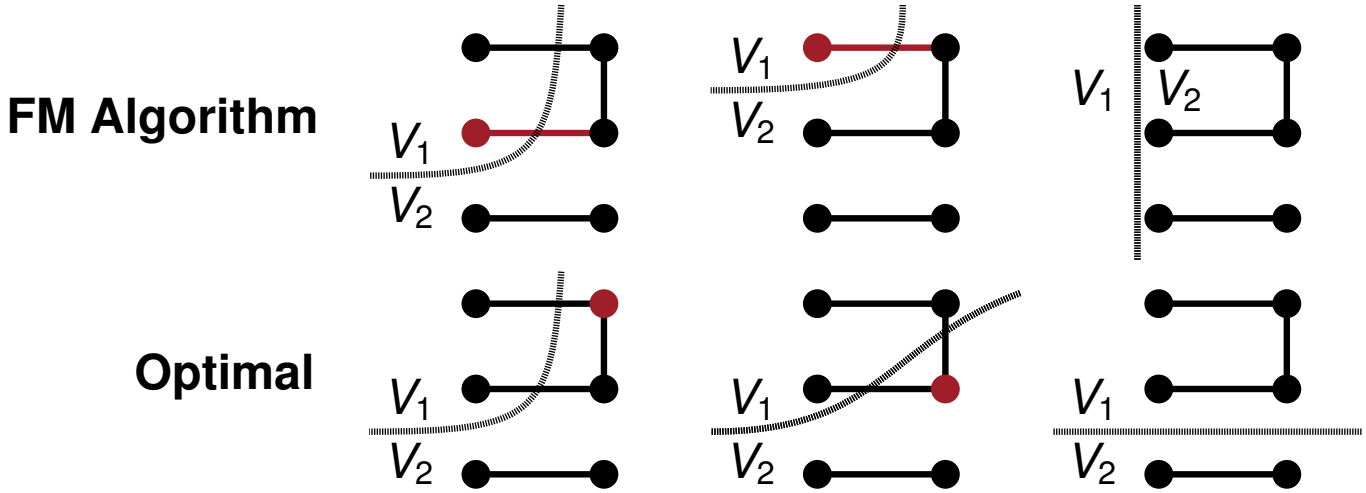




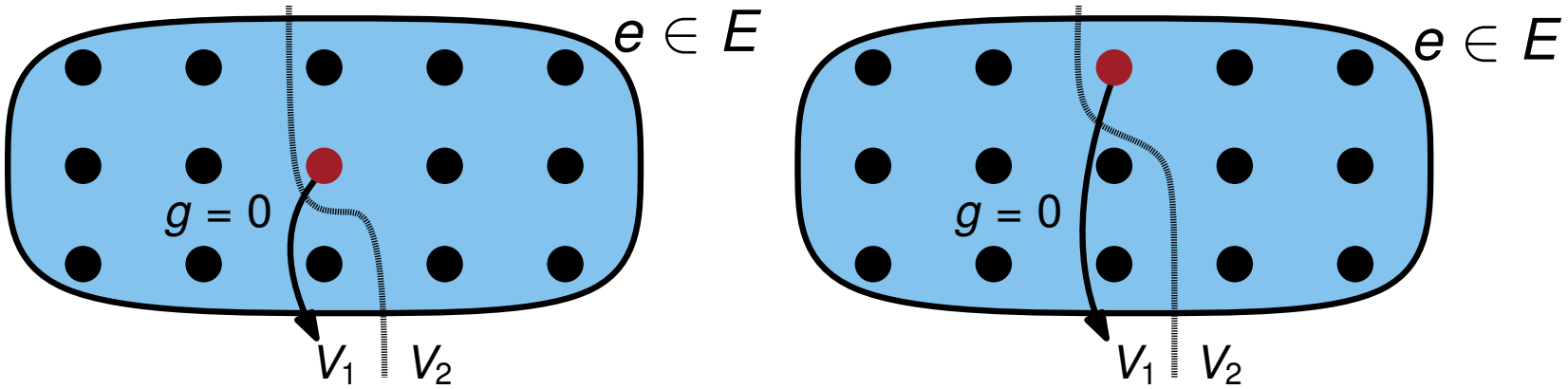


FM Algorithm - Disadvantages

- Incorporate only **local** informations about problem structure

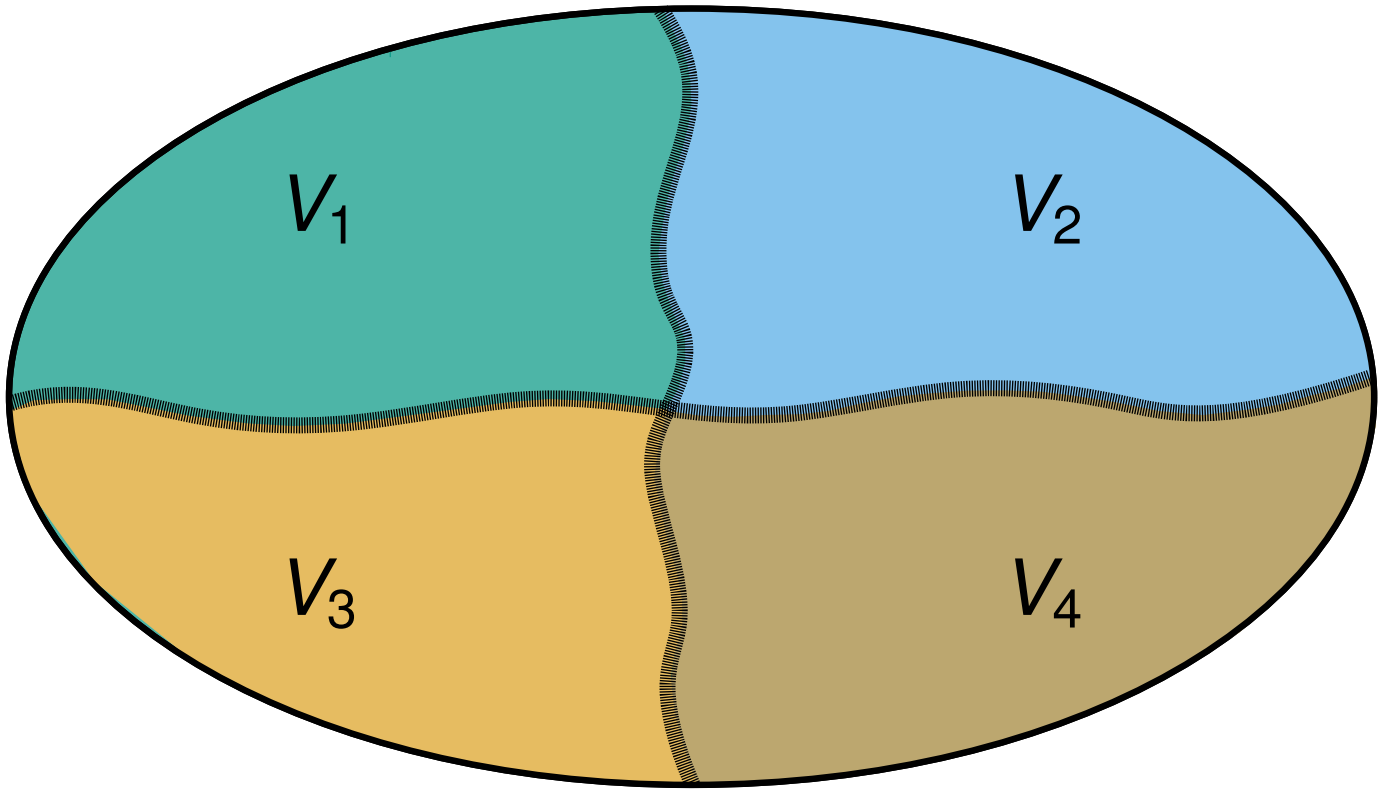


- **Zero-Gain** Moves



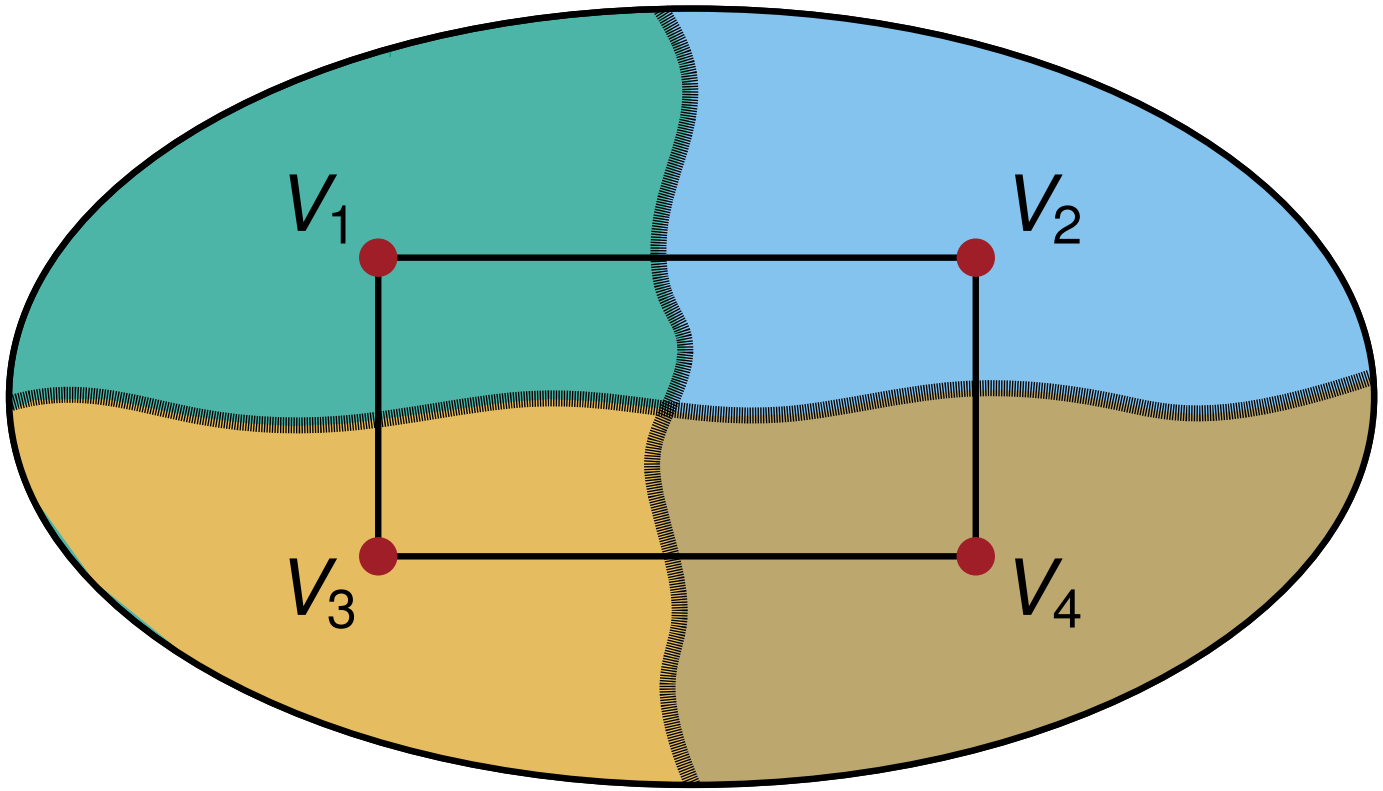
Active Block Scheduling

KaFFPa [Sanders, Schulz 11]



Active Block Scheduling

KaFFPa [Sanders, Schulz 11]



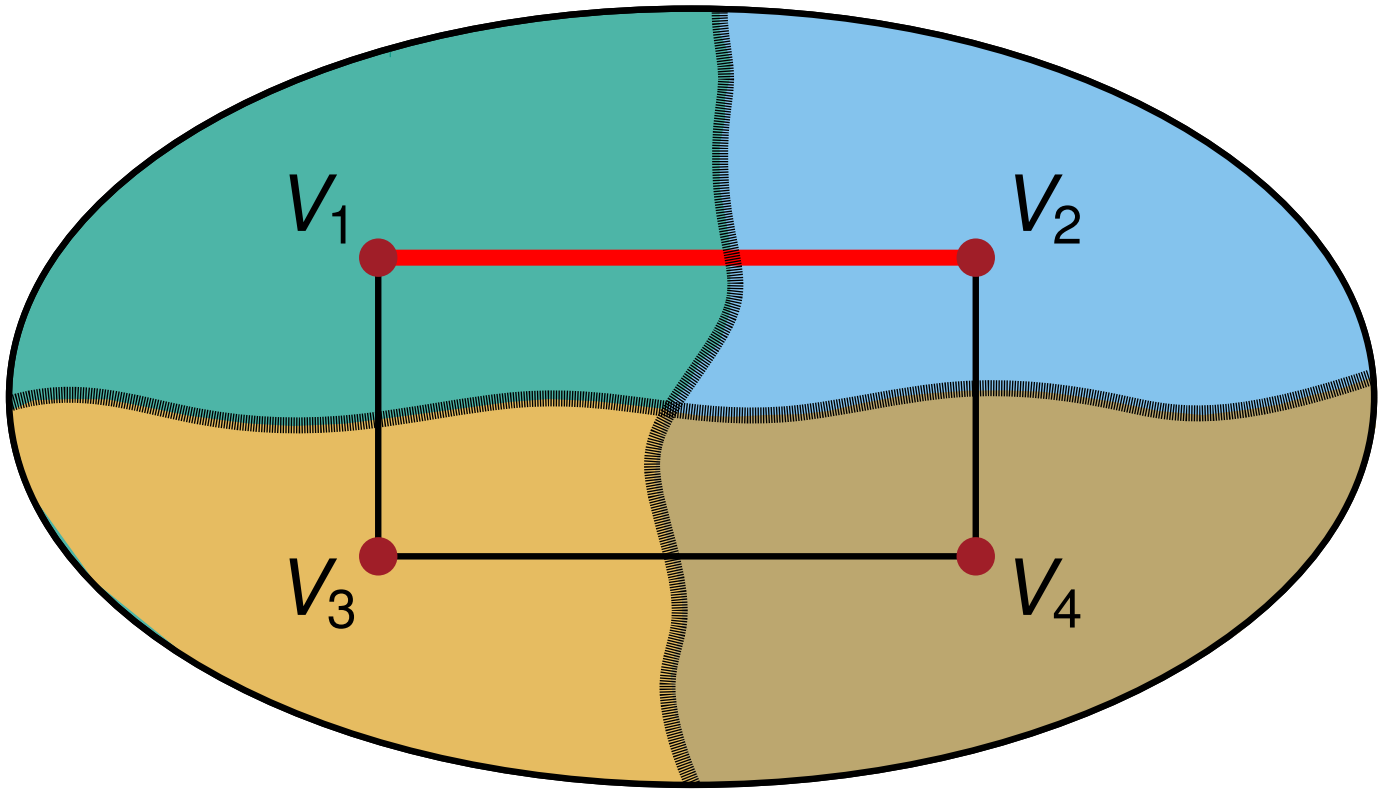
Build Quotient Graph

Active Block Scheduling

KaFFPa [Sanders, Schulz 11]

Round 1

refine(V_1, V_2) = **Improvement!**



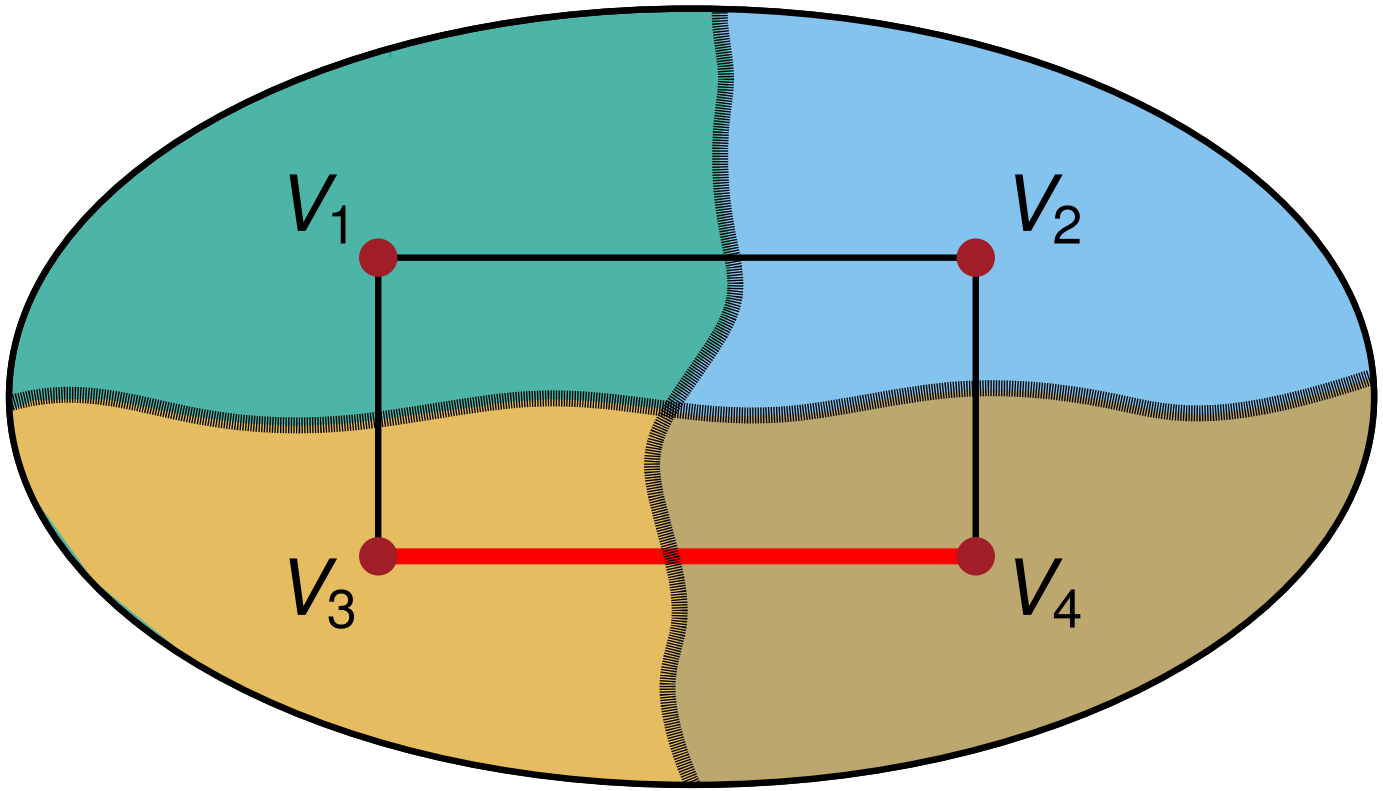
Using 2-way refinement algorithm for **active** blocks of the quotient graph

Active Block Scheduling

KaFFPa [Sanders, Schulz 11]

Round 1

refine(V_3, V_4) = **No Improvement!**



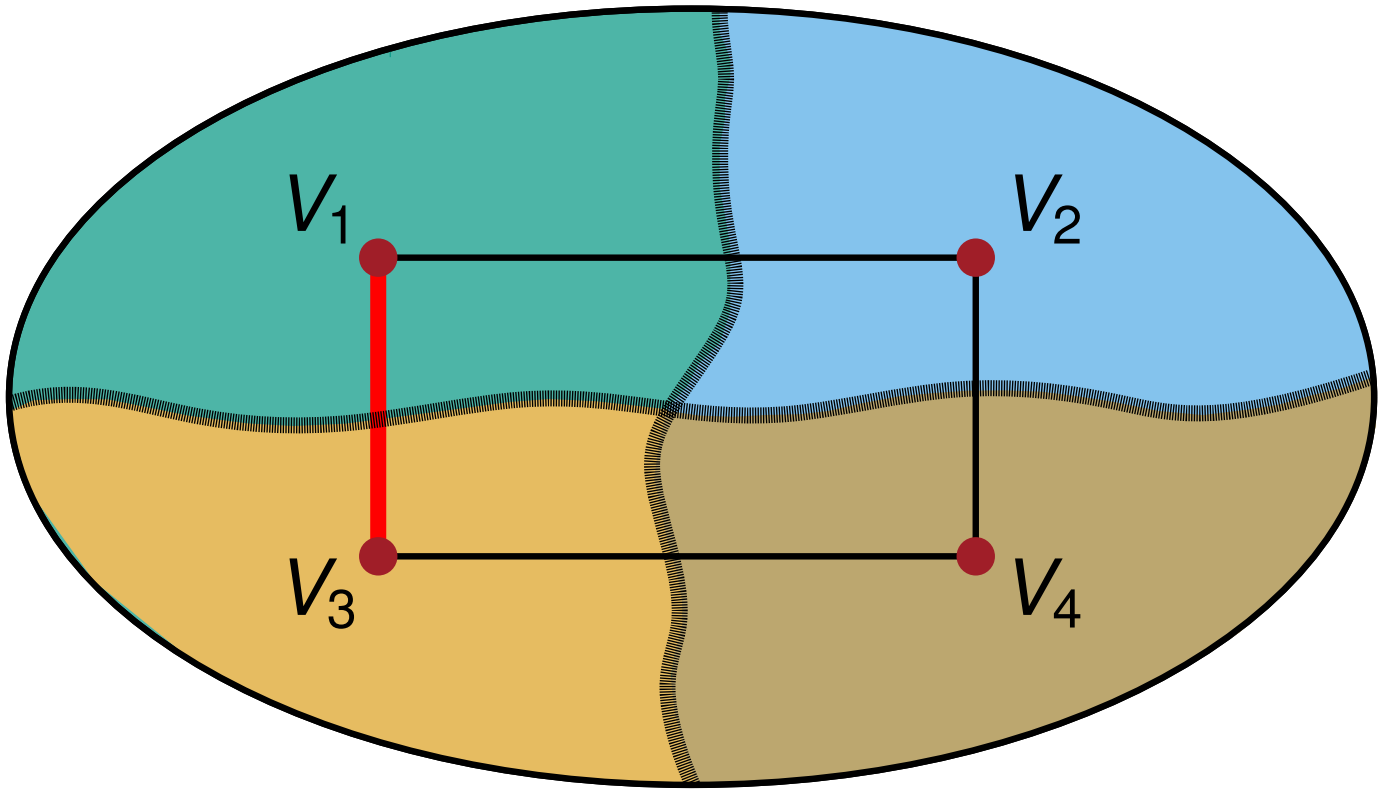
Using 2-way refinement algorithm for **active** blocks of the quotient graph

Active Block Scheduling

KaFFPa [Sanders, Schulz 11]

Round 1

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



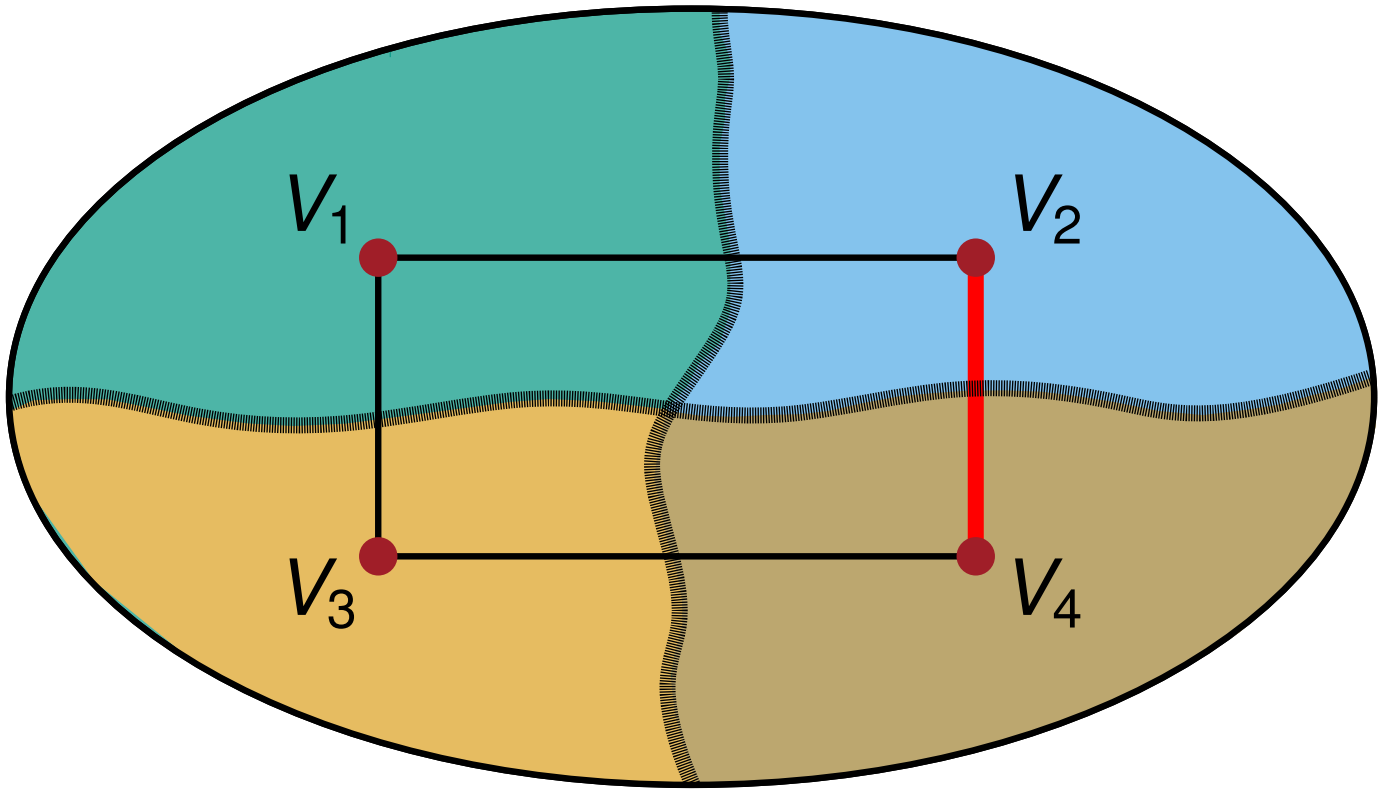
Using 2-way refinement algorithm for **active** blocks of the quotient graph

Active Block Scheduling

KaFFPa [Sanders, Schulz 11]

Round 1

refine(V_2, V_4) = **No Improvement!**

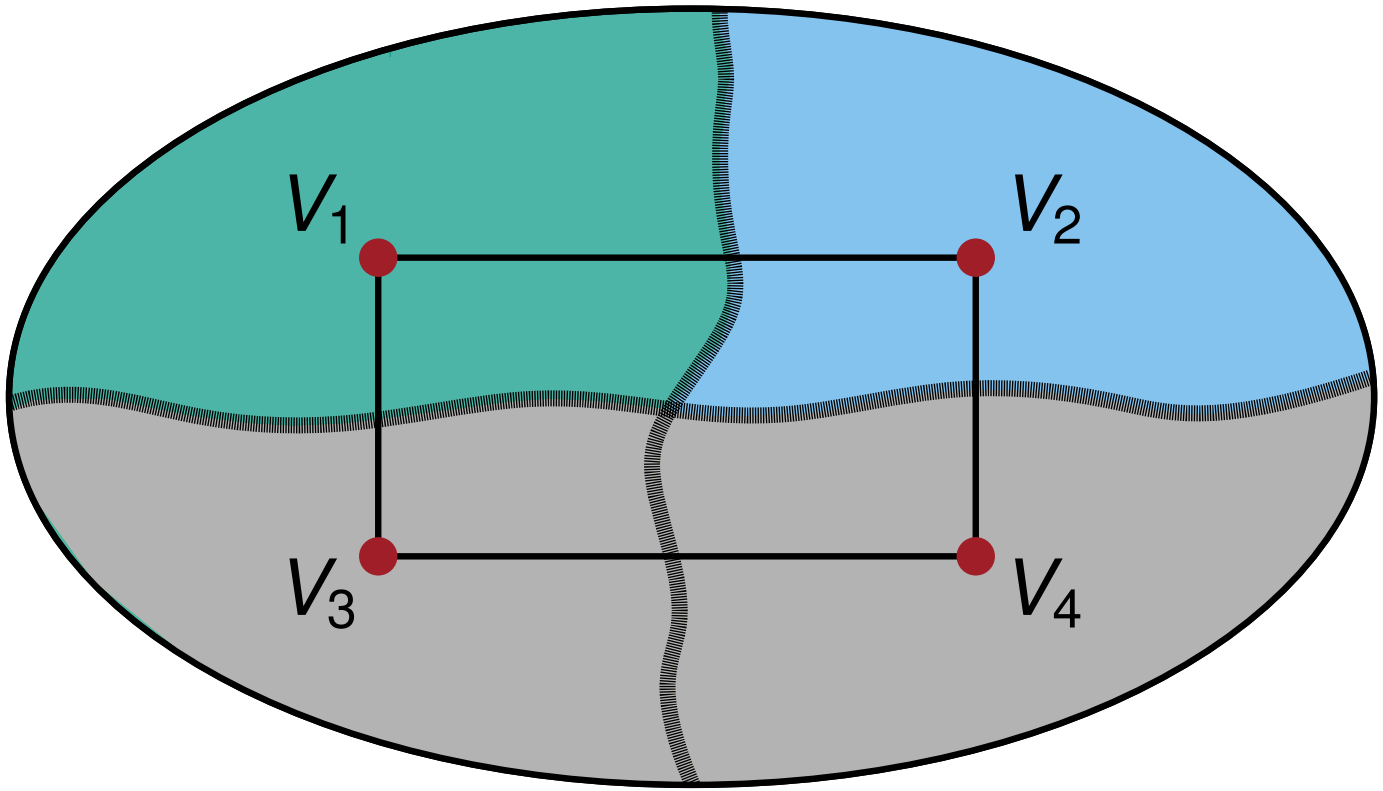


Using 2-way refinement algorithm for **active** blocks of the quotient graph

Active Block Scheduling

KaFFPa [Sanders, Schulz 11]

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



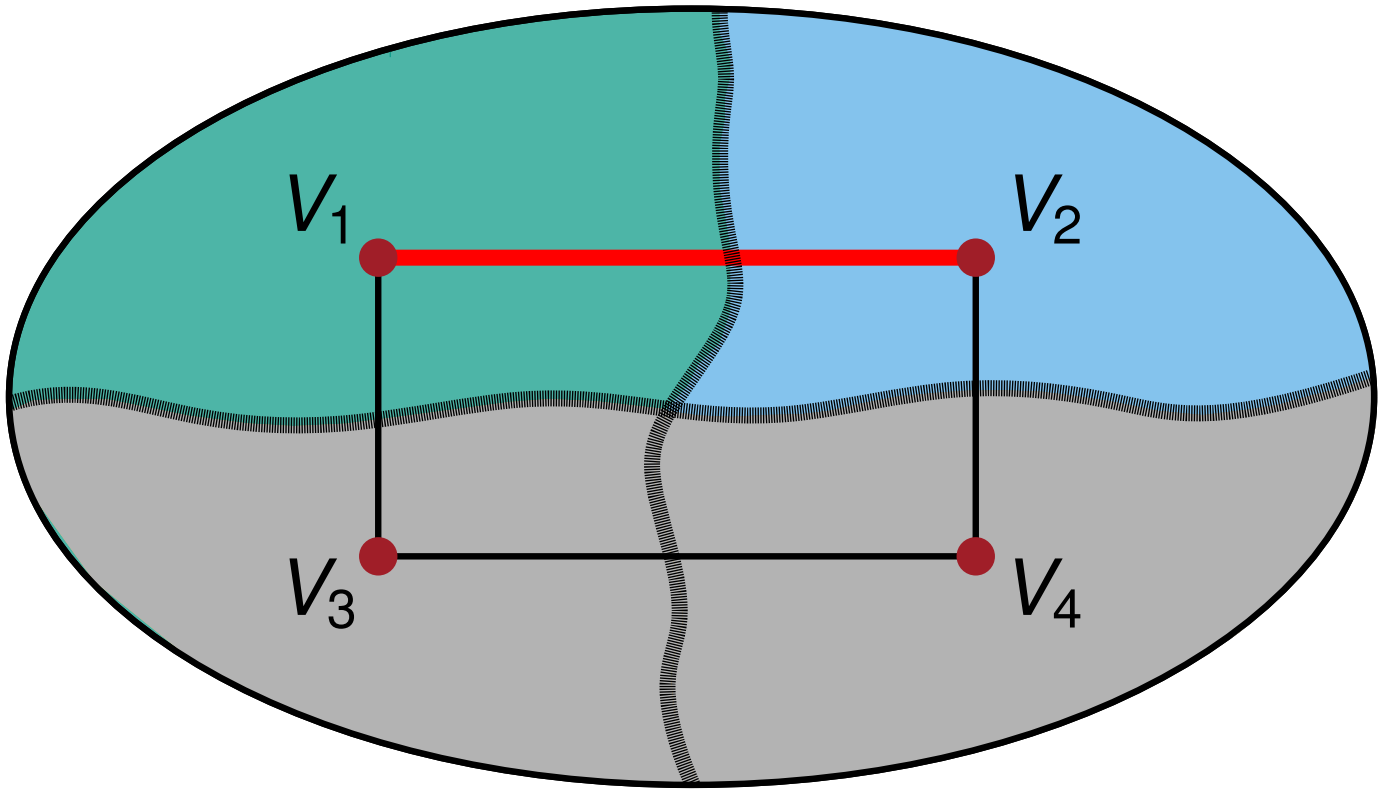
Using 2-way refinement algorithm for **active** blocks of the quotient graph

Active Block Scheduling

KaFFPa [Sanders, Schulz 11]

Round 2

refine(V_1, V_2) = **No Improvement!**



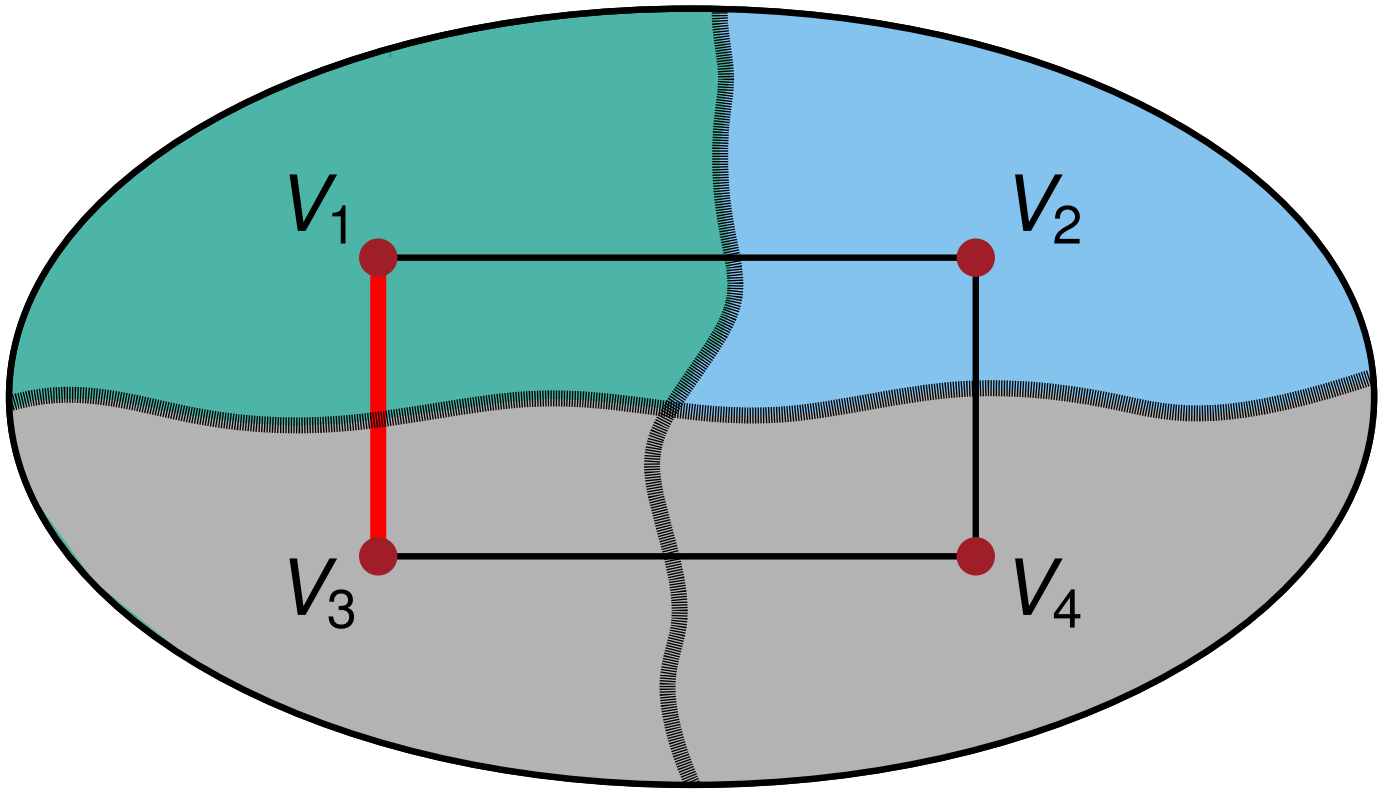
Using 2-way refinement algorithm for **active** blocks of the quotient graph

Active Block Scheduling

KaFFPa [Sanders, Schulz 11]

Round 2

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



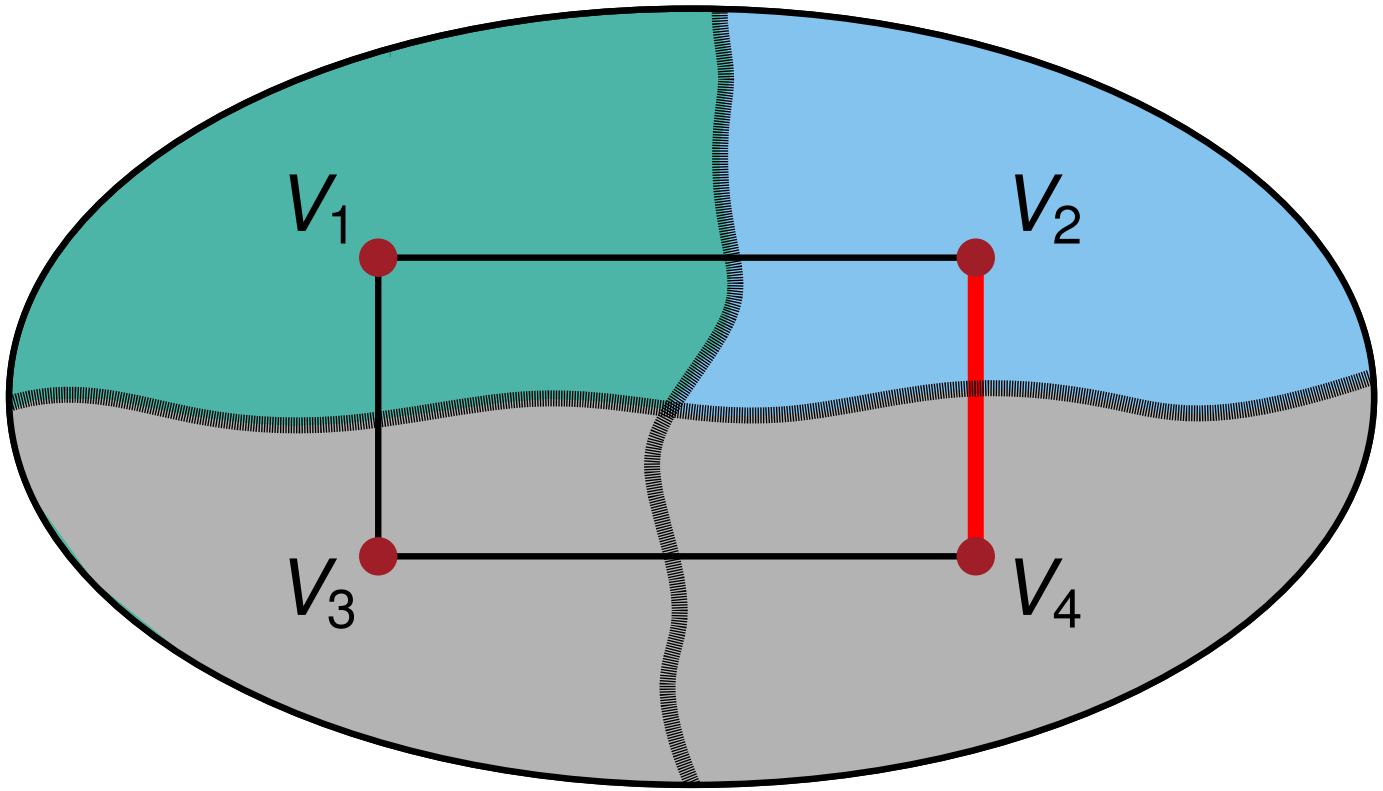
Using 2-way refinement algorithm for **active** blocks of the quotient graph

Active Block Scheduling

KaFFPa [Sanders, Schulz 11]

Round 2

refine(V_2, V_4) = **No Improvement!**

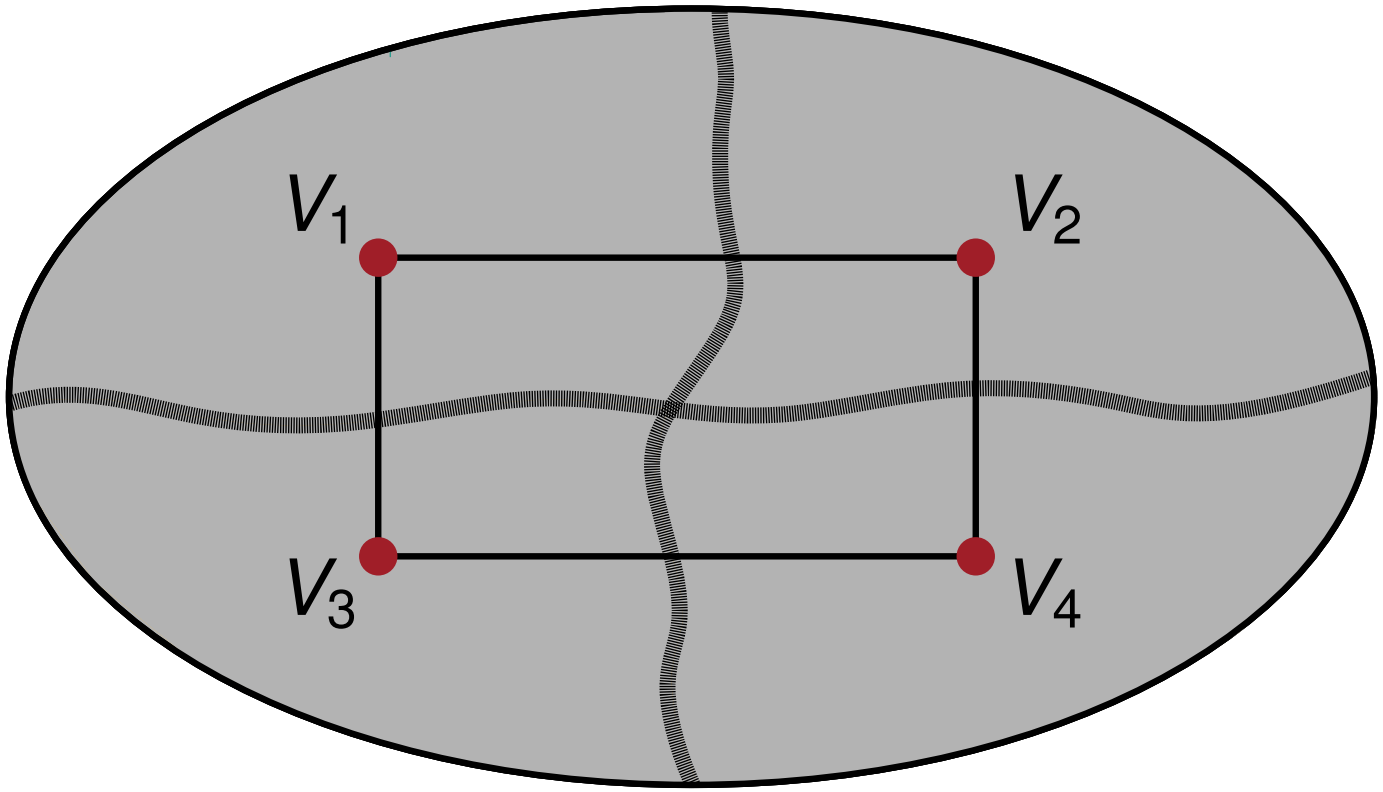


Using 2-way refinement algorithm for **active** blocks of the quotient graph

Active Block Scheduling

KaFFPa [Sanders, Schulz 11]

Round 2 Boundary did not change $\Rightarrow$ Mark block as **inactive**

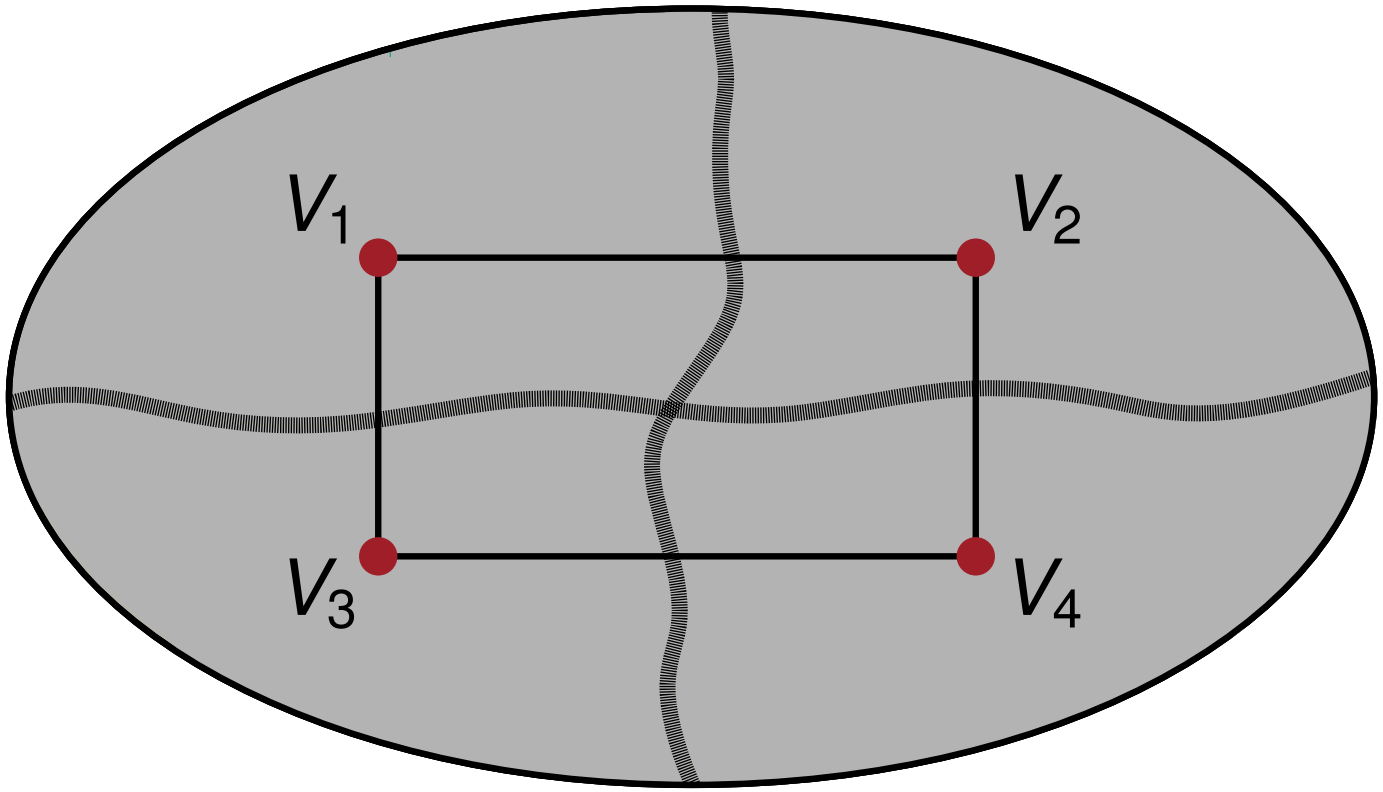


Using 2-way refinement algorithm for **active** blocks of the quotient graph

Active Block Scheduling

KaFFPa [Sanders, Schulz 11]

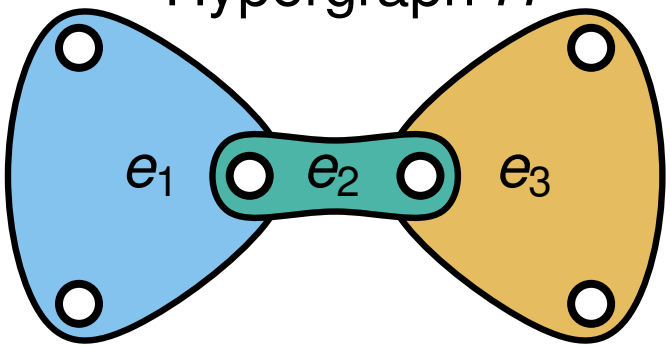
Round 2 All blocks are **inactive** $\Rightarrow$ Algorithm terminates



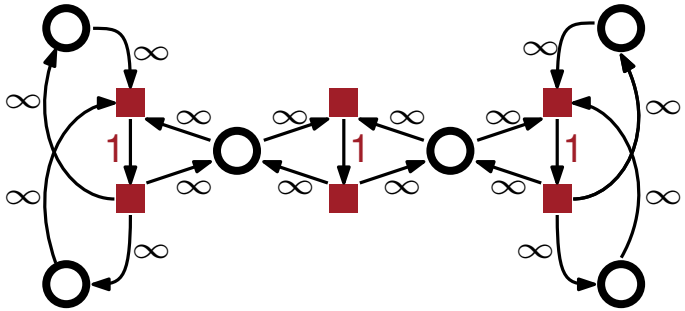
Using 2-way refinement algorithm for **active** blocks of the quotient graph

Hypergraph Flow Network - Graph Edges

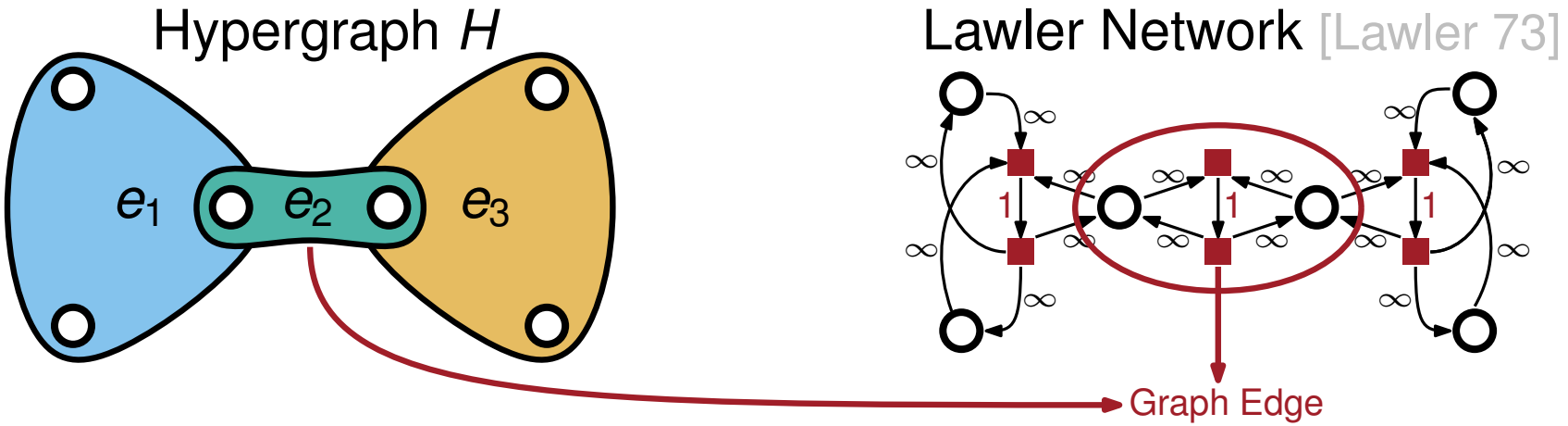
Hypergraph H



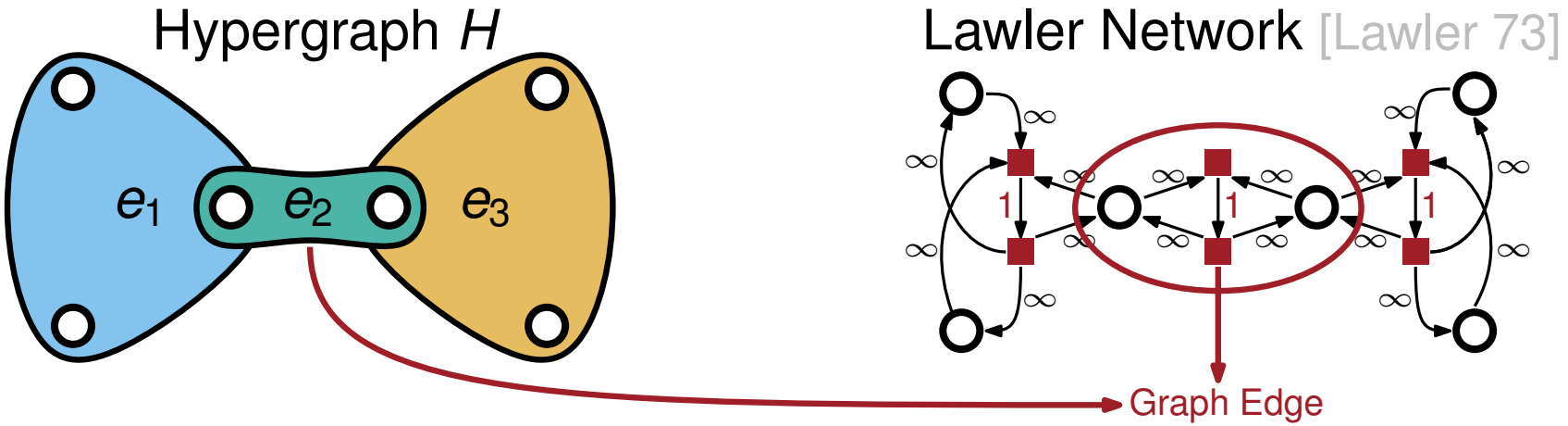
Lawler Network [Lawler 73]

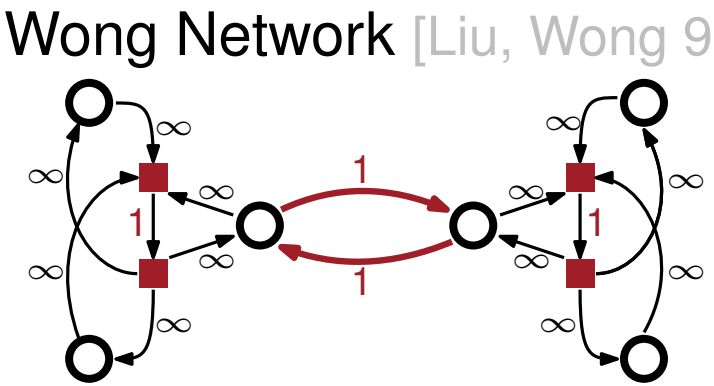


Hypergraph Flow Network - Graph Edges



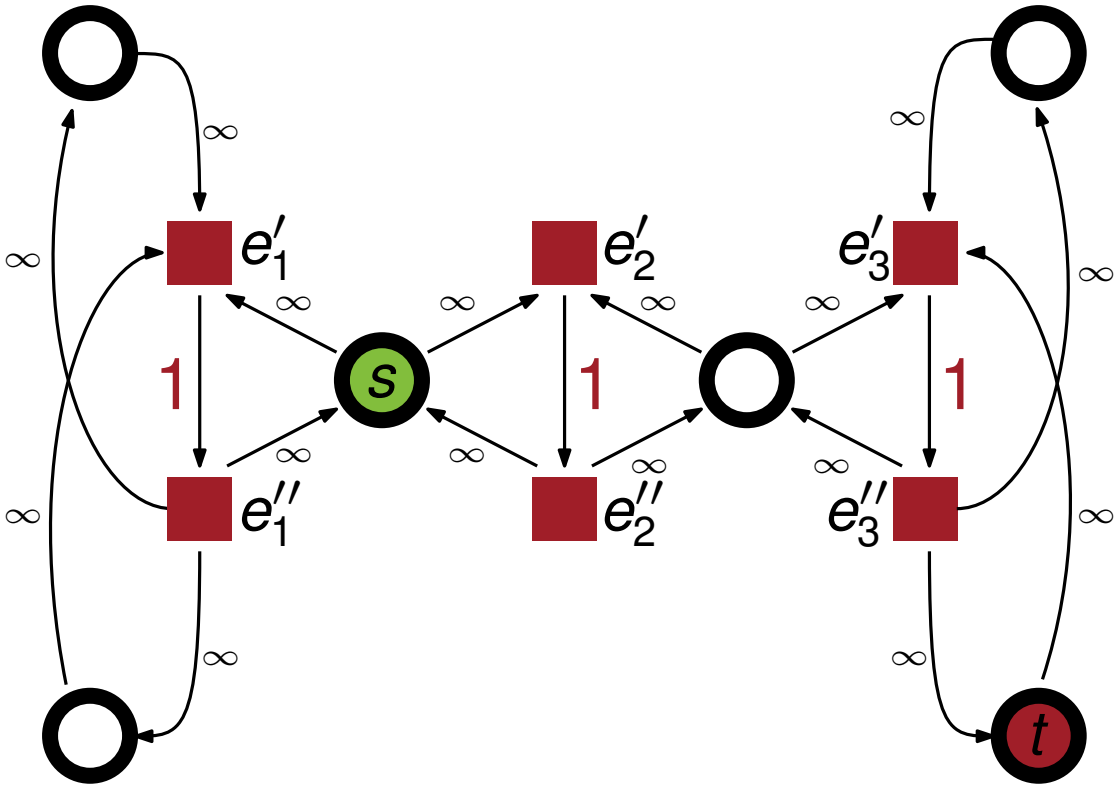
Hypergraph Flow Network - Graph Edges





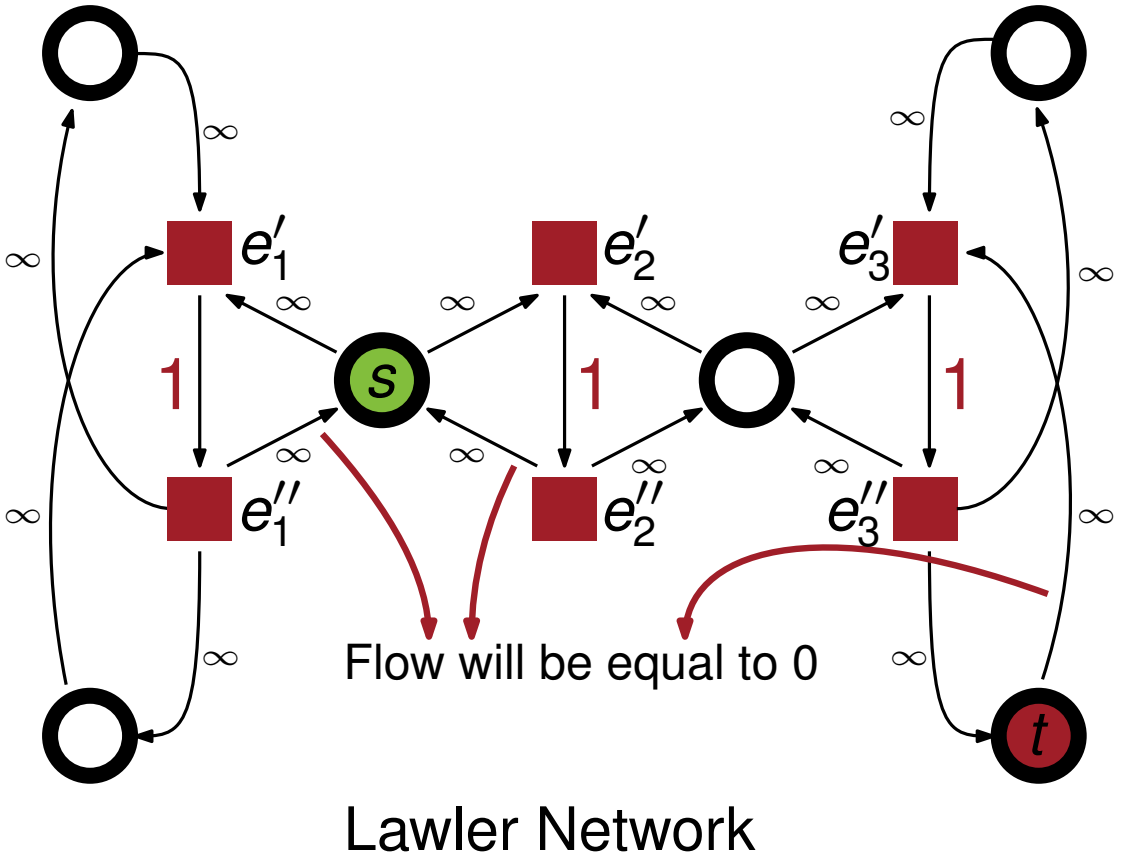
Institute of Theoretical Informatics
Algorithmics Group

Removing Source and Sink Vertices



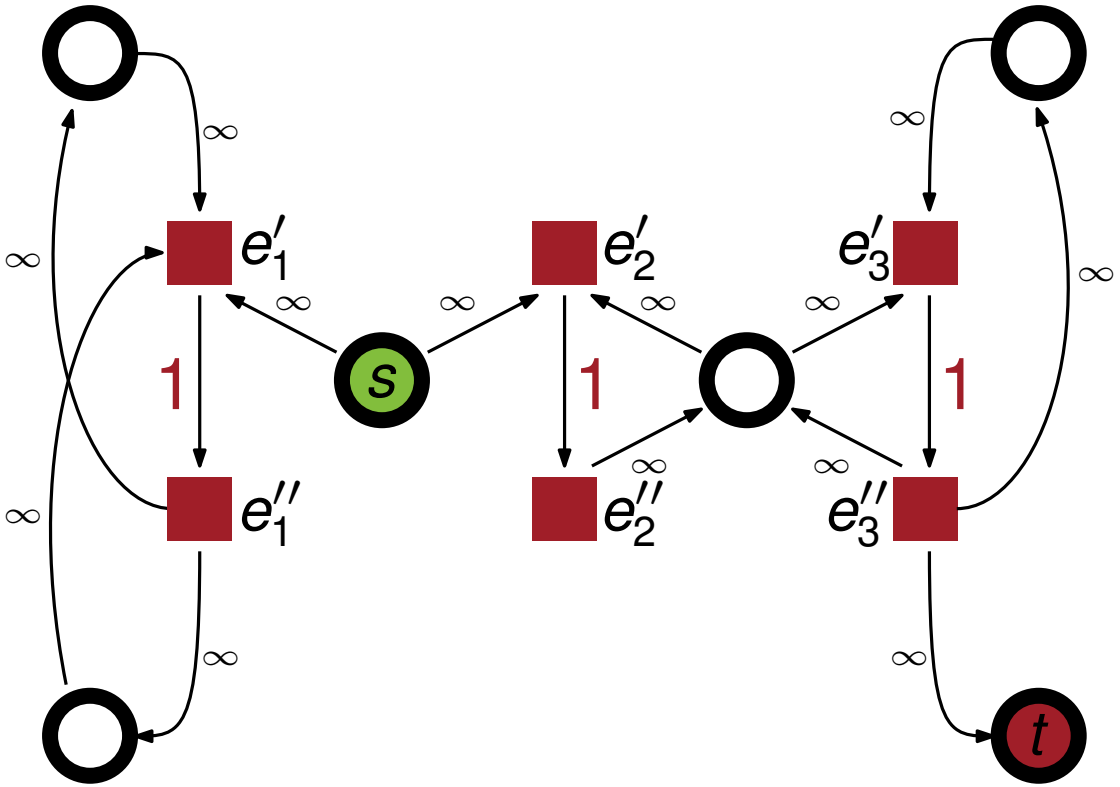
Lawler Network

Removing Source and Sink Vertices



Lawler Network

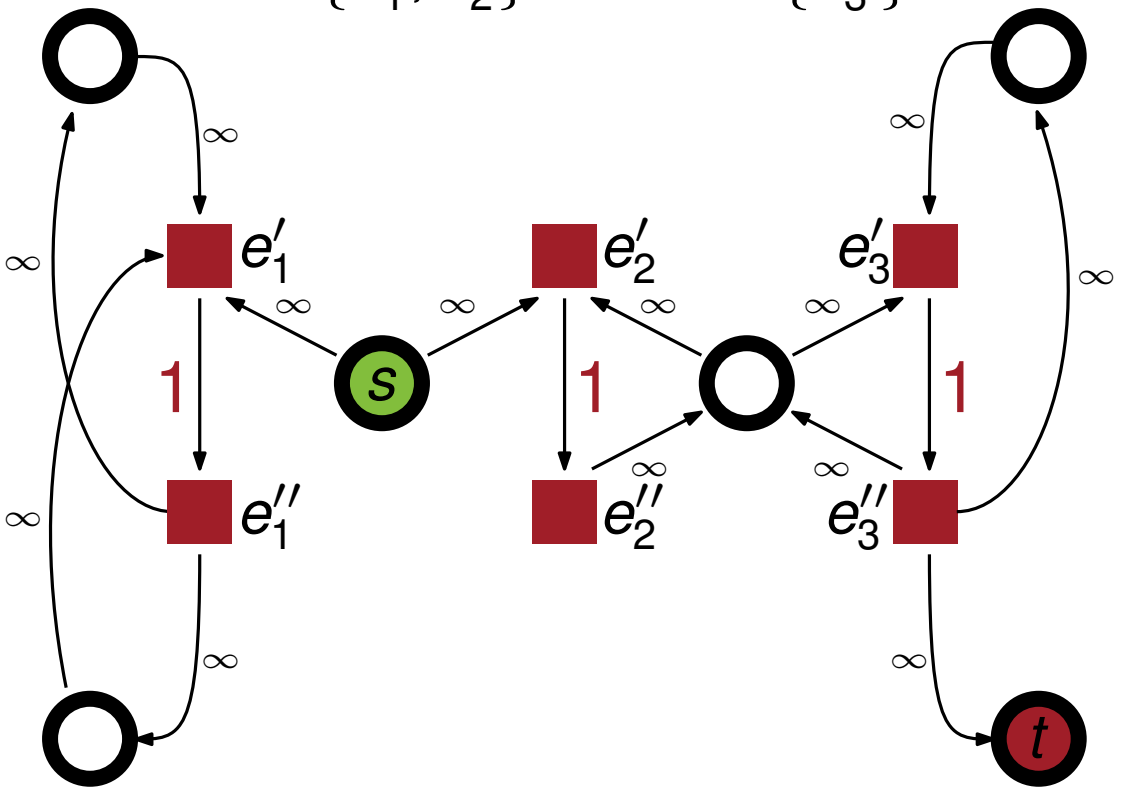
Removing Source and Sink Vertices



Lawler Network

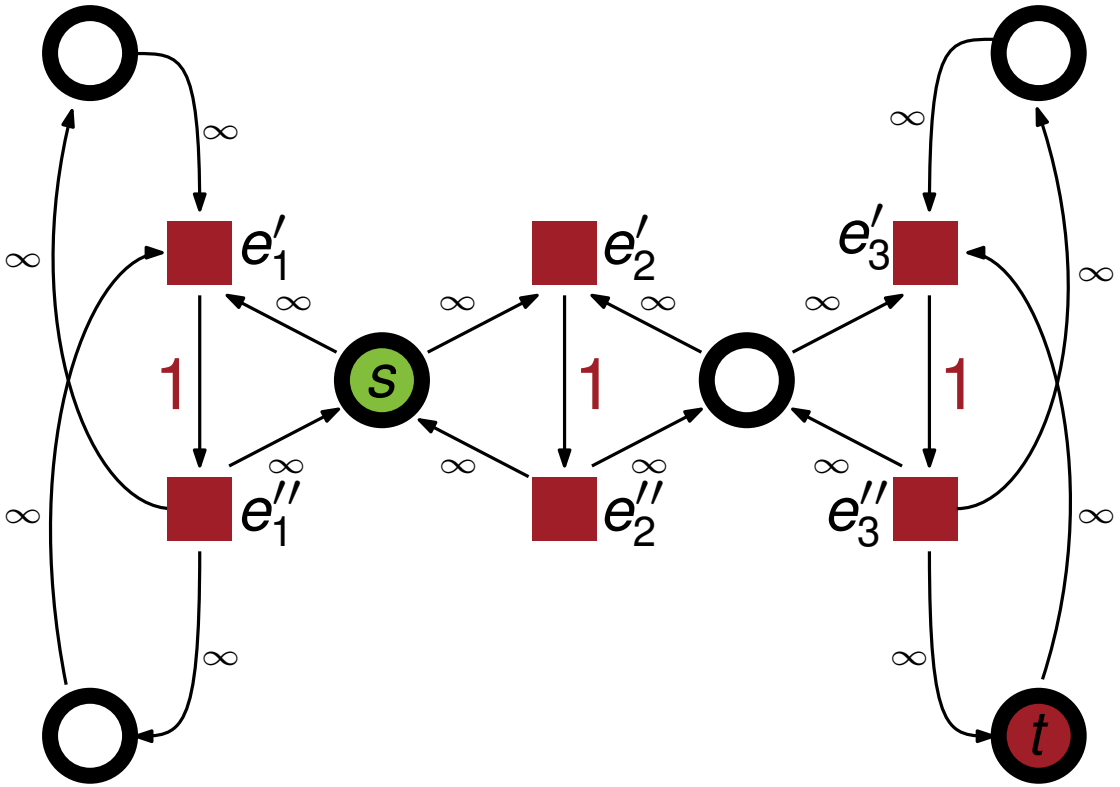
Removing Source and Sink Vertices

Corresponds to *Multi-Source Multi-Sink* problem with
$S = \{e'_1, e'_2\}$ and $T = \{e''_3\}$

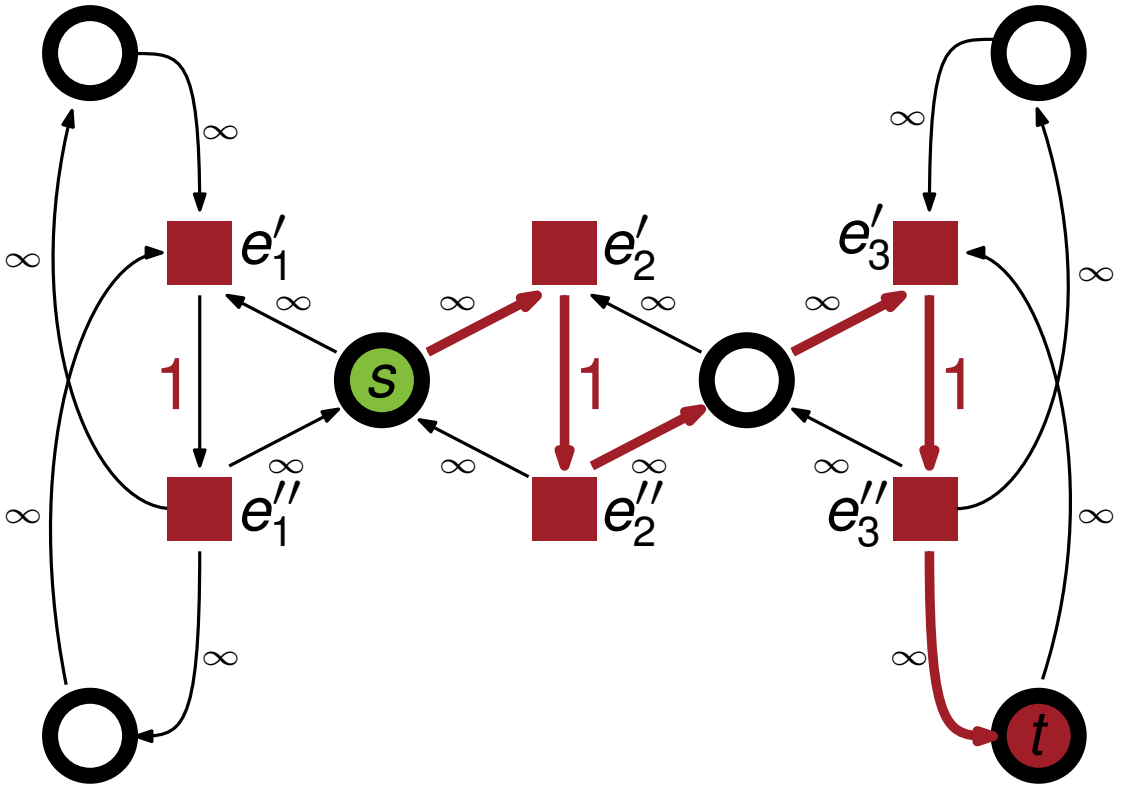


Lawler Network

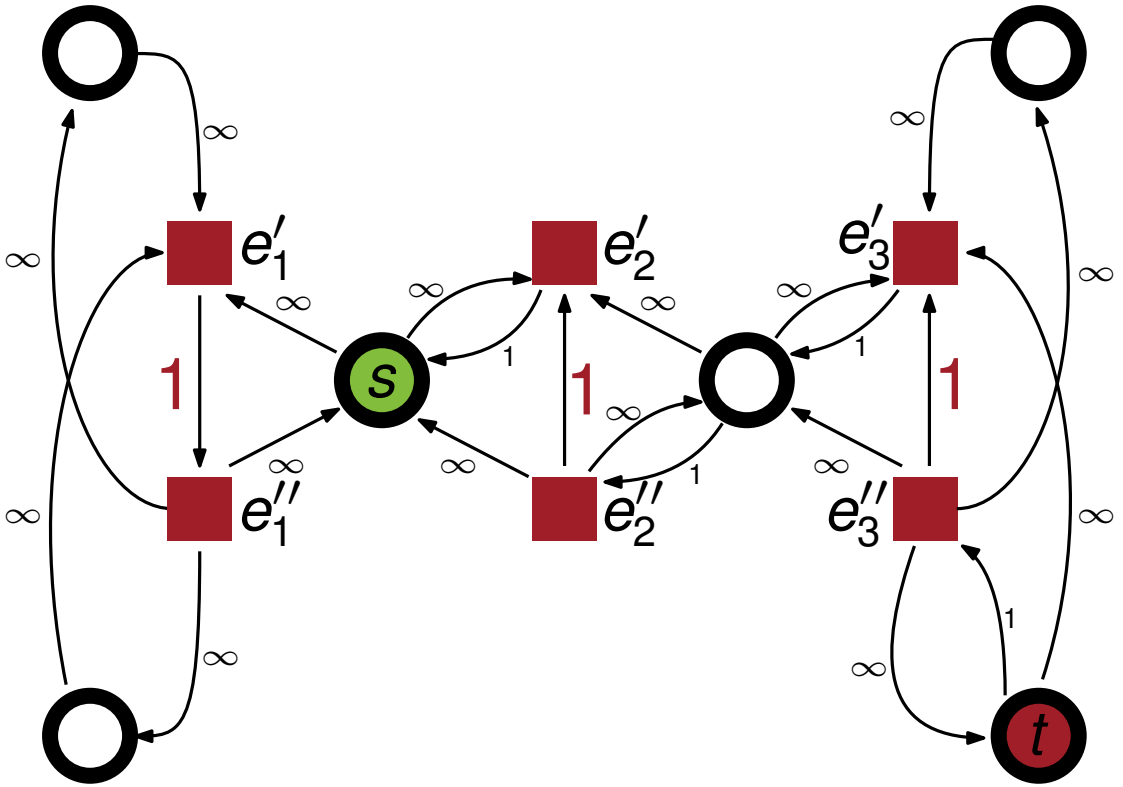
Minimum (s, t)-Bipartition



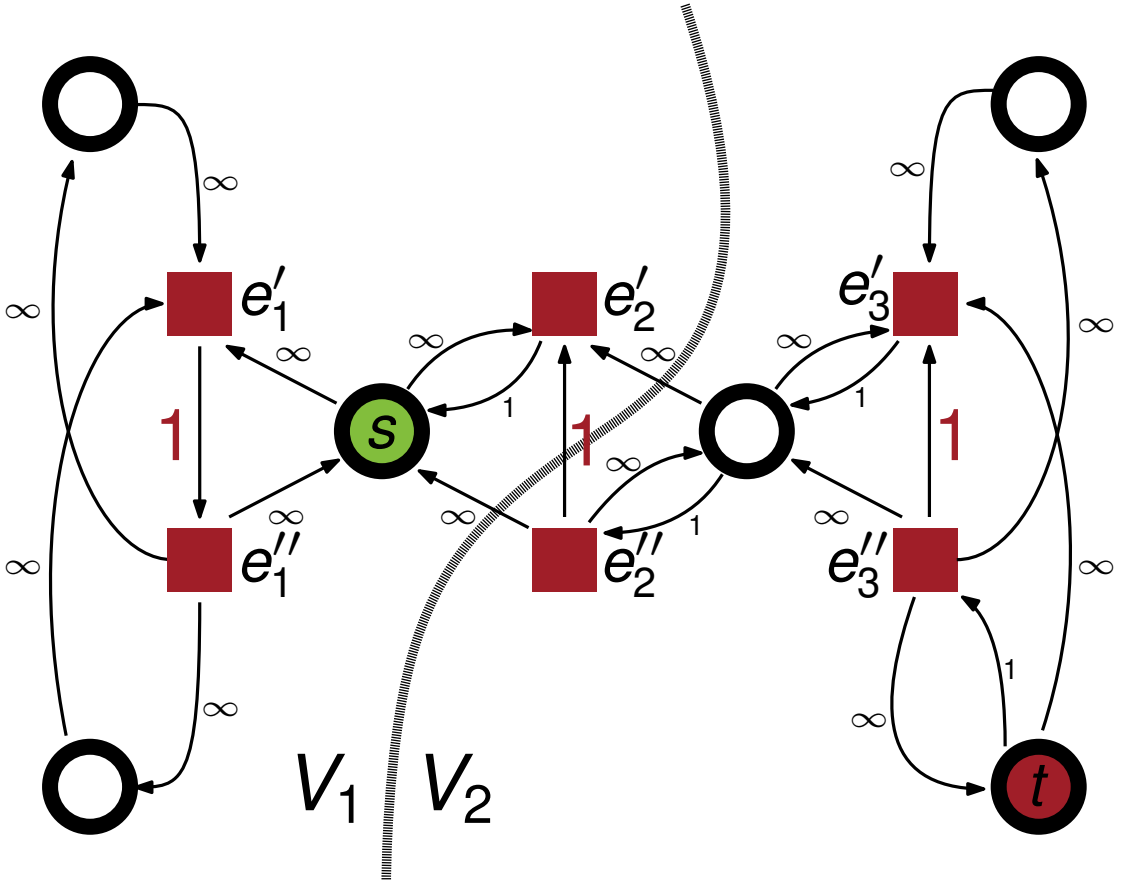
Minimum (s, t)-Bipartition



Minimum (s, t)-Bipartition



Minimum (s, t)-Bipartition

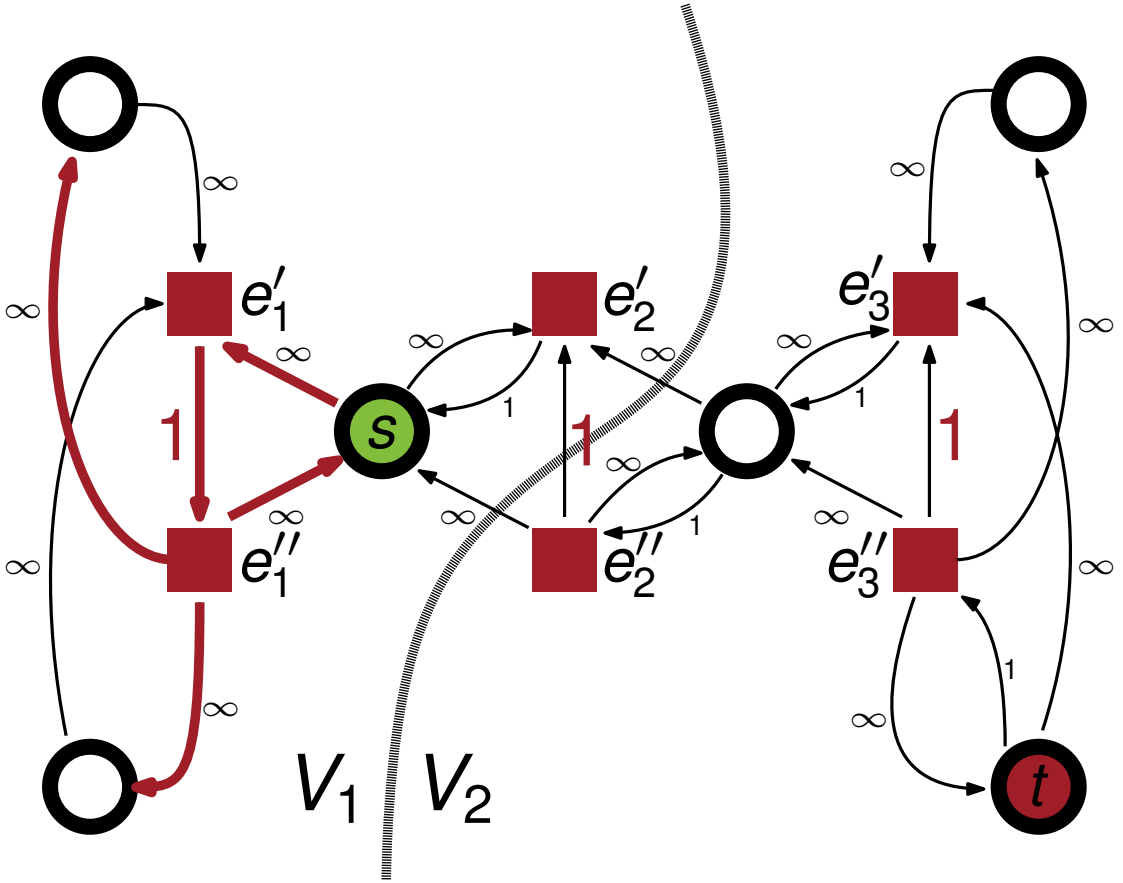


All nodes *reachable* from s are part of V_1 and $V_2 = V \setminus V_1$

Institute of Theoretical Informatics
Algorithmics Group

Minimum (s, t)-Bipartition

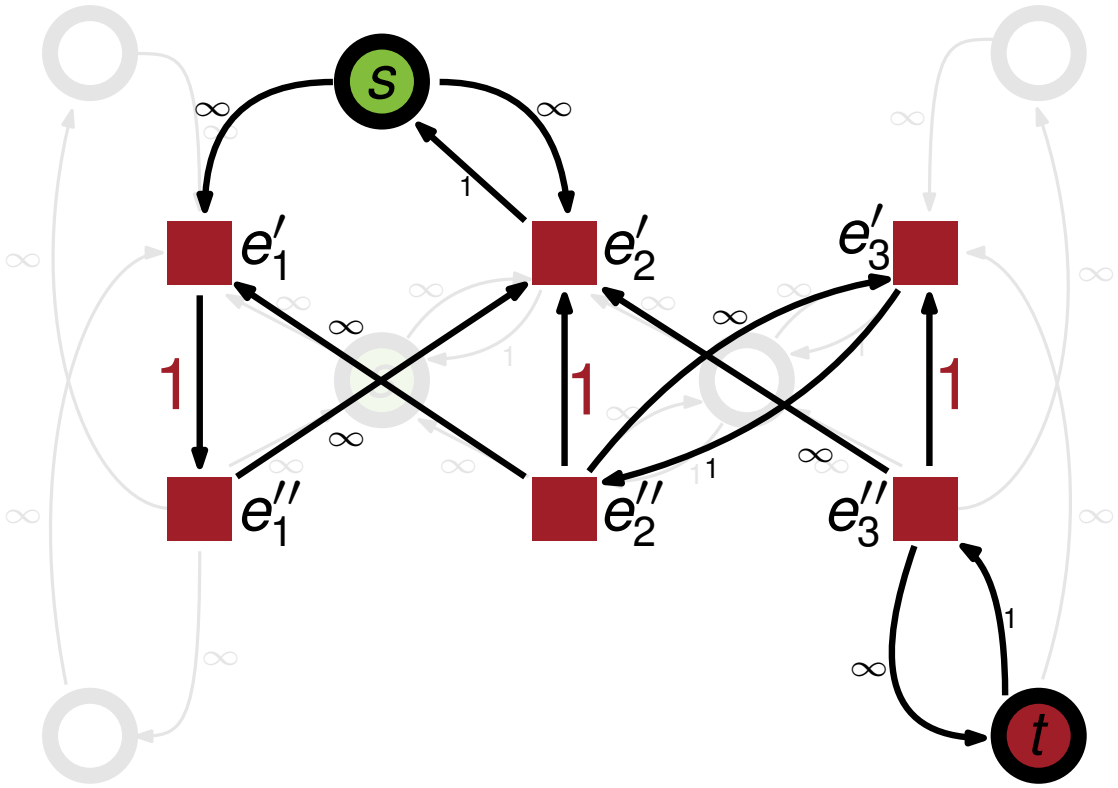
For each hypernode $v \in V_1$, there exists at least one $e \in I(v)$ with $e'' \in V_1$



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

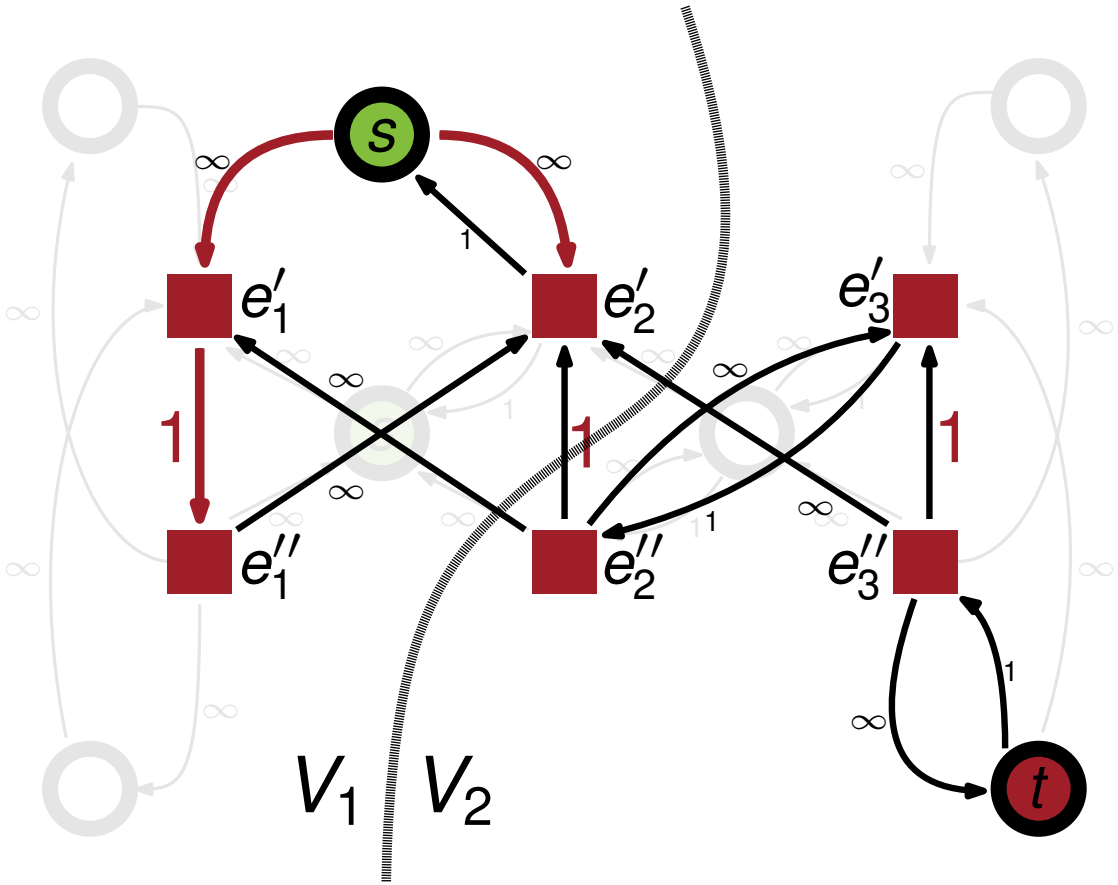
Institute of Theoretical Informatics
Algorithmics Group

Minimum (s, t)-Bipartition



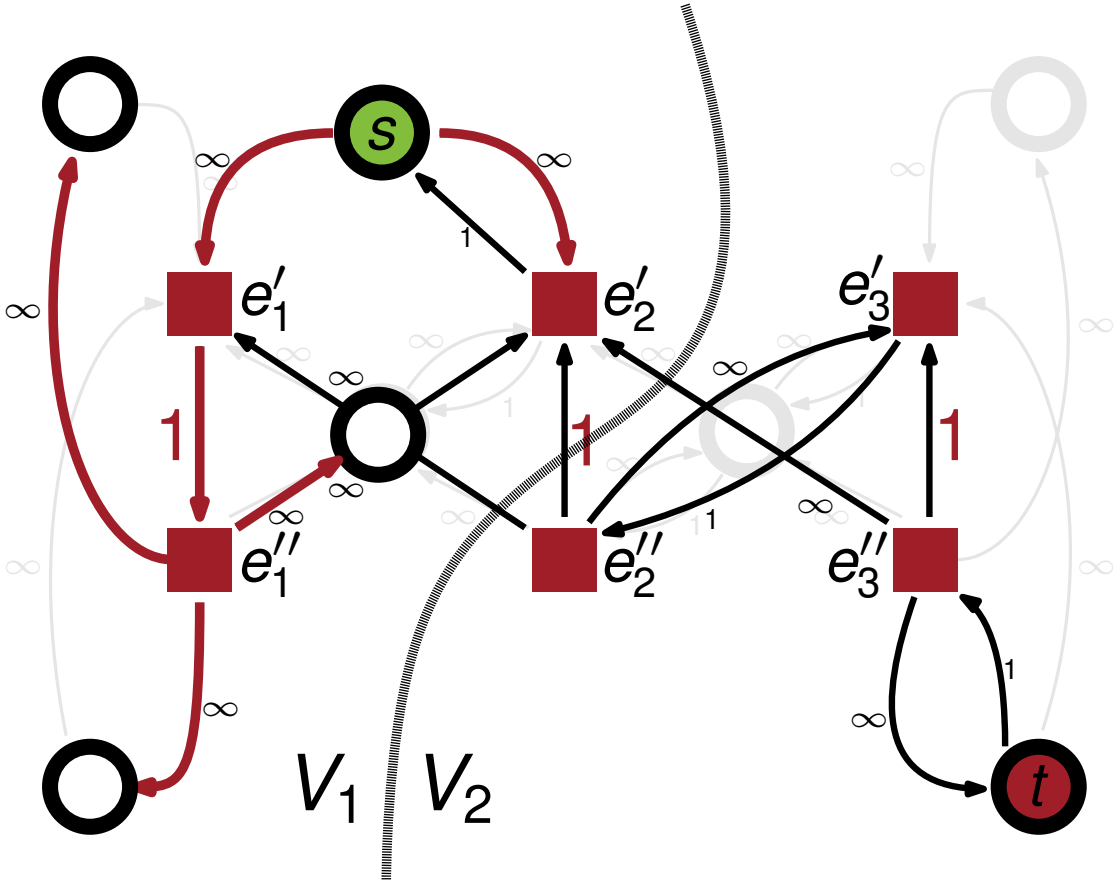
Institute of Theoretical Informatics
Algorithmics Group

Minimum (s, t)-Bipartition



Institute of Theoretical Informatics
Algorithmics Group

Minimum (s, t)-Bipartition



Institute of Theoretical Informatics
Algorithmics Group

Integration into KaHyPar

- KaHyPar is a *n*-level hypergraph partitioner

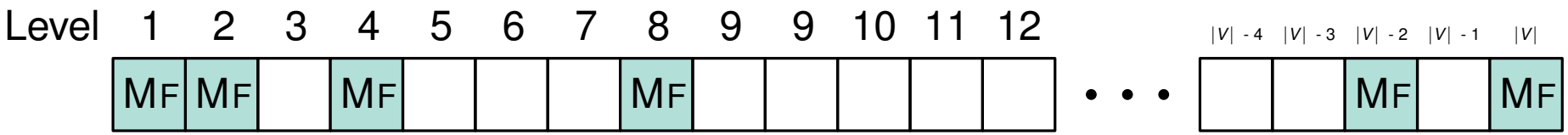
Karlsruhe Institute of Technology
Institute of Theoretical Informatics
Algorithmics Group

Integration into KaHyPar

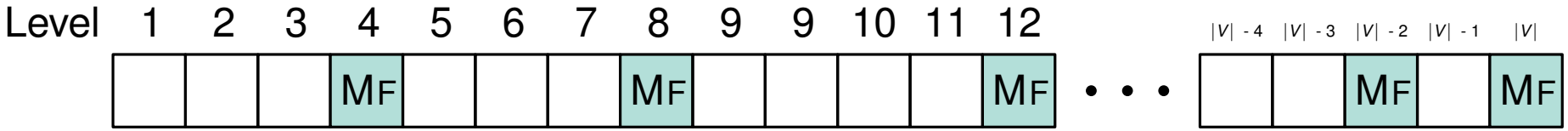
- KaHyPar is a *n*-level hypergraph partitioner

Flow Execution Policies

Exponential: Execute *Max-Flow-Min-Cut* computations (MF) on each level i with $i = 2^j$



Constant: Execute *Max-Flow-Min-Cut* computations (MF) on each level i with $i = \beta \cdot j$

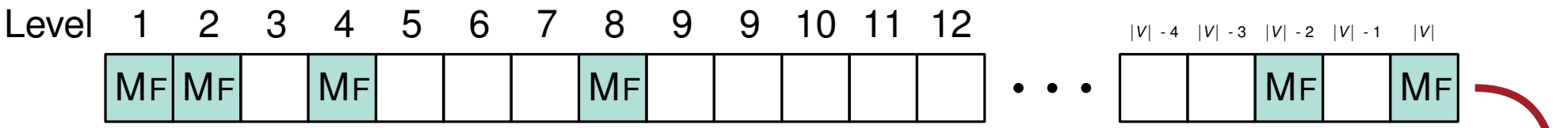


Integration into KaHyPar

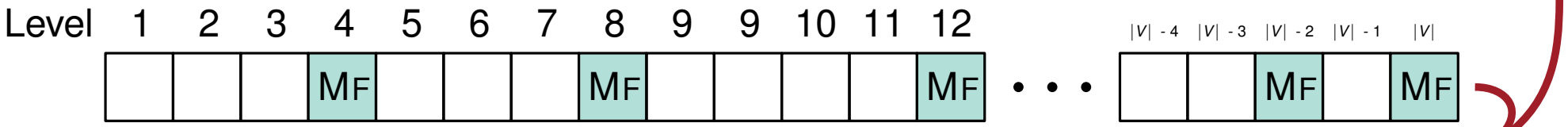
- KaHyPar is a *n*-level hypergraph partitioner

Flow Execution Policies

Exponential: Execute *Max-Flow-Min-Cut* computations (MF) on each level i with $i = 2^j$



Constant: Execute *Max-Flow-Min-Cut* computations (MF) on each level i with $i = \beta \cdot j$



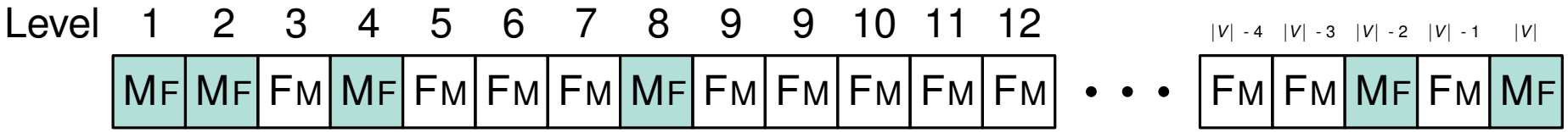
Note, each policy uses *flow*-based refinement on the **last level**

Integration into KaHyPar

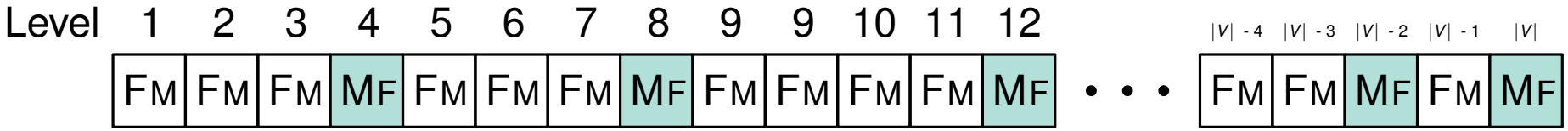
- KaHyPar is a *n*-level hypergraph partitioner

Flow Execution Policies

Exponential: Execute *Max-Flow-Min-Cut* computations (MF) on each level i with $i = 2^j$



Constant: Execute *Max-Flow-Min-Cut* computations (MF) on each level i with $i = \beta \cdot j$



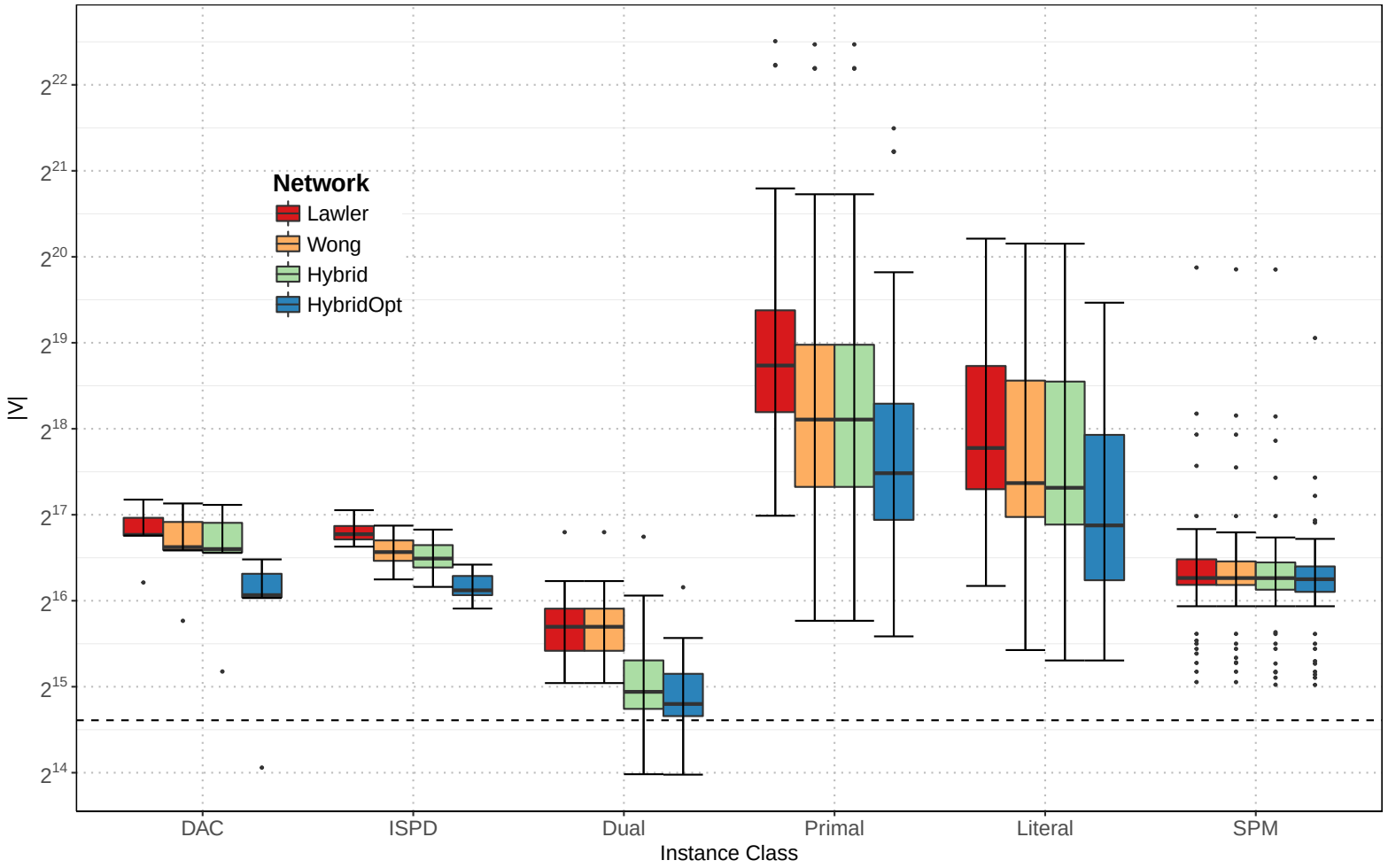
Institute of Theoretical Informatics
Algorithmics Group

Statistics Benchmark Subset

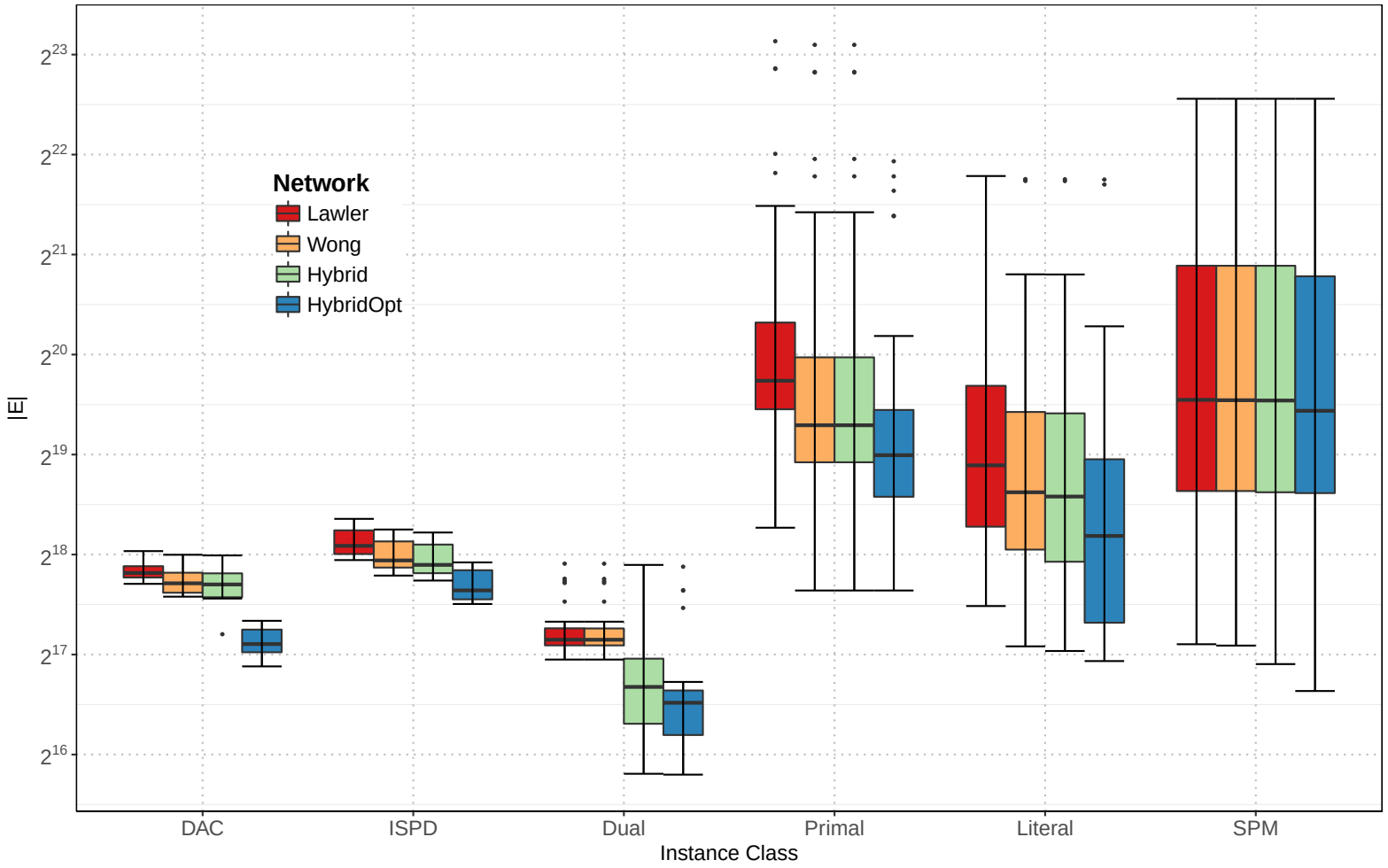
- $\overline{x}$ = mean
- $\widetilde{x}$ = median

Type	#	$\overline{d(v)}$	$\widetilde{d(v)}$	$\overline{\lvert e \rvert}$	$\widetilde{\lvert e \rvert}$
DAC	5	3.32	3.28	3.37	3.35
ISPD98	10	4.20	4.24	3.89	3.90
PRIMAL	30	16.29	9.97	2.63	2.39
LITERAL	30	8.21	4.99	2.63	2.39
DUAL	30	2.63	2.38	16.29	9.97
SPM	60	24.78	14.15	26.58	15.01

Flow Networks - Number of Nodes



Flow Networks - Number of Edges



Flow Configuration

- +/− F = Flow-based refinement
- +/− M = Most Balanced Minimum Cut
- +/− FM = FM Heuristic
- CONSTANT128 = (+F, +M, +FM) with constant flow execution policy ($\beta = 128$)

	(+F,-M,-FM)		(+F,+M,-FM)		(+F,-M,+FM)		(+F,+M,+FM)		CONSTANT128	
α'	Avg [%]	$t[s]$	Avg [%]	$t[s]$	Avg [%]	$t[s]$	Avg [%]	$t[s]$	Avg [%]	$t[s]$
1	−6.09	12.9	−5.60	13.4	0.25	15.0	0.23	15.2	0.54	67.9
2	−3.19	15.8	−2.07	16.7	0.59	17.0	0.73	17.5	1.11	140.2
4	−1.82	20.4	−0.19	21.9	0.90	20.4	1.21	21.5	1.66	269.6
8	−0.85	28.5	0.98	30.7	1.24	26.5	1.71	28.7	2.20	512.1
16	−0.19	43.3	1.75	46.7	1.60	37.5	2.21	41.3	2.76	973.9
Ref.	(-F,-M,+FM)		6373.88	13.73						

Institute of Theoretical Informatics
Algorithmics Group

Effectiveness Tests

- $+/-$ F = Flow-based refinement
- $+/-$ M = Most Balanced Minimum Cut
- $+/-$ FM = FM Heuristic

Config α'	(+F,-M,-FM) Avg [%]	(+F,+M,-FM) Avg [%]	(+F,-M,+FM) Avg [%]	(+F,+M,+FM) Avg [%]
1	−6.06	−5.52	0.23	0.24
2	−3.15	−2.06	0.55	0.73
4	−1.89	−0.19	0.86	1.20
8	−0.87	0.96	1.20	1.69
16	−0.29	1.66	1.52	2.17
Ref.	(-F,-M,+FM)	6377.15		

Quality - Full Benchmark Set

Algorithm	Min. $(\lambda - 1)$			
	ALL	DAC	ISPD98	PRIMAL
KaHyPar-MF	7542.88	16828.15	5511.40	15236.13
KaHyPar-CA	2.22	2.80	1.92	1.85
hMetis-R	14.40	4.75	2.76	3.88
hMetis-K	12.92	7.77	2.17	4.78
PaToH-Q	11.48	15.24	9.53	14.36
PaToH-D	12.06	15.57	10.90	12.47
	LITERAL	DUAL	SPM	WEBSOC
KaHyPar-MF	15197.60	2927.42	6010.05	7478.06
KaHyPar-CA	2.46	3.33	1.74	3.91
hMetis-R	4.17	31.20	16.37	41.92
hMetis-K	6.91	21.51	16.23	40.45
PaToH-Q	14.98	11.44	8.36	18.45
PaToH-D	15.17	13.64	9.45	23.04

Quality - Full Benchmark Set

Algorithm	Avg. $(\lambda - 1)$			
	ALL	DAC	ISPD98	PRIMAL
KaHyPar-MF	7782.51	17476.91	5643.99	15863.75
KaHyPar-CA	2.44	3.04	2.16	2.03
hMetis-R	13.61	3.53	1.60	2.03
hMetis-K	13.23	7.82	1.37	3.89
PaToH-Q	8.67	11.97	7.35	10.83
PaToH-D	14.35	19.21	13.11	15.45
	LITERAL	DUAL	SPM	WEBSOC
KaHyPar-MF	15770.06	3038.55	6143.73	7783.87
KaHyPar-CA	2.66	3.39	2.07	3.84
hMetis-R	2.72	30.28	16.42	41.32
hMetis-K	8.49	22.26	16.33	40.14
PaToH-Q	11.78	8.08	6.33	15.12
PaToH-D	17.72	15.47	11.43	24.18

Running Time - Full Benchmark Set

Algorithm	Running Time $t[s]$			
	ALL	DAC	ISPD98	PRIMAL
KaHyPar-MF	55.67	504.27	20.83	61.78
KaHyPar-CA	31.05	368.97	12.35	32.91
hMetis-R	79.23	446.36	29.03	66.25
hMetis-K	57.86	240.92	23.18	44.23
PaToH-Q	5.89	28.34	1.89	6.90
PaToH-D	1.22	6.45	0.35	1.12
	LITERAL	DUAL	SPM	WEBSOC
KaHyPar-MF	119.51	97.22	27.40	110.15
KaHyPar-CA	64.65	68.27	13.91	67.14
hMetis-R	142.12	200.36	41.79	89.69
hMetis-K	94.89	125.55	35.95	111.95
PaToH-Q	9.24	10.57	3.42	4.71
PaToH-D	1.58	2.87	0.77	0.88

Quality per k - Full Benchmark Set

Algorithm	Min. $(\lambda - 1)$			
	ALL	$k = 2$	$k = 4$	$k = 8$
KaHyPar-MF	7542.88	1005.76	2985.22	5805.19
KaHyPar-CA	2.22	1.71	2.16	2.51
hMetis-R	14.40	22.25	17.62	15.63
hMetis-K	12.92	21.82	13.66	12.76
PaToH-Q	11.48	14.92	12.60	11.81
PaToH-D	12.06	8.54	10.41	13.64
	$k = 16$	$k = 32$	$k = 64$	$k = 128$
KaHyPar-MF	9097.31	14352.34	21537.33	31312.48
KaHyPar-CA	2.51	2.45	2.16	2.05
hMetis-R	14.29	11.94	9.80	8.01
hMetis-K	13.49	10.62	9.18	7.81
PaToH-Q	11.66	10.66	9.77	8.63
PaToH-D	14.50	12.70	12.66	11.89

Quality per k - Full Benchmark Set

Algorithm	Avg. $(\lambda - 1)$			
	ALL	$k = 2$	$k = 4$	$k = 8$
KaHyPar-MF	7782.51	1057.93	3130.20	6032.58
KaHyPar-CA	2.44	2.27	2.57	2.80
hMetis-R	13.61	21.38	15.92	14.47
hMetis-K	13.23	21.63	15.15	13.61
PaToH-Q	8.67	10.51	8.36	8.35
PaToH-D	14.35	13.26	14.24	16.07
	$k = 16$	$k = 32$	$k = 64$	$k = 128$
KaHyPar-MF	9362.55	14693.96	21893.59	31706.57
KaHyPar-CA	2.68	2.48	2.24	2.05
hMetis-R	13.63	11.35	9.63	7.80
hMetis-K	13.49	10.52	9.30	7.83
PaToH-Q	9.09	8.54	8.27	7.48
PaToH-D	16.59	13.87	13.61	12.66

Running Time per k - Full Benchmark Set

Algorithm	Running Time $t[s]$			
	ALL	$k = 2$	$k = 4$	$k = 8$
KaHyPar-MF	55.67	19.75	32.89	47.52
KaHyPar-CA	31.05	12.68	17.16	23.88
hMetis-R	79.23	27.87	51.59	74.74
hMetis-K	57.86	25.47	32.27	42.50
PaToH-Q	5.89	1.93	3.61	5.44
PaToH-D	1.22	0.43	0.77	1.12
	$k = 16$	$k = 32$	$k = 64$	$k = 128$
KaHyPar-MF	60.38	78.51	100.34	119.15
KaHyPar-CA	31.01	41.69	57.35	76.61
hMetis-R	91.09	109.13	128.66	149.34
hMetis-K	53.41	74.00	109.12	152.92
PaToH-Q	7.01	8.40	10.06	11.44
PaToH-D	1.42	1.71	2.02	2.29