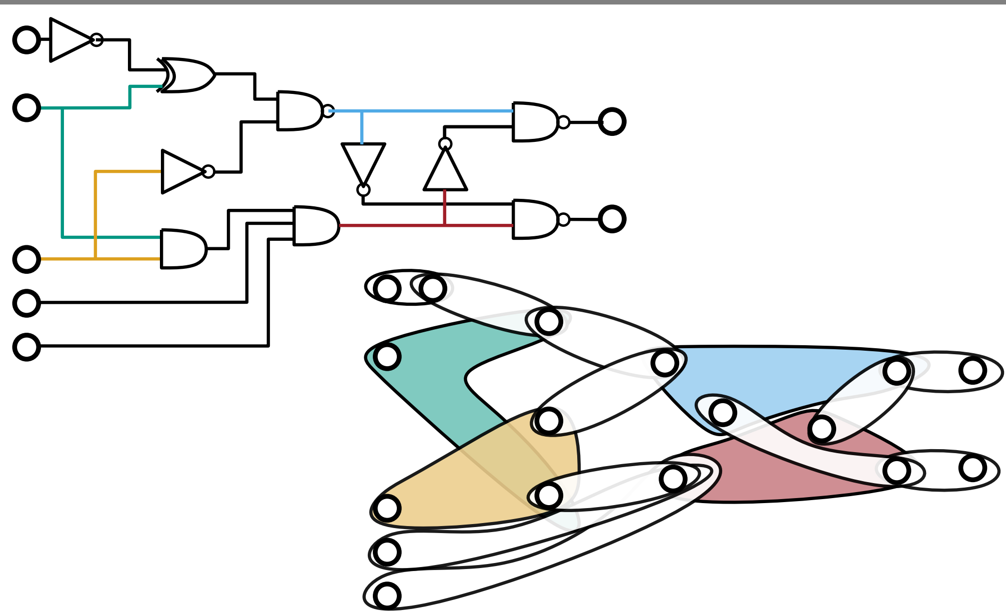
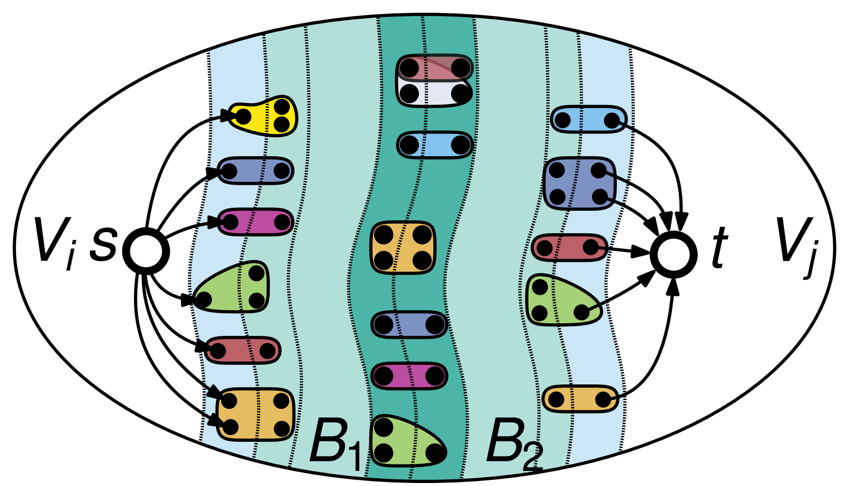
Network Flow-Based Refinement for Multilevel Hypergraph Partitioning

SEA’18 · June 27, 2018

Tobias Heuer, Peter Sanders, Sebastian Schlag

Institute of Theoretical Informatics ·



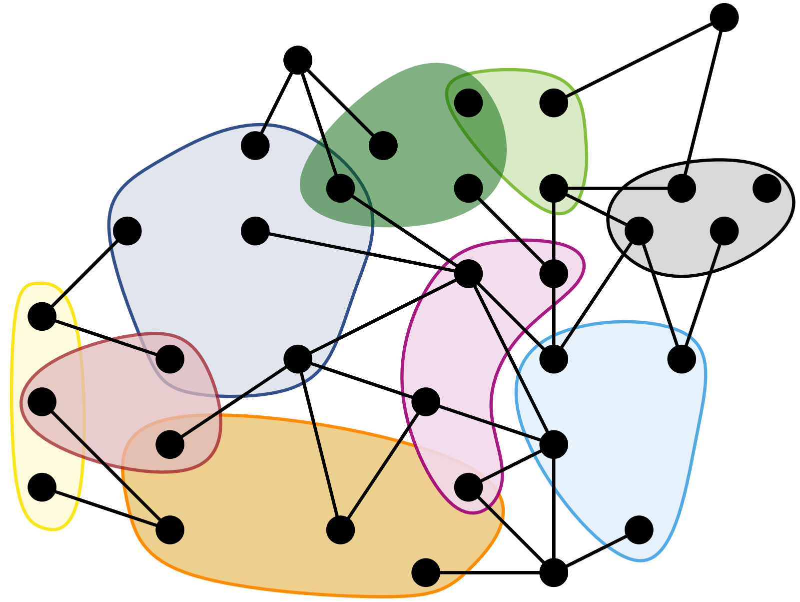


Hypergraphs

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

- graphs $\Rightarrow$ dyadic ($\mathbf{2}$**-ary**) relationships
- hypergraphs $\Rightarrow$ ($\mathbf{d}$**-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 $\omega : E \to \mathbb{R}_{\geq 1}$

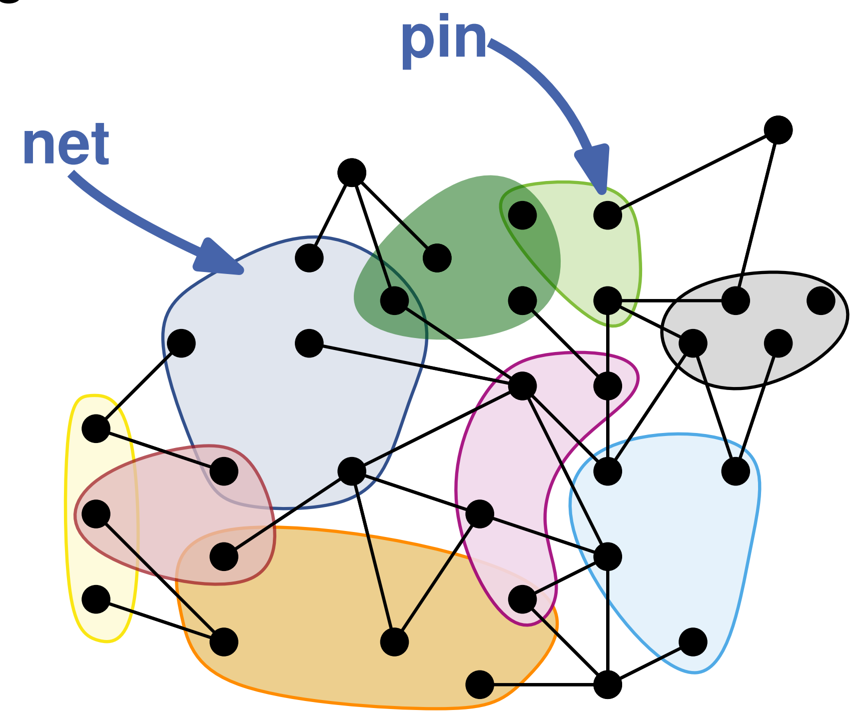


Hypergraphs

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

- graphs $\Rightarrow$ dyadic ($\mathbf{2}$**-ary**) relationships
- hypergraphs $\Rightarrow$ ($\mathbf{d}$**-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 $\omega : E \rightarrow \mathbb{R}_{\geq 1}$

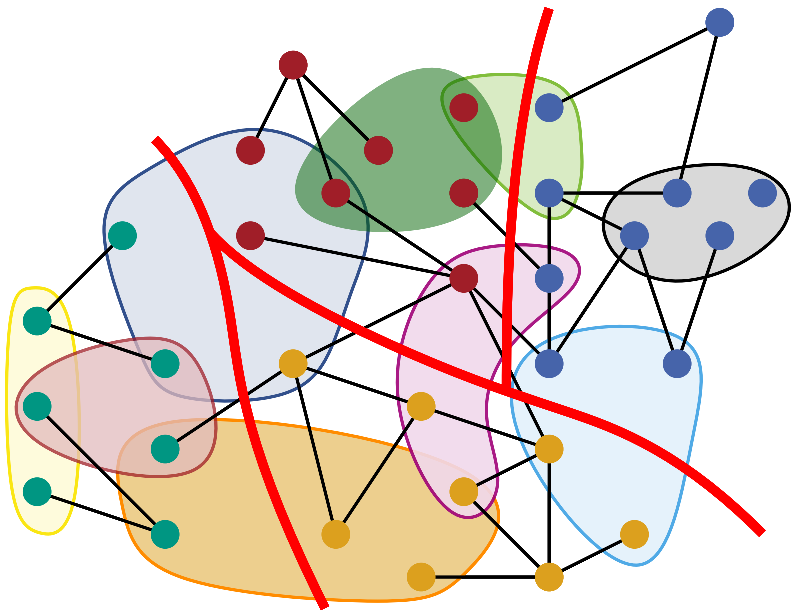


ε-Balanced Hypergraph Partitioning Problem

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

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

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



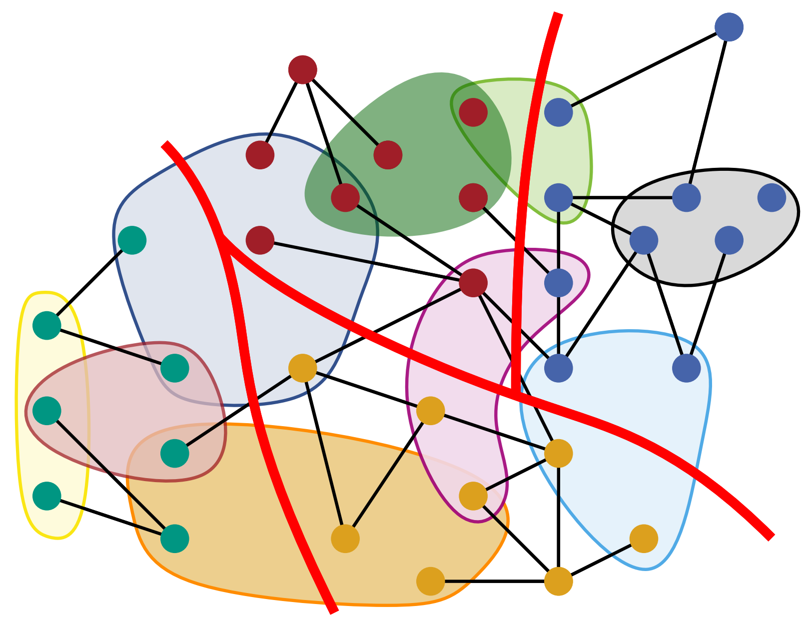
ε-Balanced Hypergraph Partitioning Problem

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

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

imbalance parameter

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



ε-Balanced Hypergraph Partitioning Problem

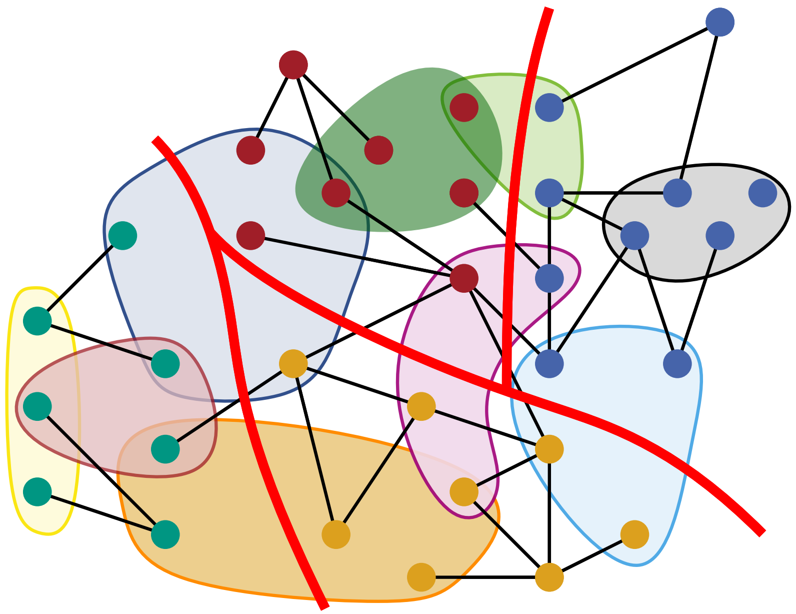
Partition hypergraph $H = (V, E, c, \omega)$ into $\mathbf{k}$ disjoint blocks $\Pi = \{V_1, \ldots, V_k\}$ such that:

- 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**:



ε-Balanced Hypergraph Partitioning Problem

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

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

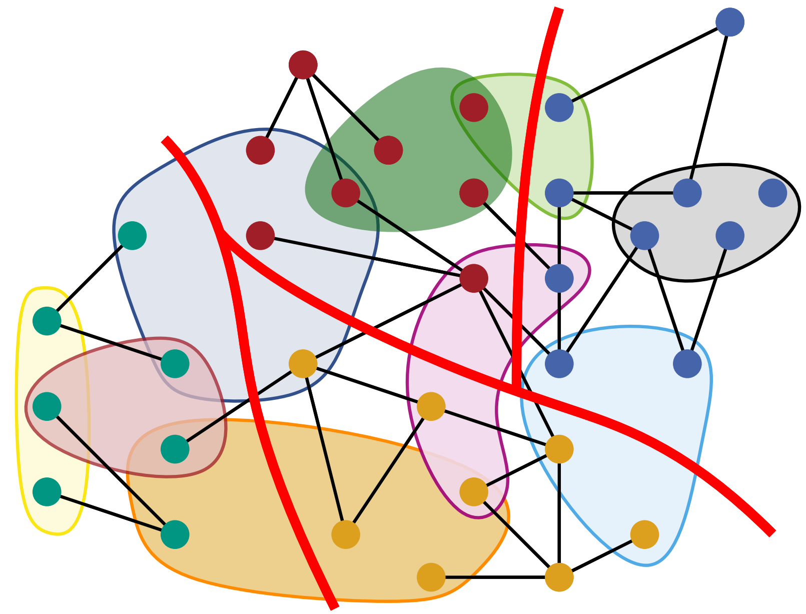
imbalance parameter

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

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

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

connectivity:
 # blocks connected by net *e*



ε-Balanced Hypergraph Partitioning Problem

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

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

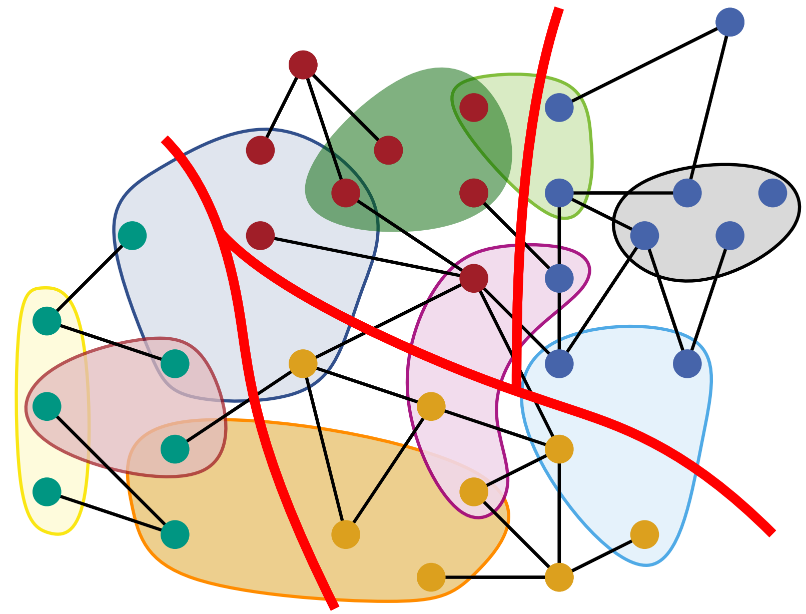
$$c(V_i) \leq (1 + \varepsilon) \left\lceil \frac{c(V)}{k} \right\rceil$$

imbalance parameter

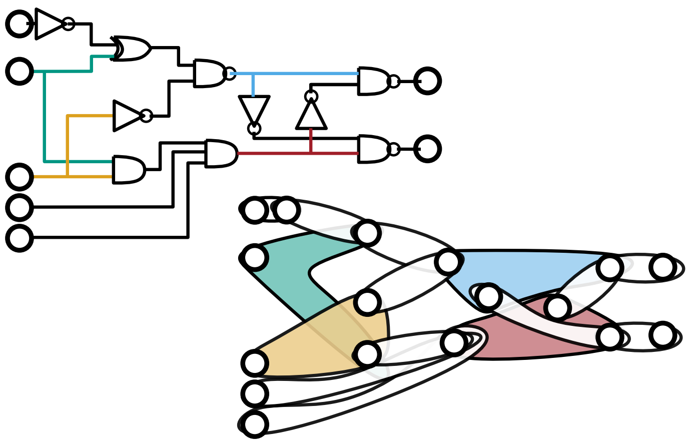
- **connectivity** objective is **minimized**:

$$\sum_{e \in \text{cut}} (\lambda - 1)\, \omega(e) = 12$$

connectivity:
 # blocks connected by net e



Applications

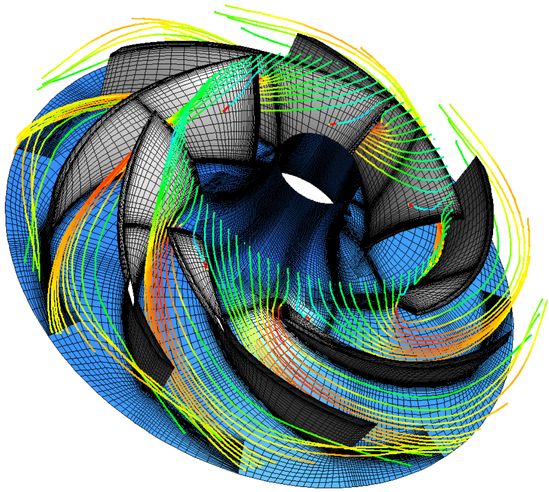


VLSI Design

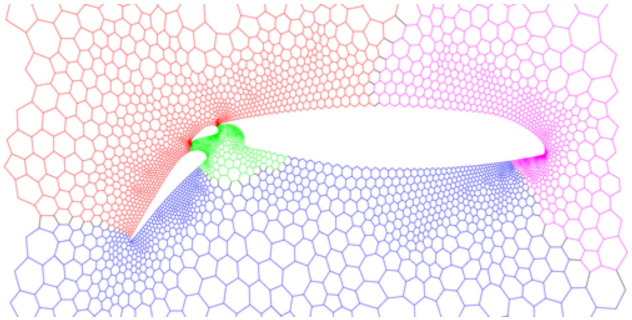
Warehouse Optimization

Complex Networks

Route Planning



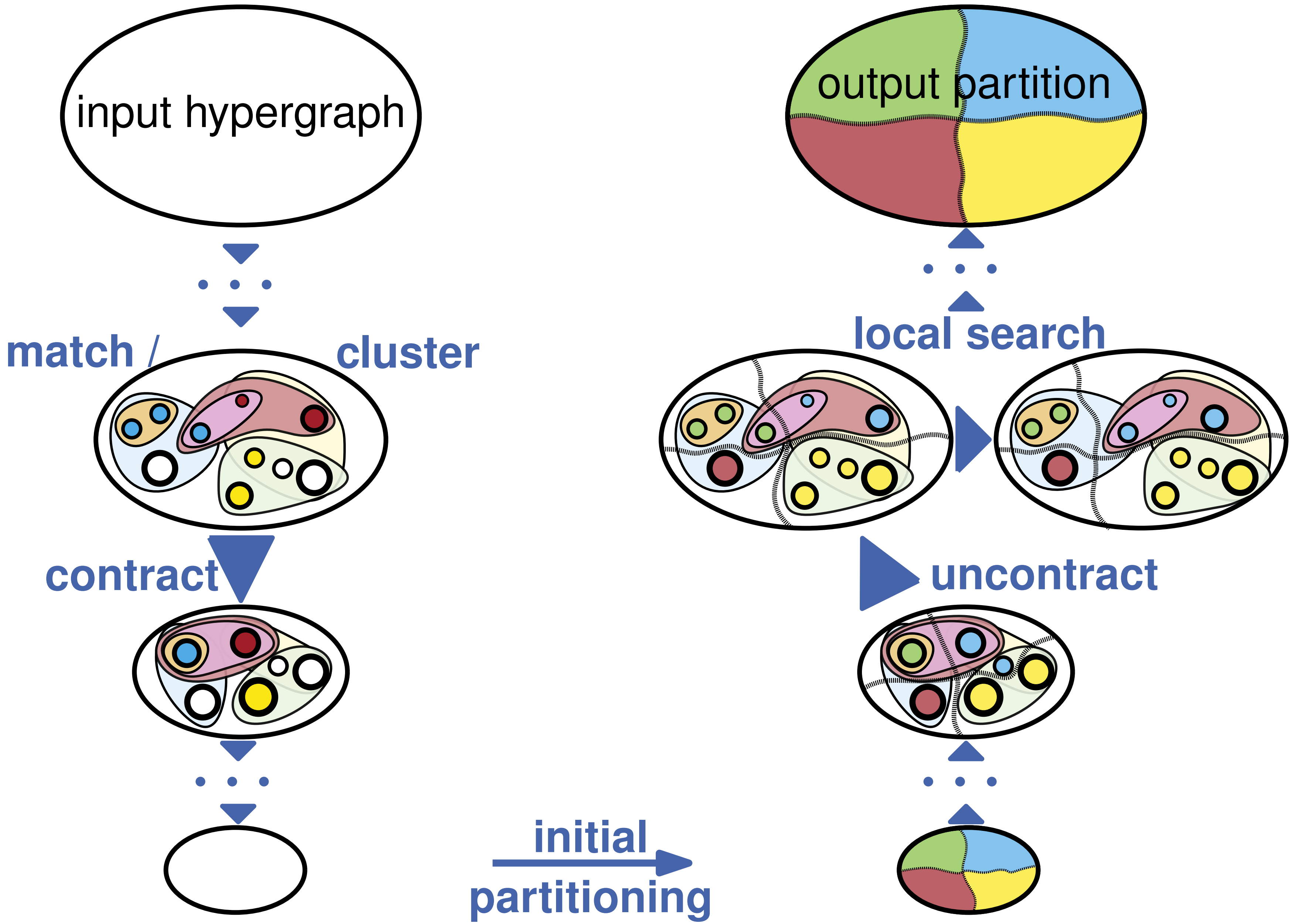
Simulation



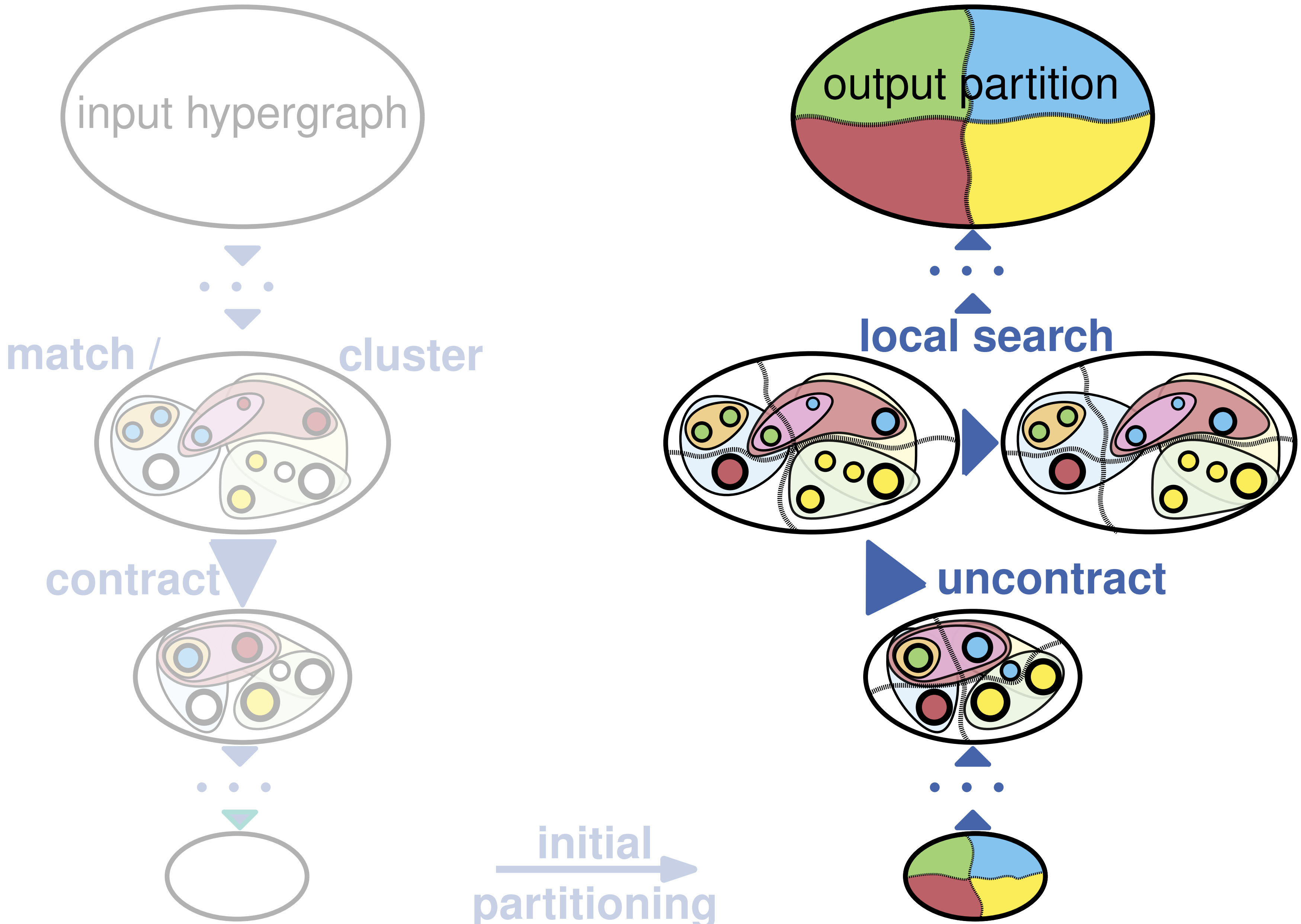
$$\mathbb{R}^{n\times n} \ni Ax = b \in \mathbb{R}^n$$

Scientific Computing

The Multilevel Framework



This Talk: Refinement Phase

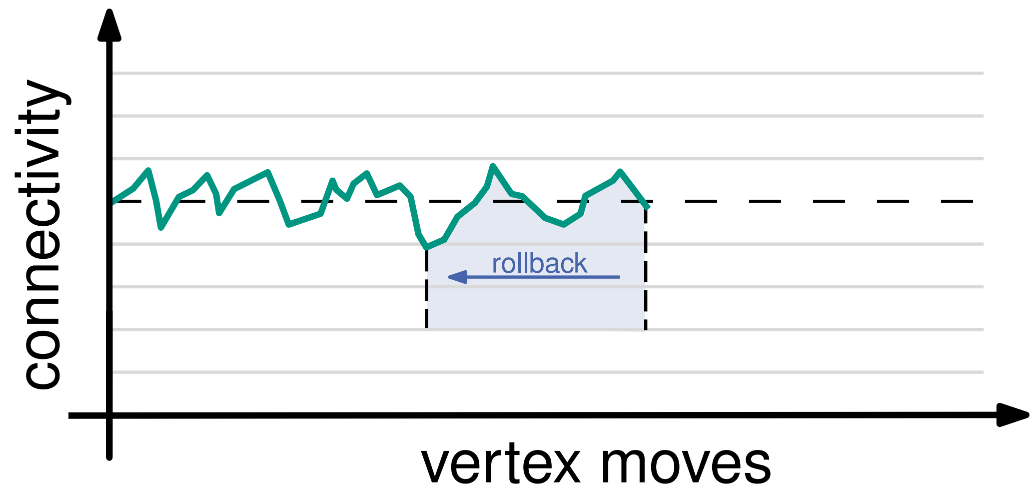


State-of-the-Art Multilevel HGP Refinement

Algorithm 1: FM Local Search

while *improvement found* **do**
  while ¬ *done* **do**
    `find` best move
    `perform` best move
  `rollback` to best solution

pass

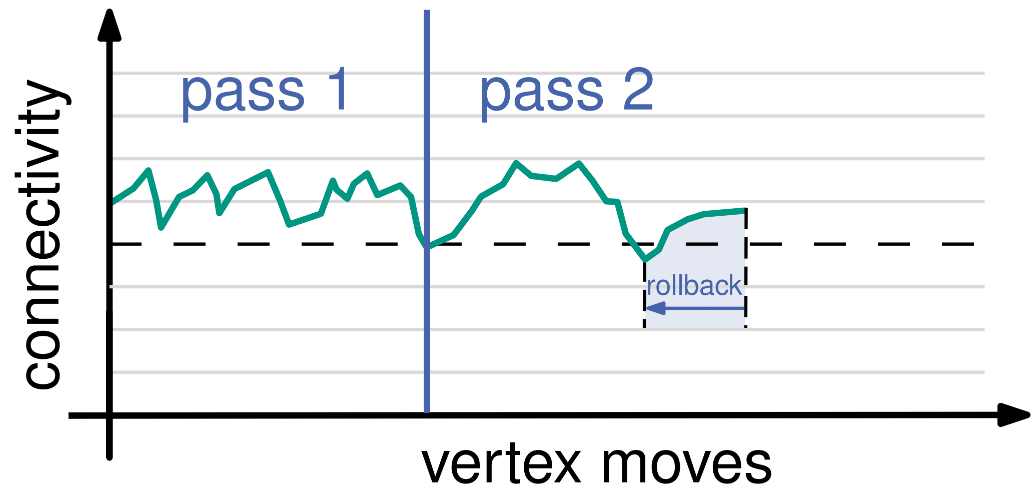


State-of-the-Art Multilevel HGP Refinement

Algorithm 1: FM Local Search

while *improvement found* **do**
  while ¬ *done* **do**
    `find` best move
    `perform` best move
  `rollback` to best solution

pass

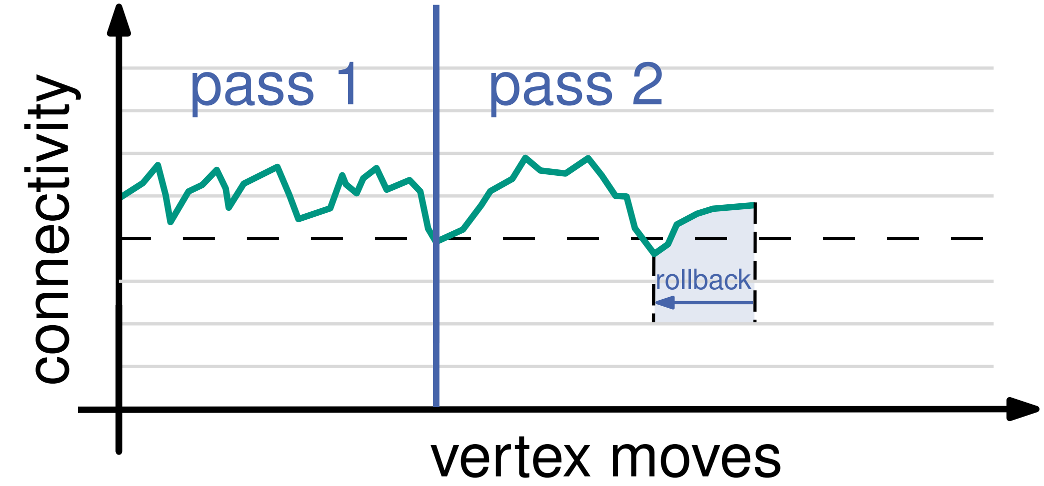


State-of-the-Art Multilevel HGP Refinement

Algorithm 1: FM Local Search

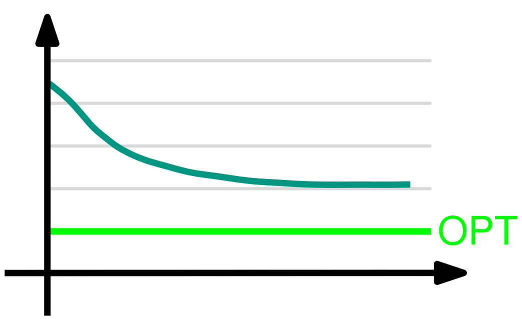
while *improvement found* **do**
  while ¬ *done* **do**
    `find` best move
    `perform` best move
  `rollback` to best solution

pass

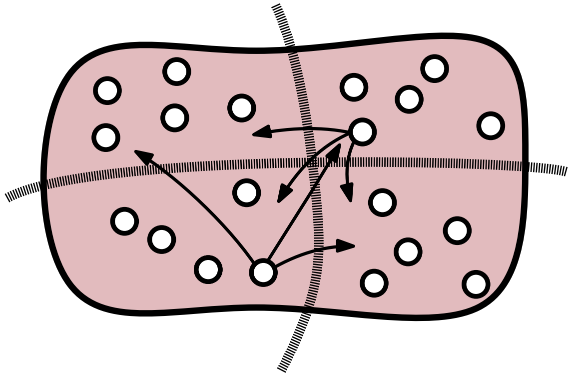


Known Limitations:

✗ prone to get **stuck** in local optima



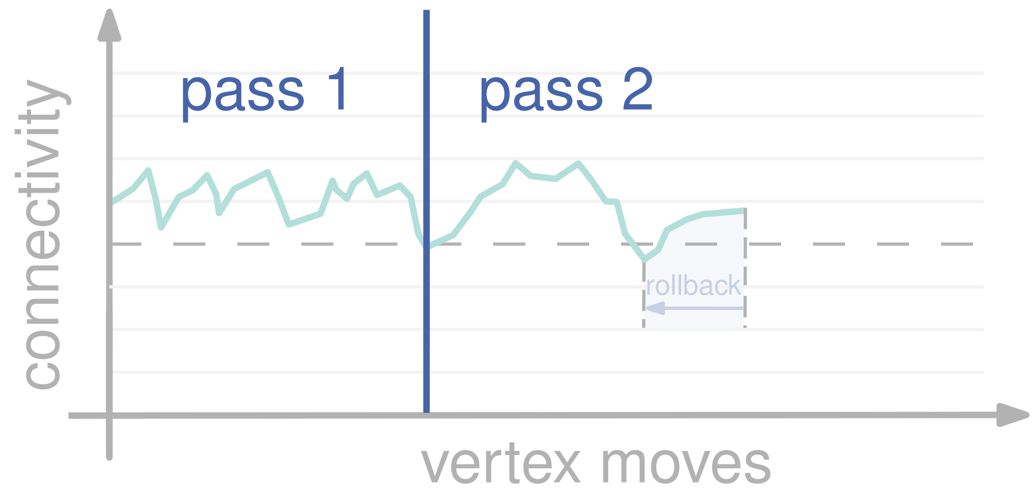
✗ large nets ⇝ **zero** gain moves



State-of-the-Art Multilevel HGP Refinement

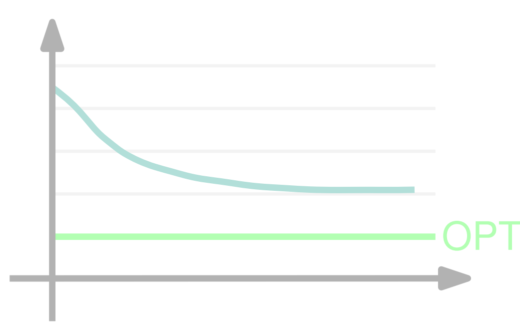
Algorithm 1: FM Local Search

```
while improvement found do
    while ¬ done do
        find best move
        perform best move                 pass
    rollback to best solution
```

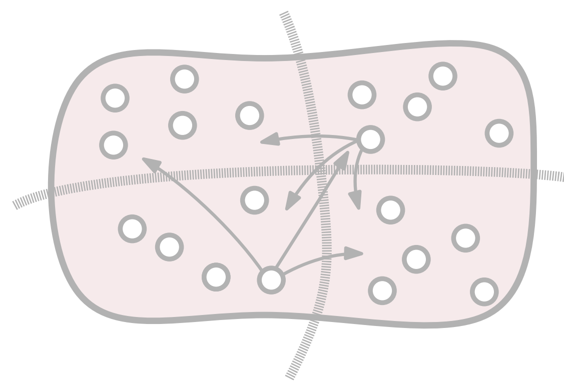


Are there viable alternatives?

✗ prone to get **stuck** in local optima



✗ large nets ⇝ **zero** gain moves

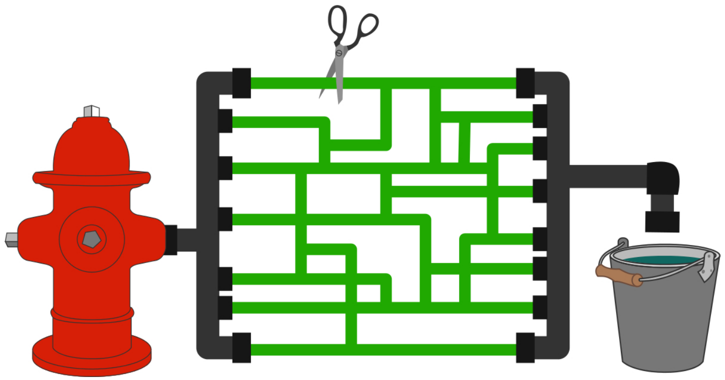


Flow-Based Refinement for Graph Partitioning

Goal: balanced partition with minimum cut

makes the problem **hard**!

Goal: ~~balanced~~ partition with minimum cut

$\Rightarrow$ 

https://brilliant.org/wiki/max-flow-min-cut-algorithm/

network flows

+

max-flow min-cut theorem

$\rightsquigarrow$

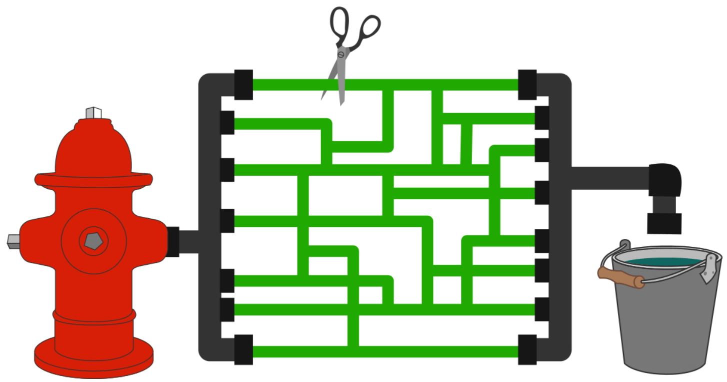
min. (s, t)-cuts

Flow-Based Refinement for Graph Partitioning

Karlsruhe Institute of Technology

Goal: ~~balanced~~ partition with minimum cut

$\Rightarrow$



https://brilliant.org/wiki/max-flow-min-cut-algorithm/

network flows

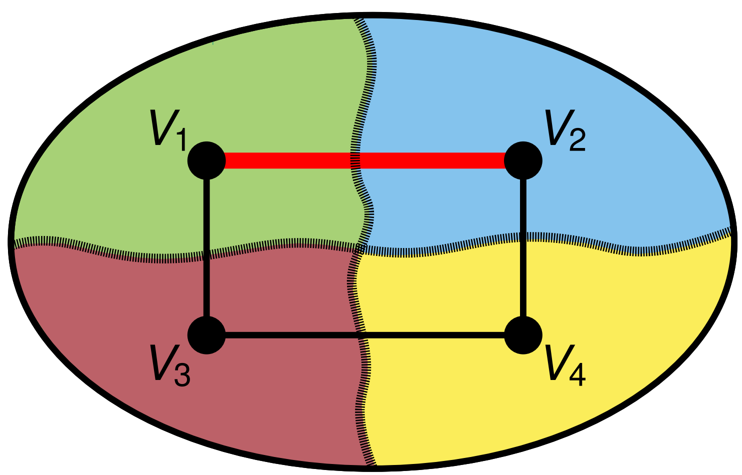
+

max-flow min-cut theorem

$\rightsquigarrow$

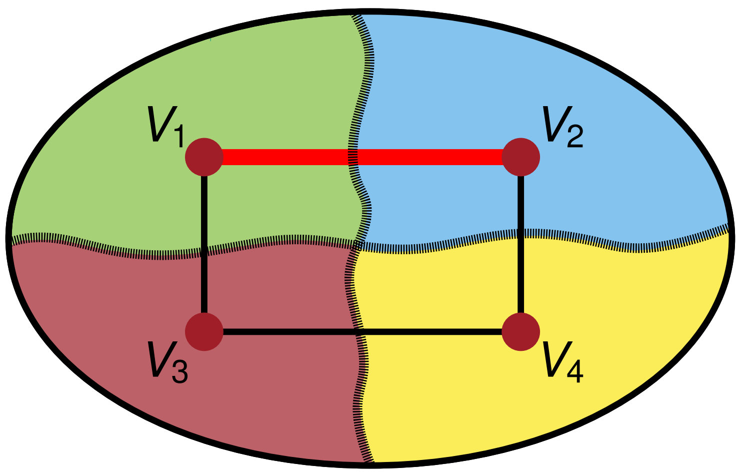
min. (s, t)-cuts

$\Rightarrow$ employed for graph partitioning in **KaFFPa** [Sanders, Schulz 11]

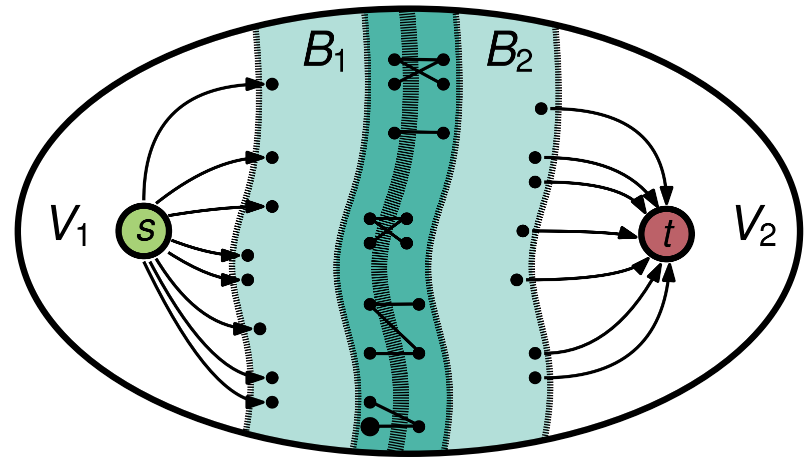


k-way refinement via **pairwise** flow-based improvements

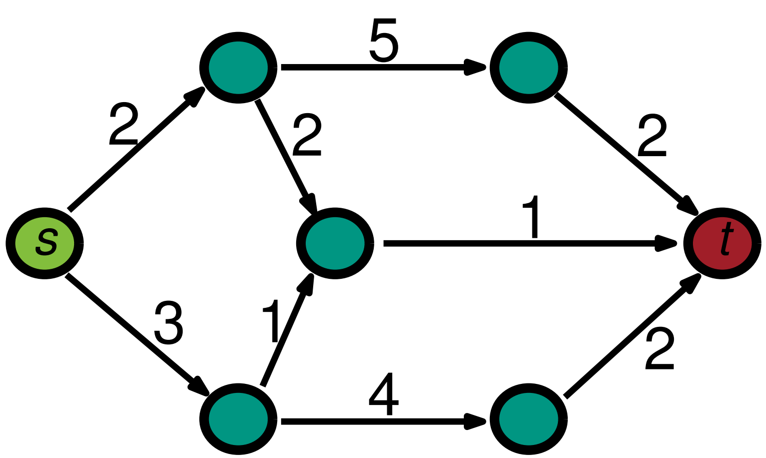
The KaFFPa Framework [Sanders, Schulz 11]



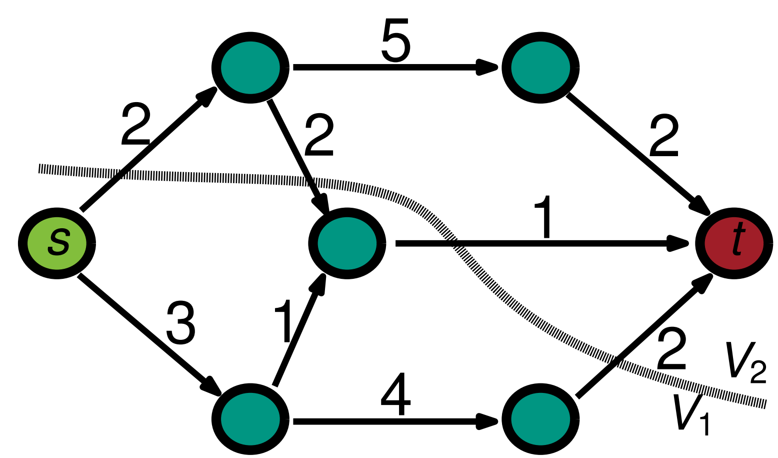
select two adjacent blocks for refinement



build flow network

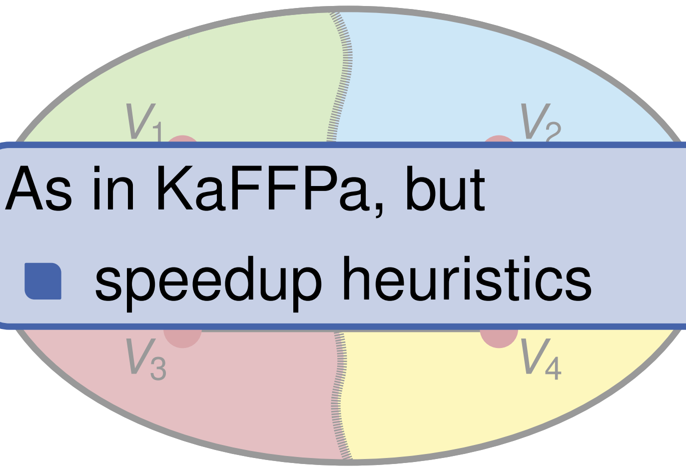


solve flow problem



find most-balanced minimum cut

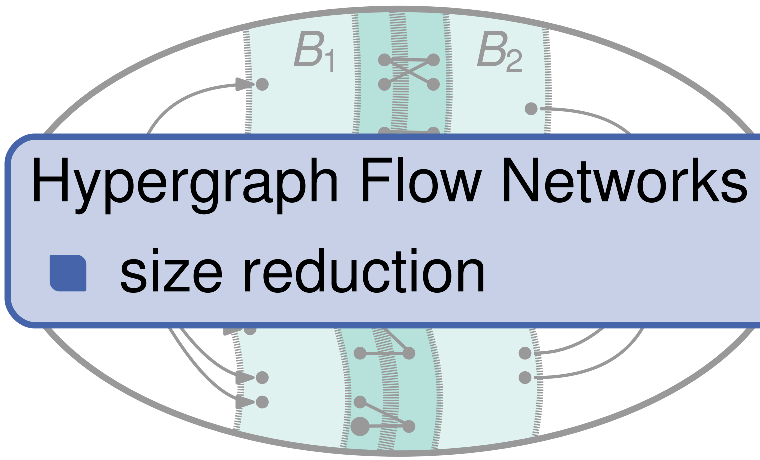
Our Refinement Framework/ Contributions



As in KaFFPa, but

- speedup heuristics

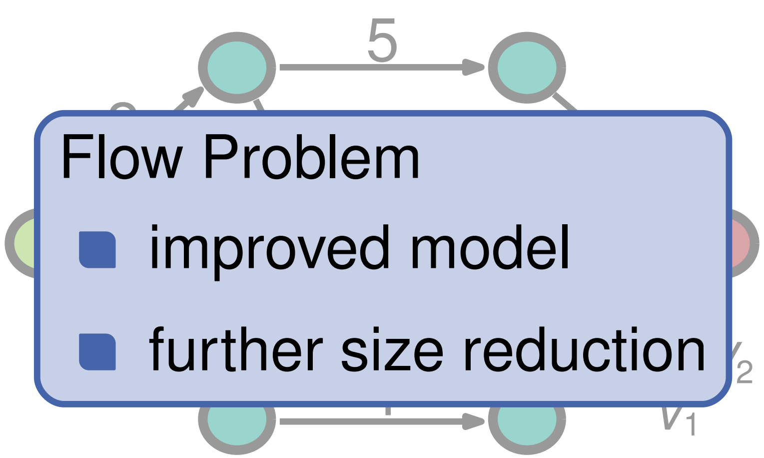
select two adjacent blocks for refinement



Hypergraph Flow Networks

- size reduction

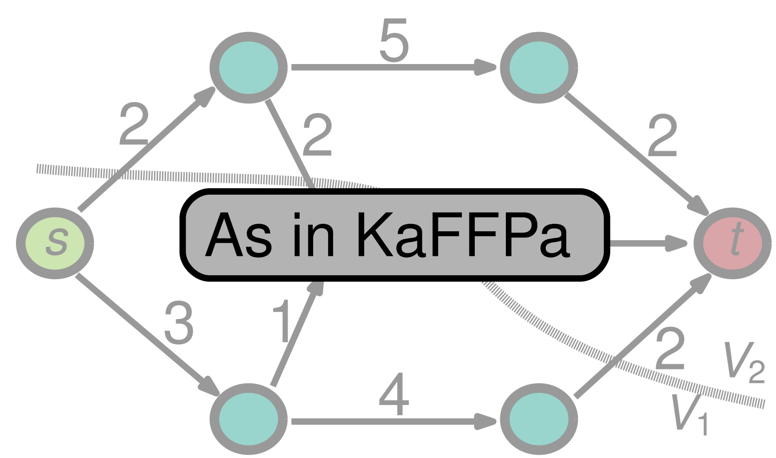
build flow network



Flow Problem

- improved model
- further size reduction

solve flow problem



As in KaFFPa

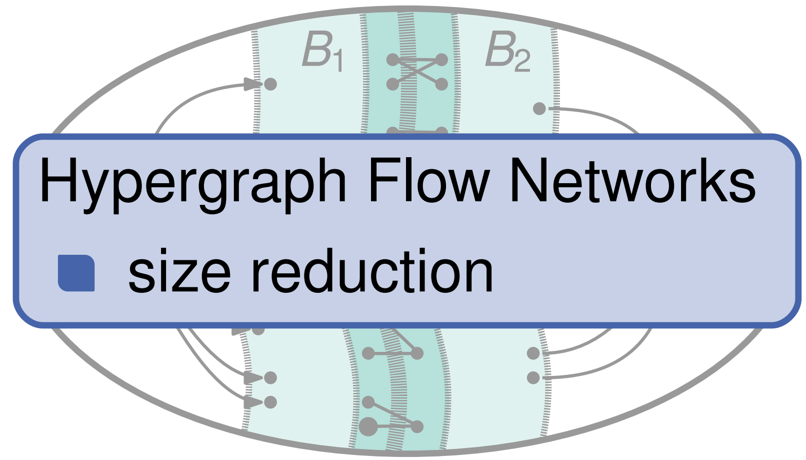
find most-balanced minimum cut

I am going to talk about...

As in KaFFPa, but

- speedup heuristics

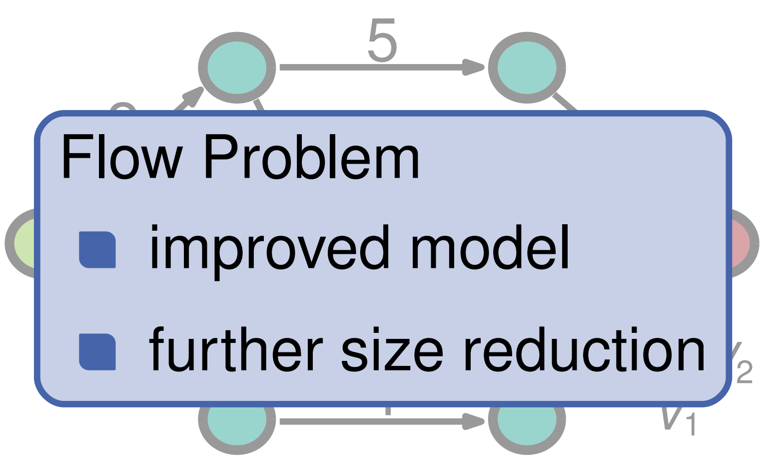
select two adjacent blocks for refinement



Hypergraph Flow Networks

- size reduction

build flow network



Flow Problem

- improved model
- further size reduction

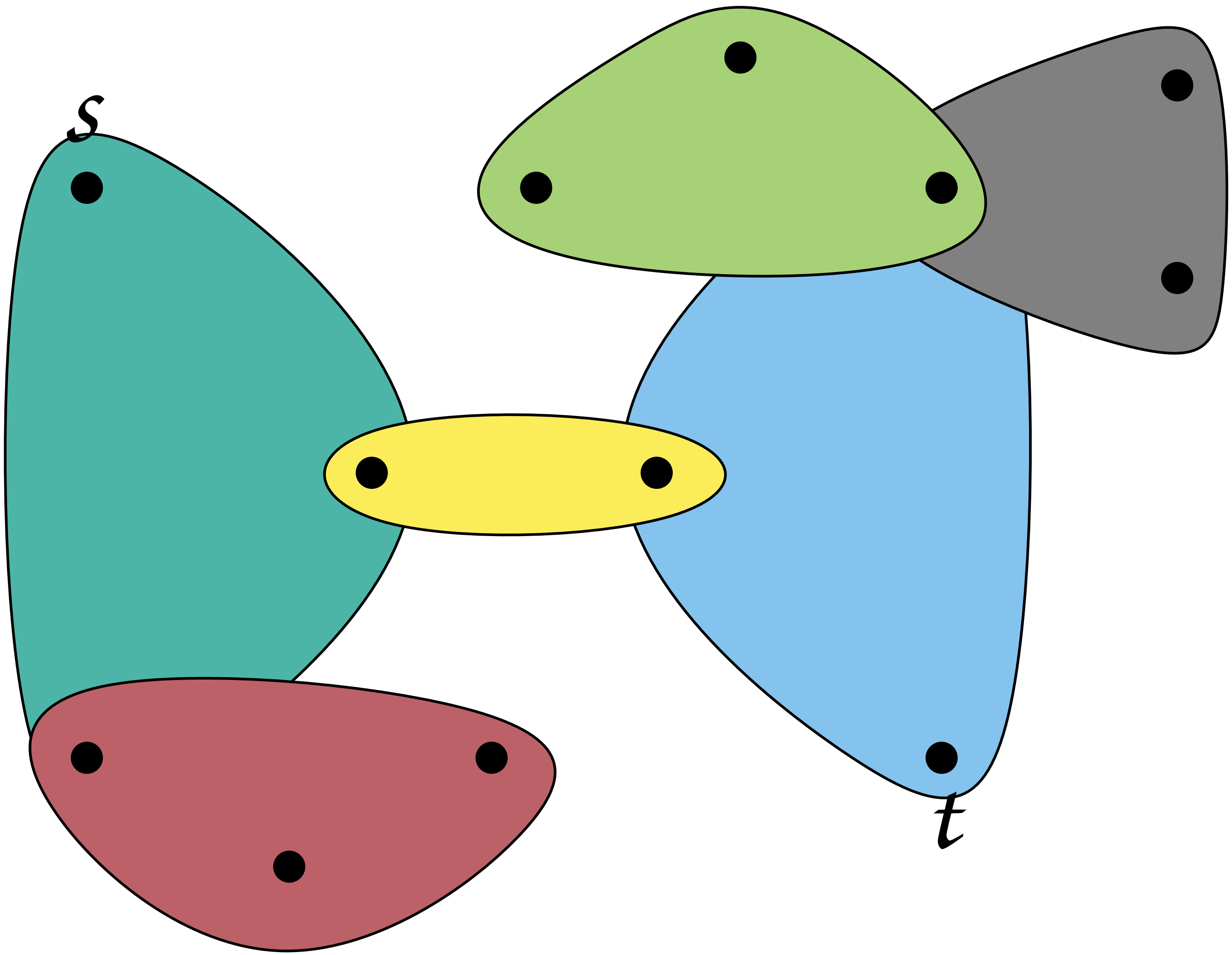
solve flow problem

As in KaFFPa

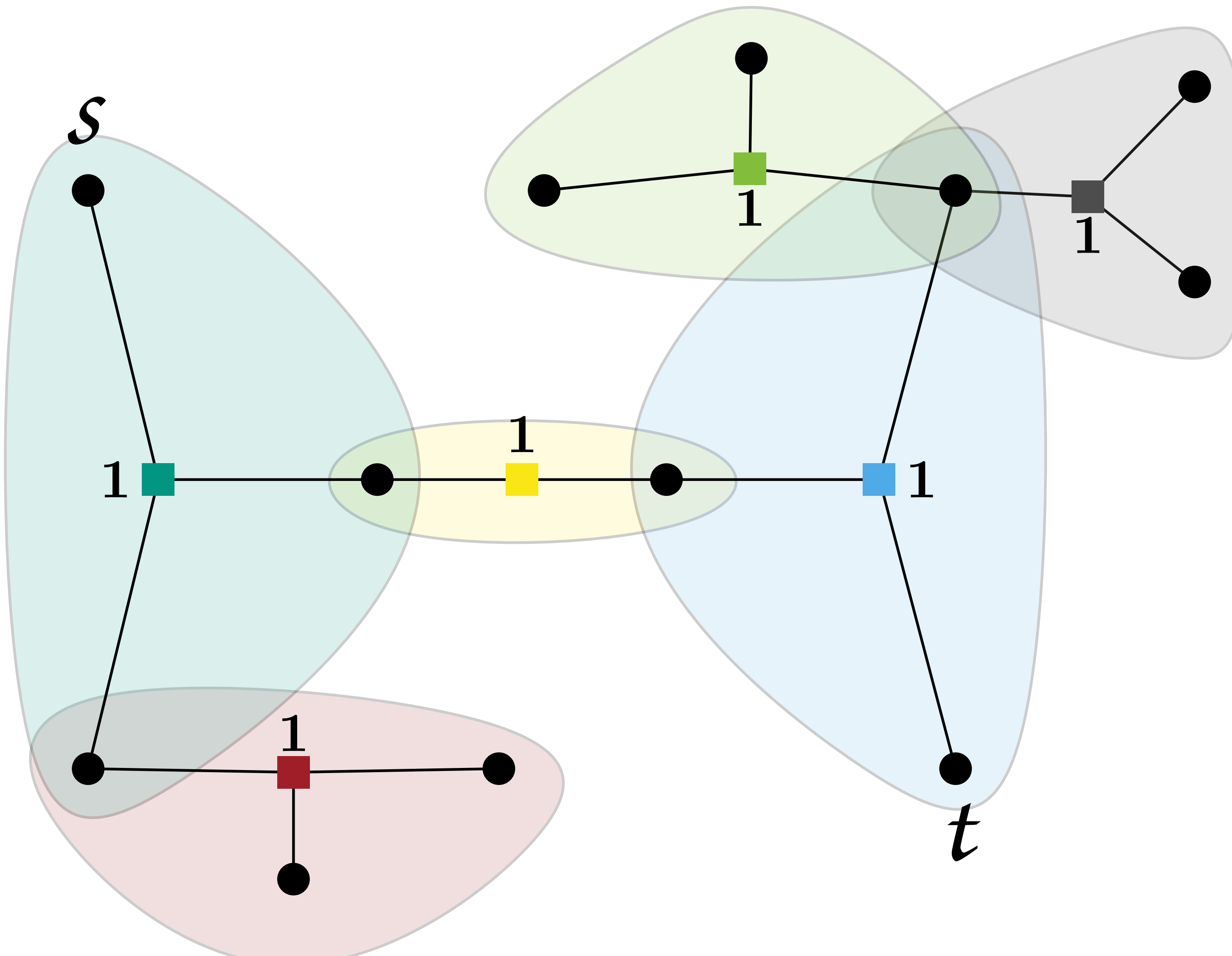
find most-balanced minimum cut

Hypergraph Flow Networks

Hypergraph H

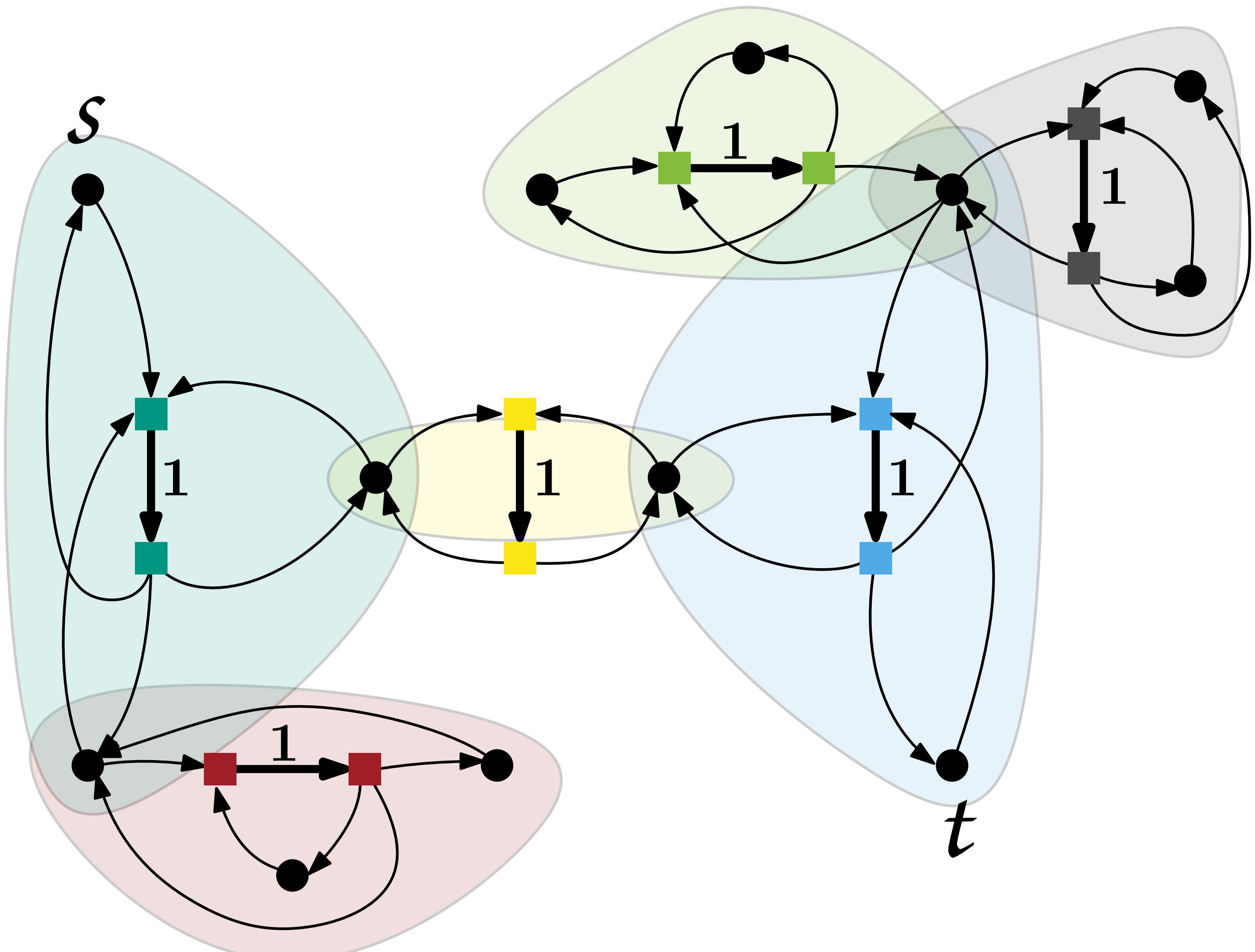


Hypergraph Flow Networks: Star-Expansion G^*



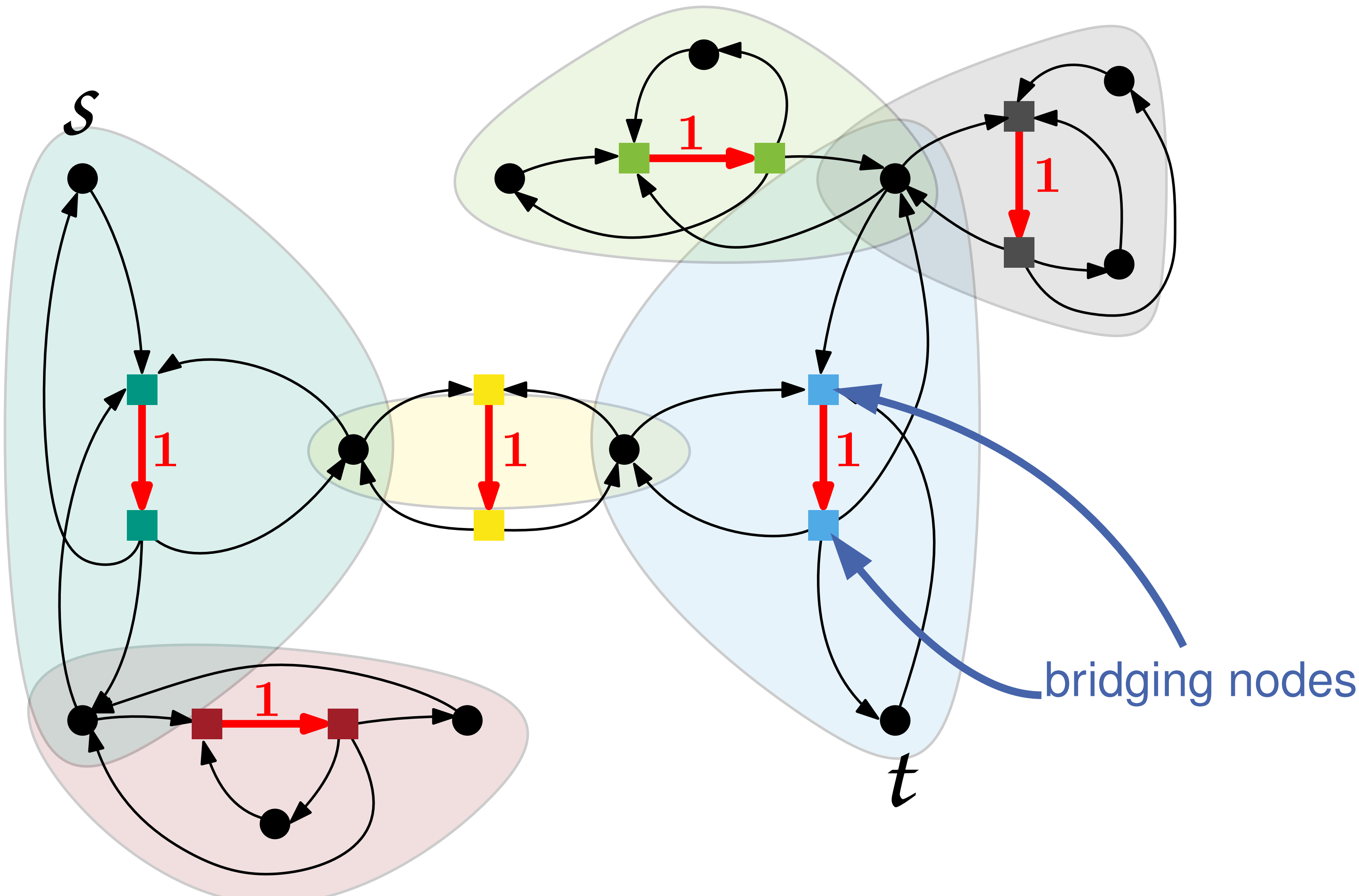
Hypergraph Flow Networks: Lawler Network [Lawler 73]

⇒ node capacities ⇝ edge capacities



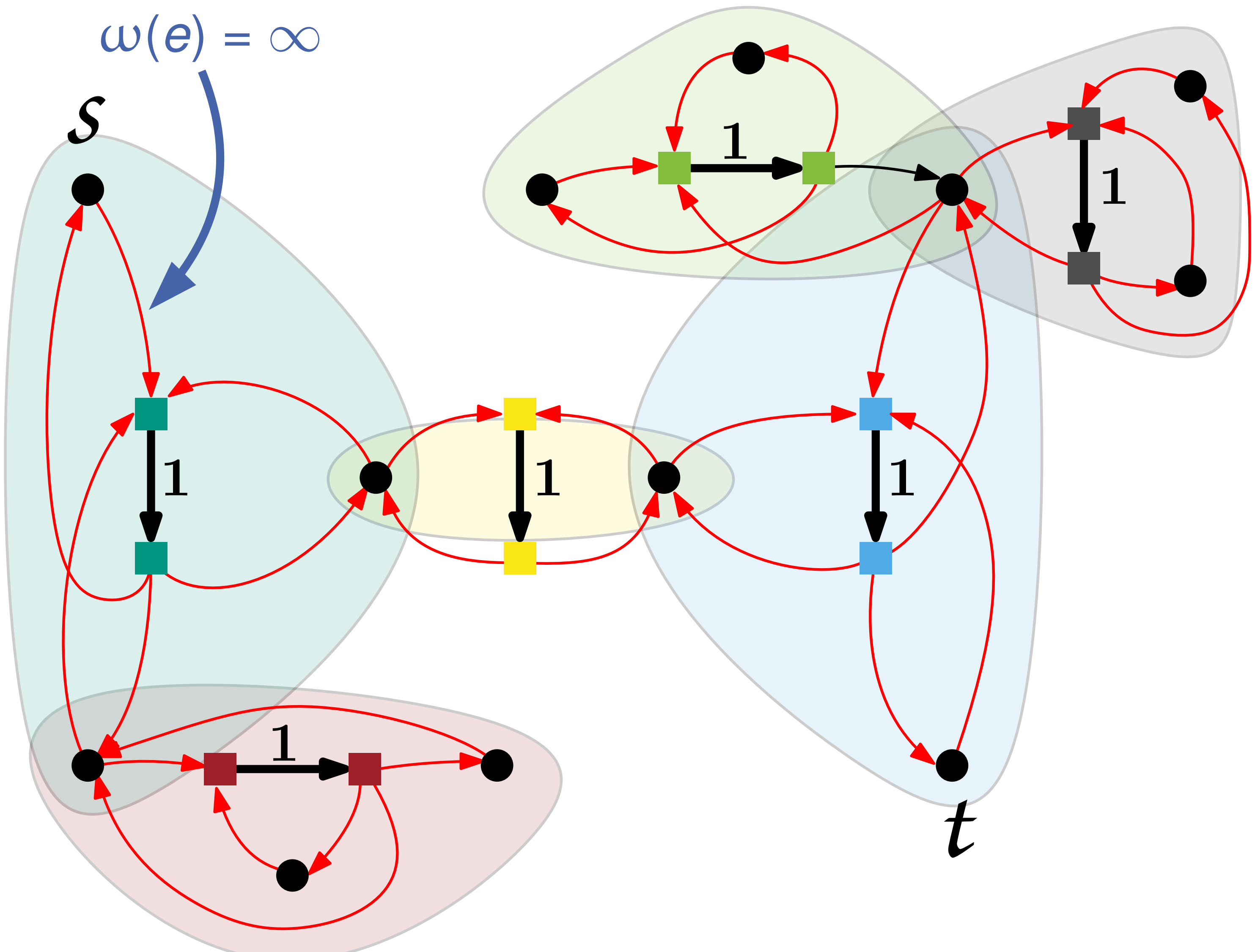
Hypergraph Flow Networks: Lawler Network [Lawler 73]

$\Rightarrow$ node capacities $\rightsquigarrow$ edge capacities



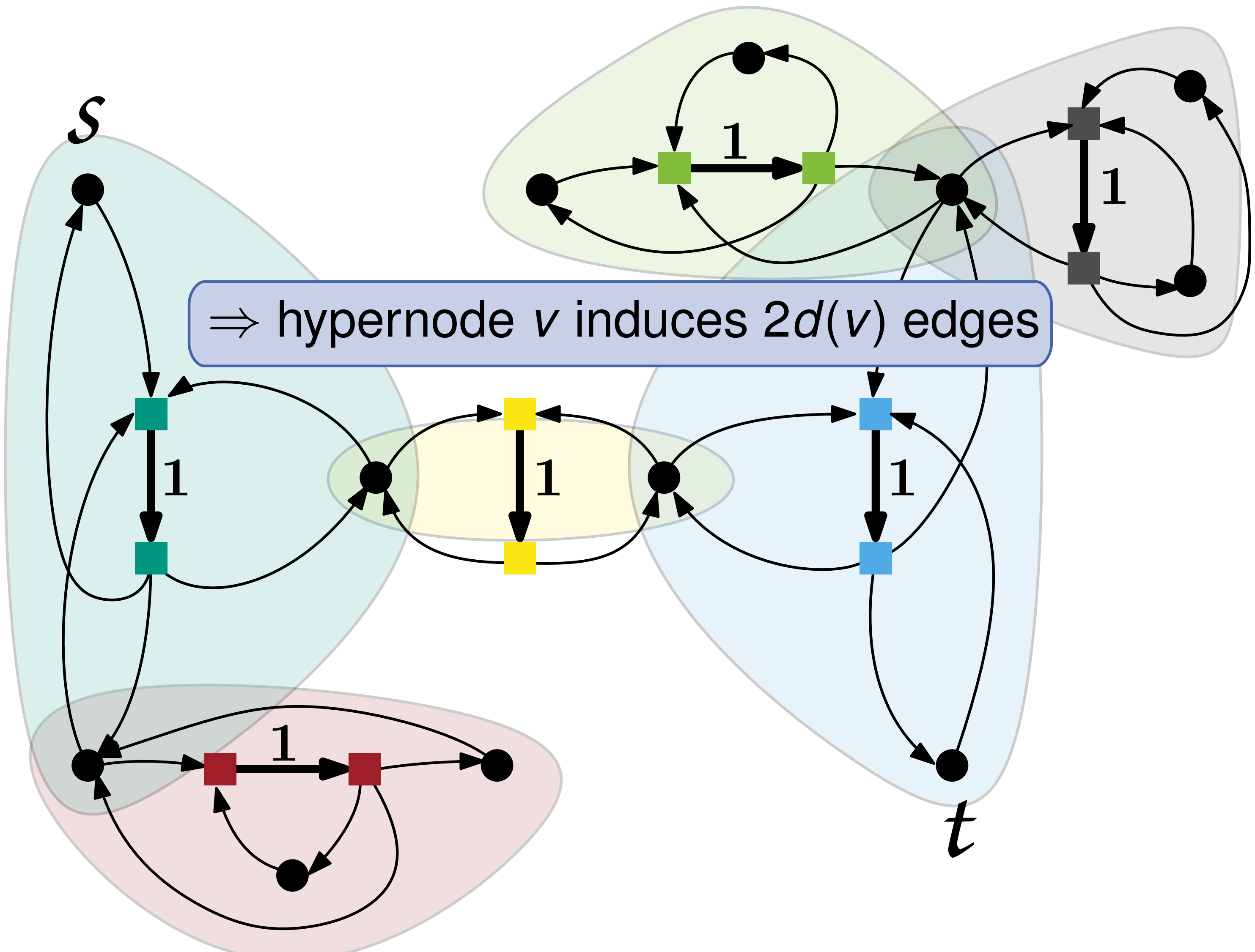
Hypergraph Flow Networks: Lawler Network [Lawler 73]

$\Rightarrow$ node capacities $\rightsquigarrow$ edge capacities



Hypergraph Flow Networks: Lawler Network [Lawler 73]

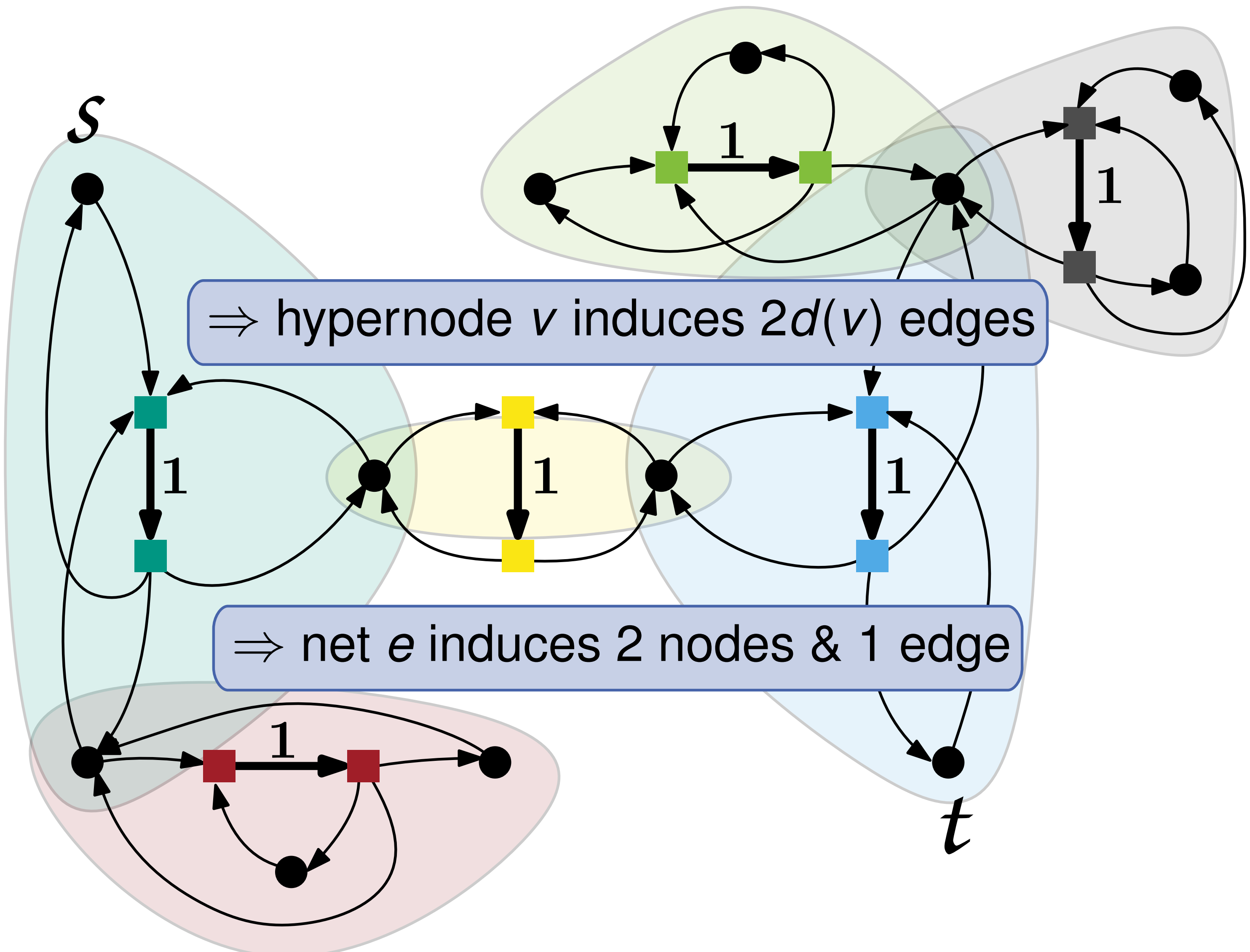
⇒ node capacities ⇝ edge capacities



⇒ hypernode v induces $2d(v)$ edges

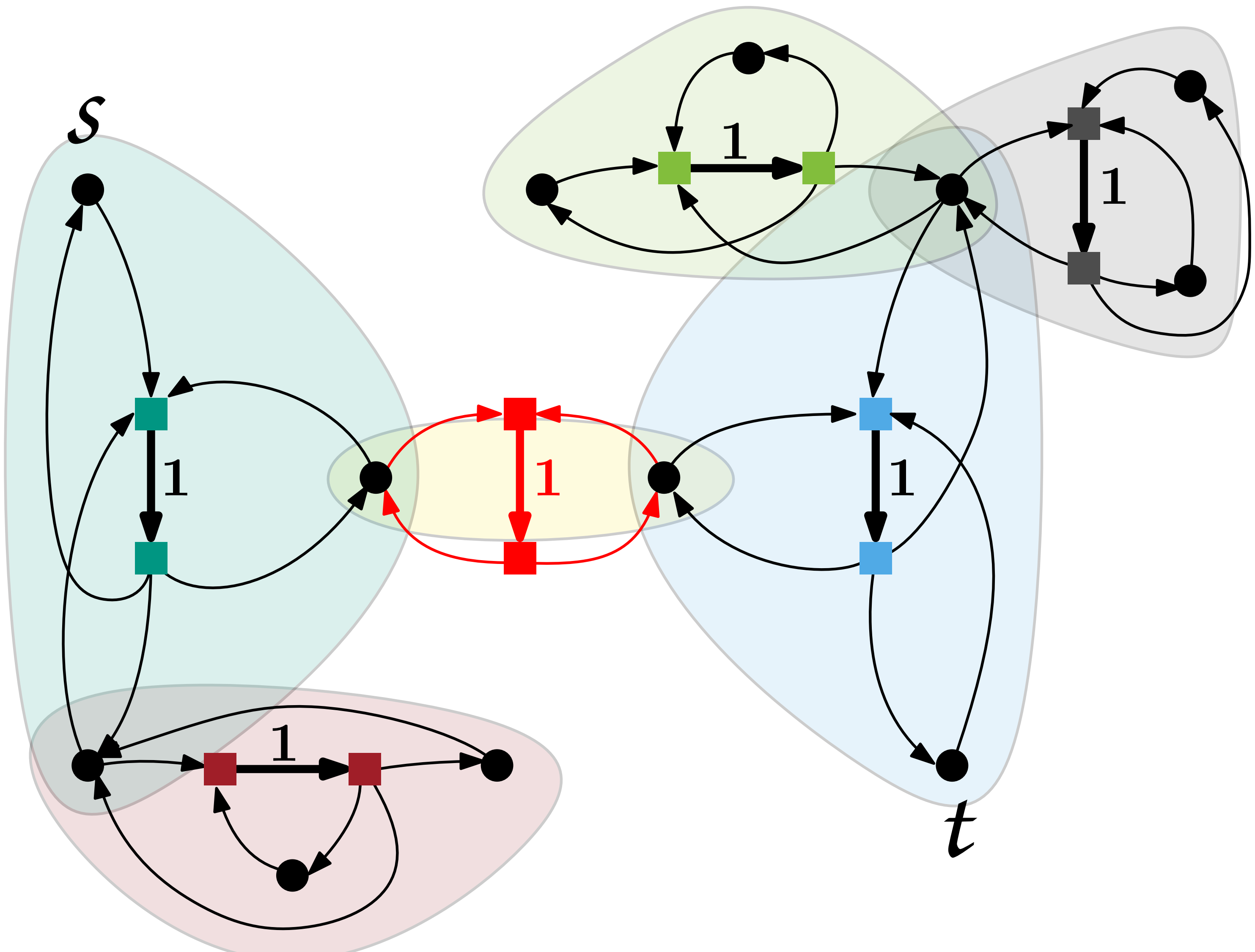
Hypergraph Flow Networks: Lawler Network [Lawler 73]

⇒ node capacities ⇝ edge capacities



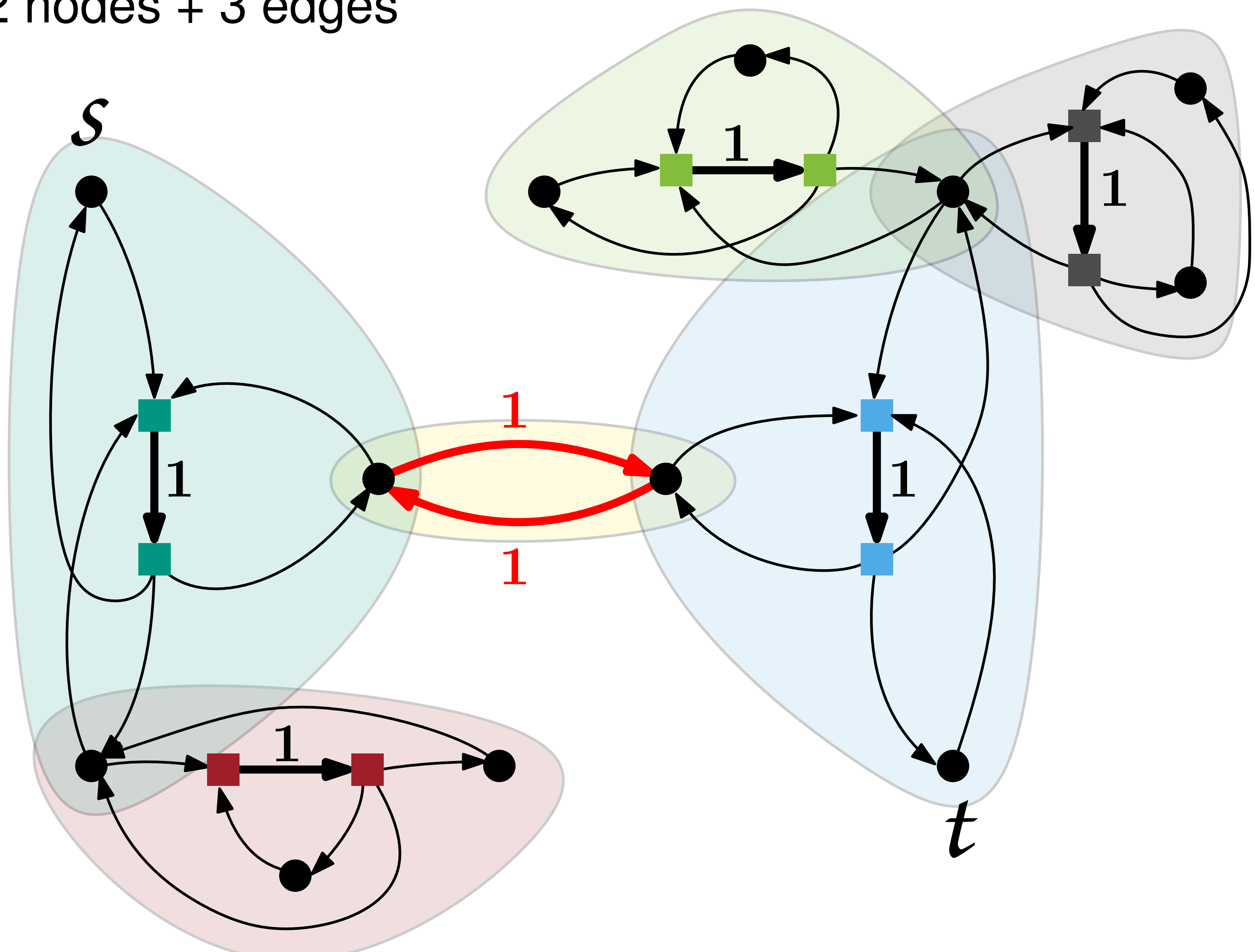
Hypergraph Flow Networks: Lawler Network [Lawler 73]

⇒ node capacities ⇝ edge capacities



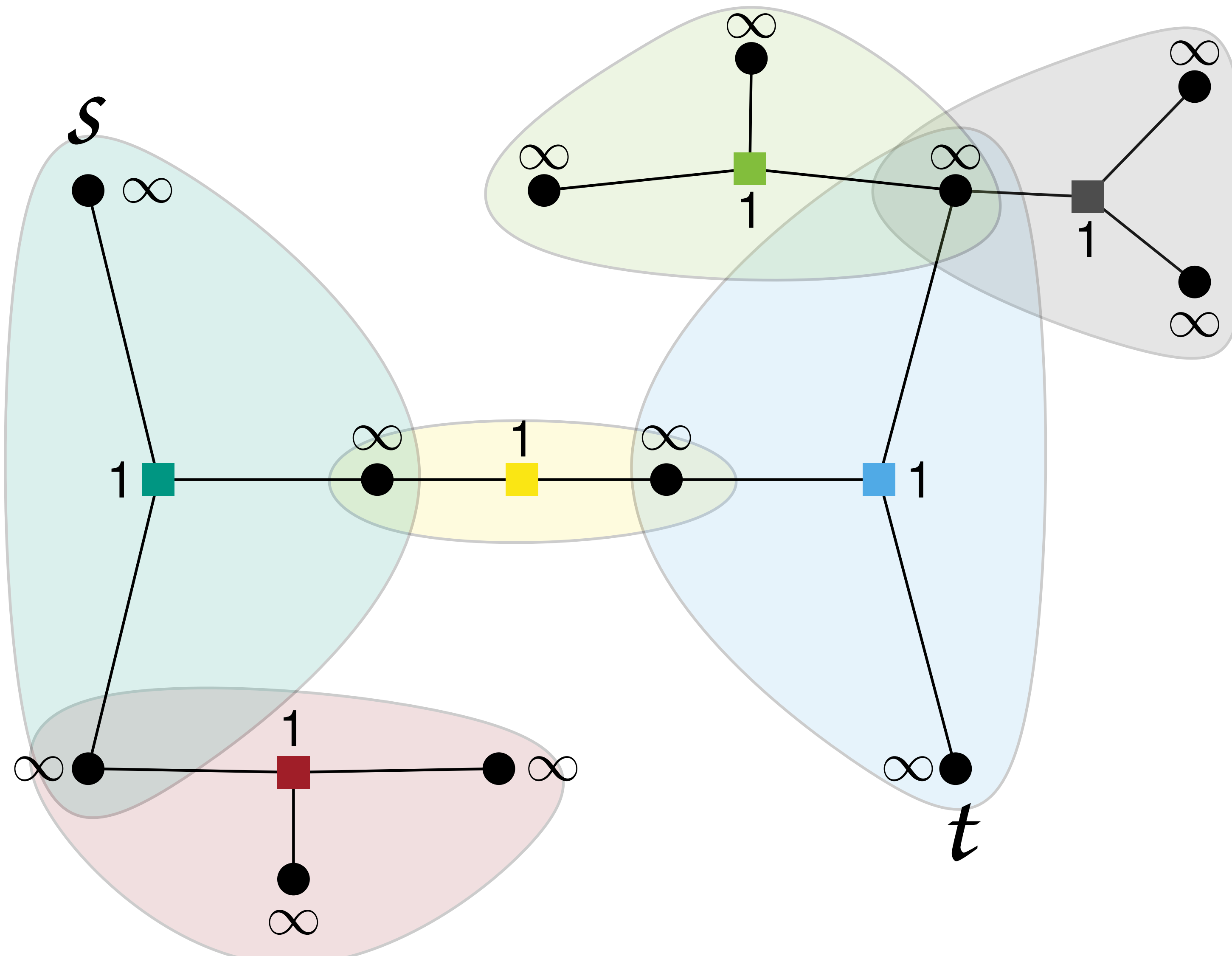
Hypergraph Flow Networks: Liu-Wong Network [LW98]

special treatment of **two-pin** nets
⇒ save 2 nodes + 3 edges



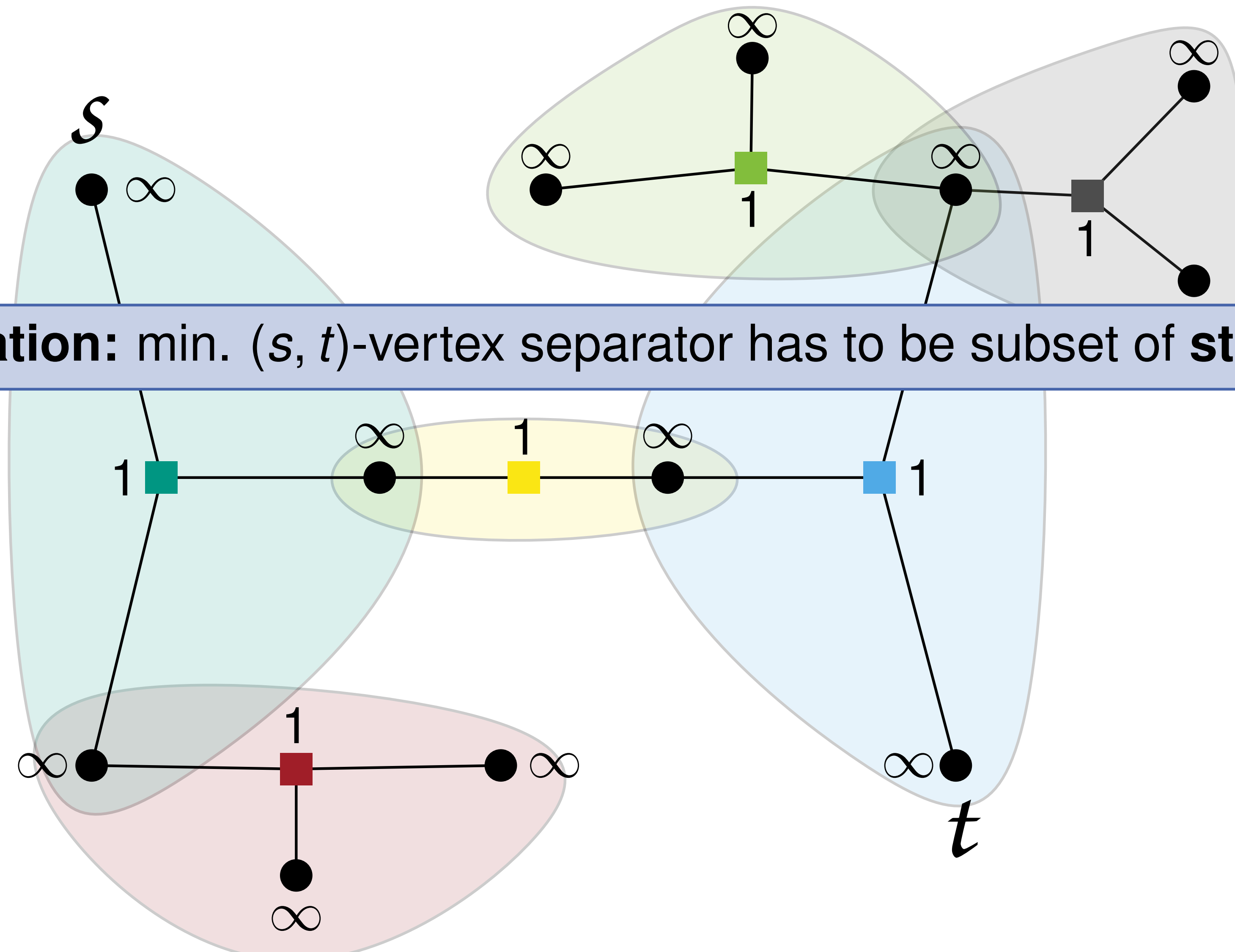
Hypergraph Flow Networks: Our Network

Minimum-Weight Vertex Separator [Hu, Moerder 85]



Hypergraph Flow Networks: Our Network

Minimum-Weight Vertex Separator [Hu, Moerder 85]

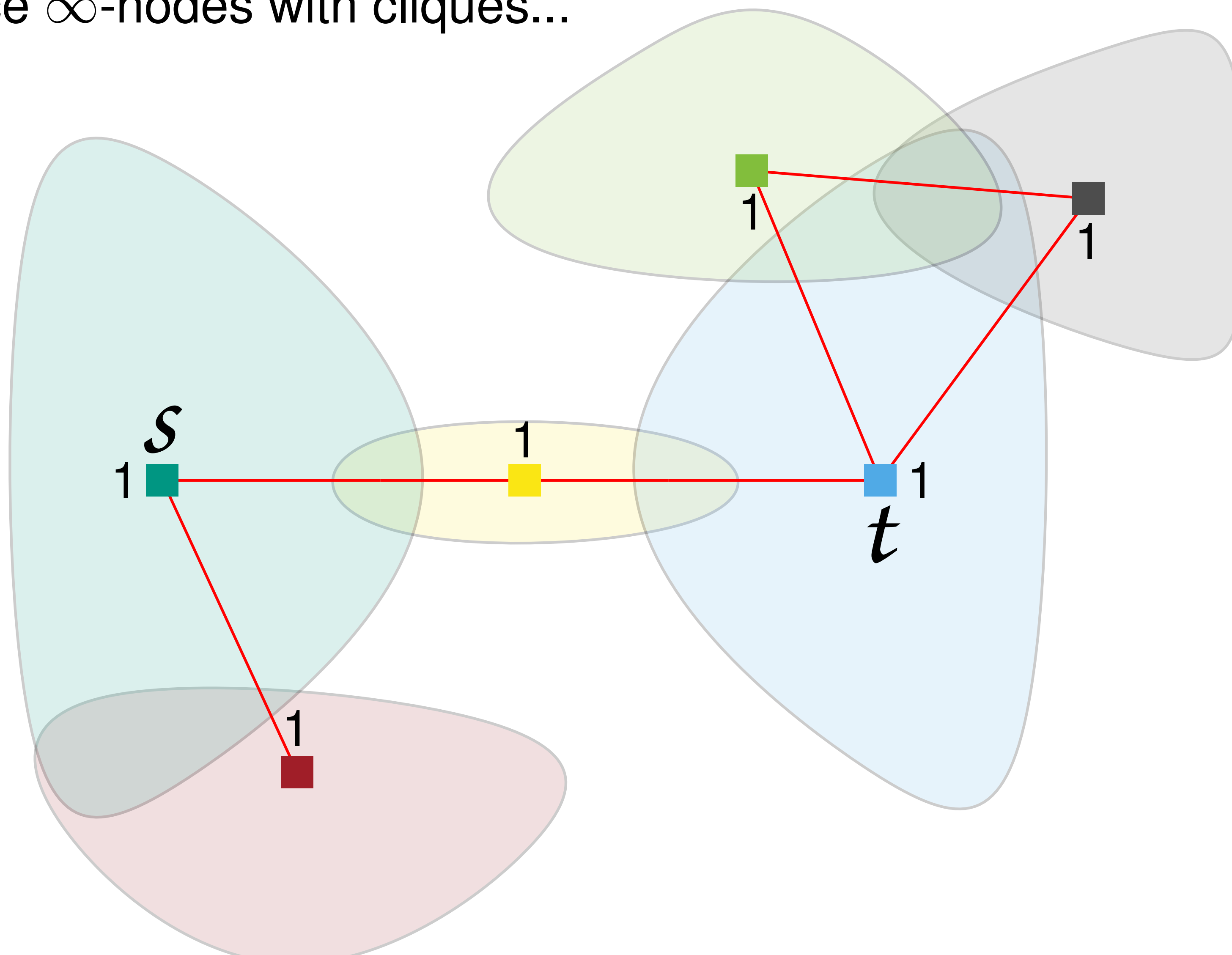


Observation: min. (s, t)-vertex separator has to be subset of **star**-nodes

Hypergraph Flow Networks: Our Network

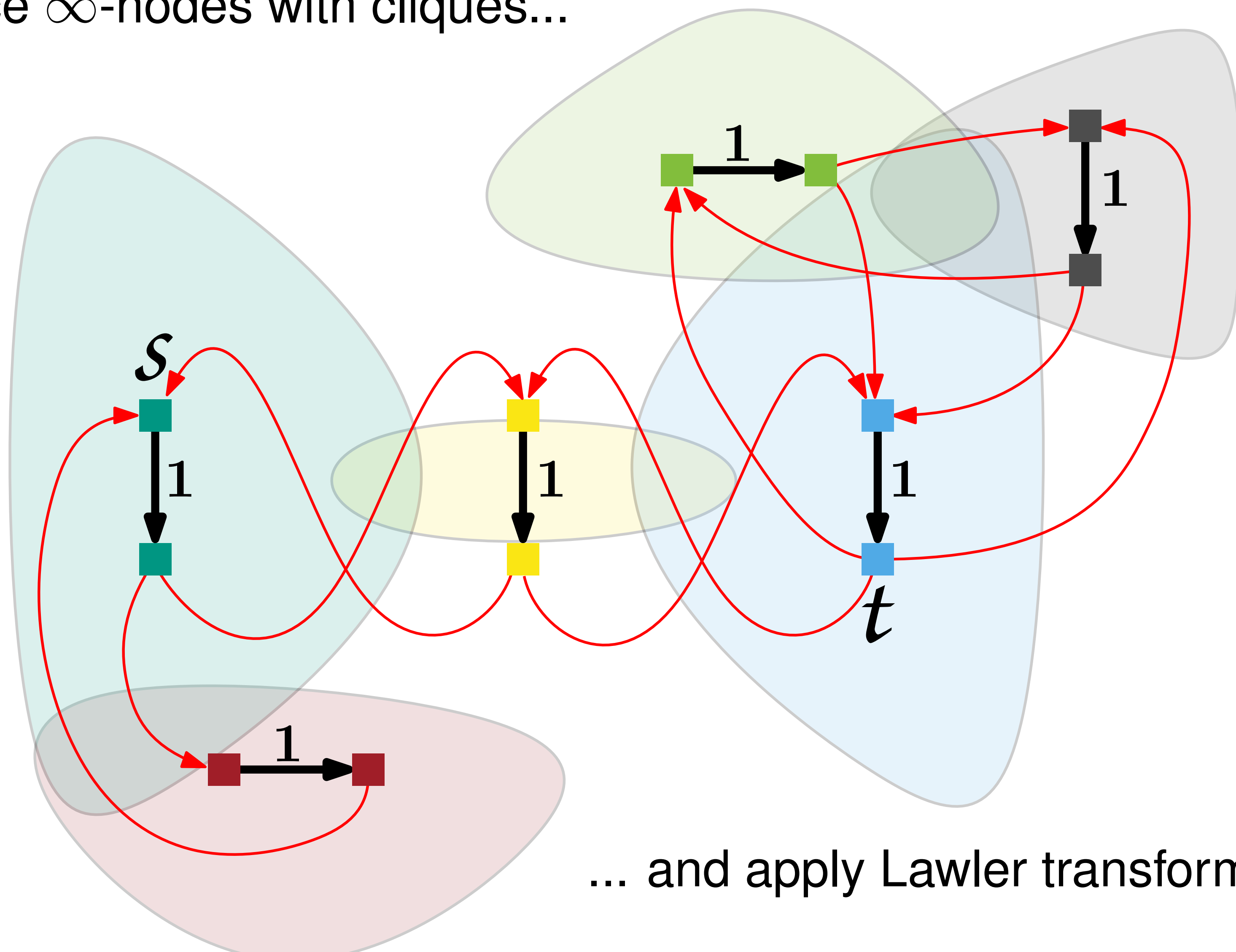
Minimum-Weight Vertex Separator [Hu, Moerder 85]

$\Rightarrow$ replace ∞-nodes with cliques...



Hypergraph Flow Networks: Our Network

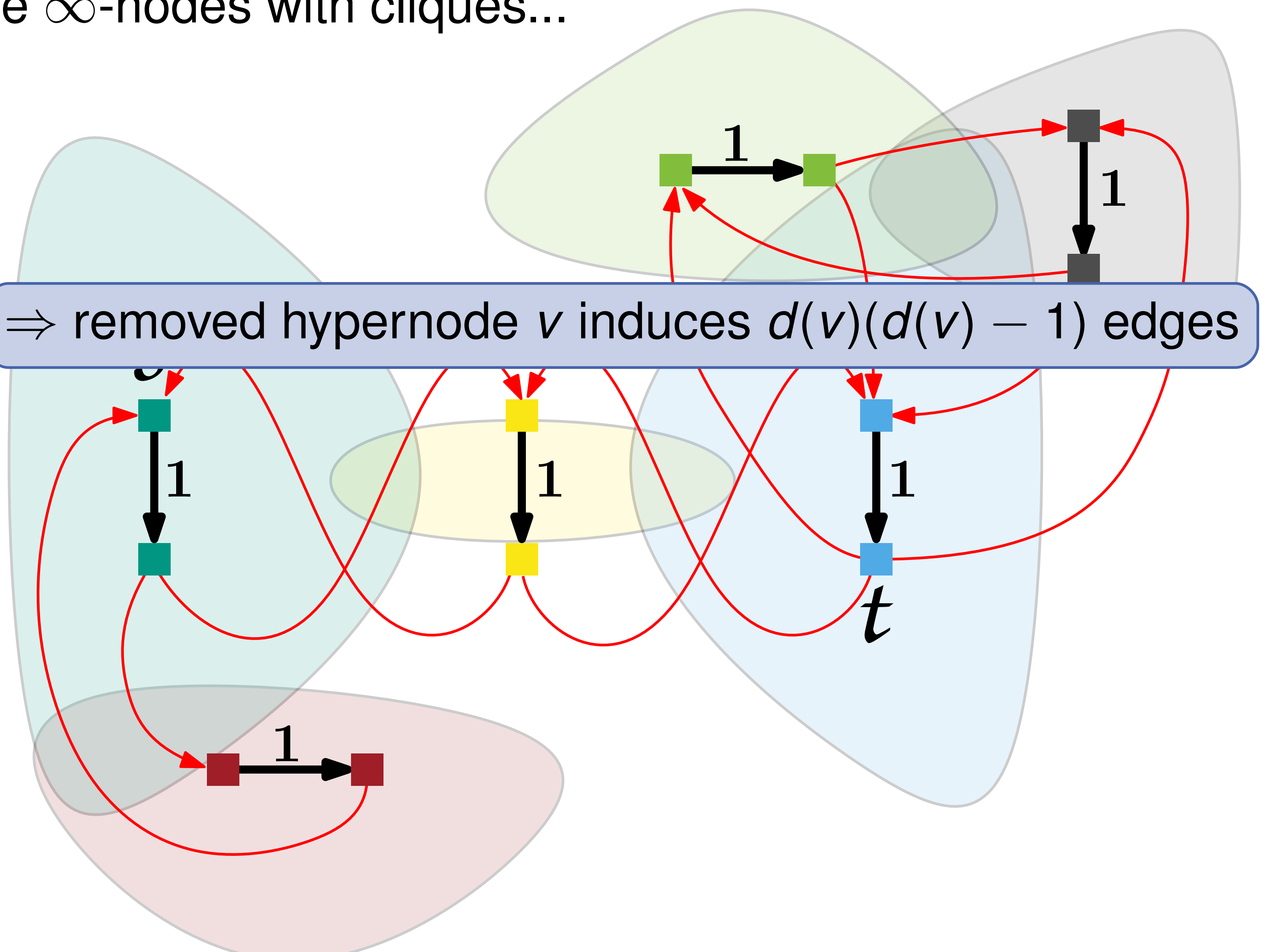
Minimum-Weight Vertex Separator [Hu, Moerder 85]
$\Rightarrow$ replace ∞-nodes with cliques...



... and apply Lawler transformation

Hypergraph Flow Networks: Our Network

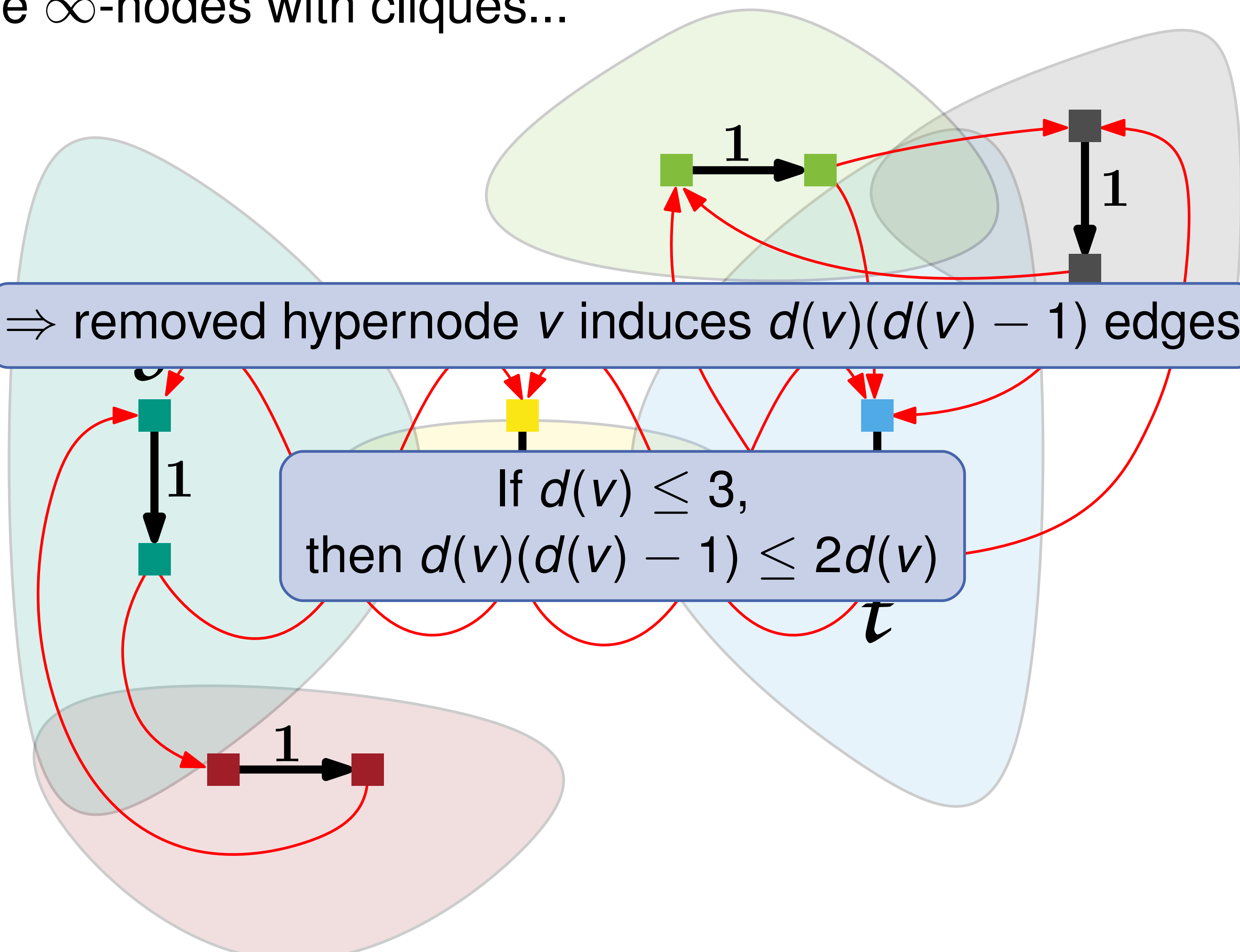
Minimum-Weight Vertex Separator [Hu, Moerder 85]
⇒ replace ∞-nodes with cliques...



⇒ removed hypernode v induces $d(v)(d(v) - 1)$ edges

Hypergraph Flow Networks: Our Network

Minimum-Weight Vertex Separator [Hu, Moerder 85]
$\Rightarrow$ replace ∞-nodes with cliques...

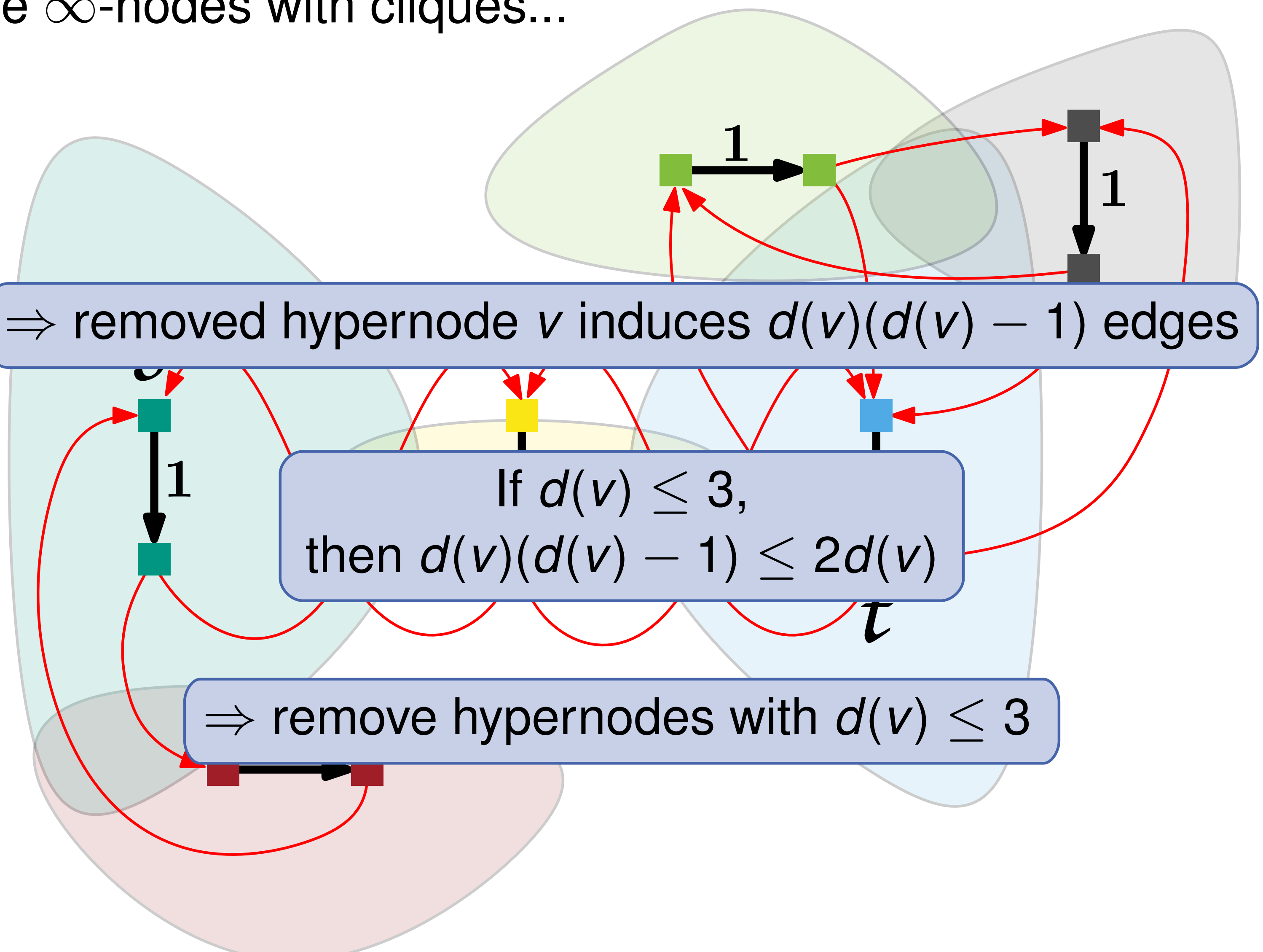


$\Rightarrow$ removed hypernode v induces $d(v)(d(v)-1)$ edges

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

Hypergraph Flow Networks: Our Network

Minimum-Weight Vertex Separator [Hu, Moerder 85]
$\Rightarrow$ replace ∞-nodes with cliques...



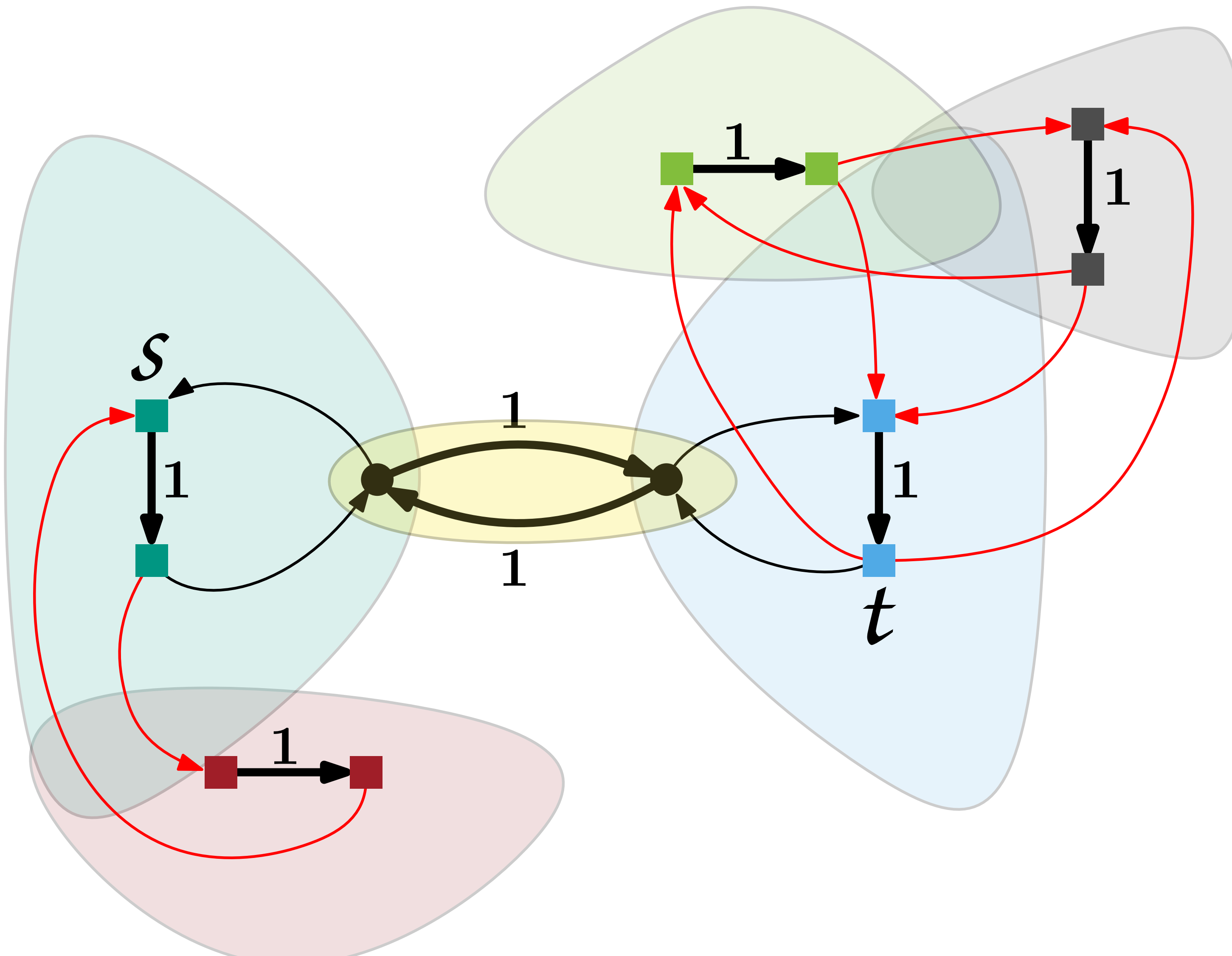
$\Rightarrow$ removed hypernode v induces $d(v)(d(v)-1)$ edges

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

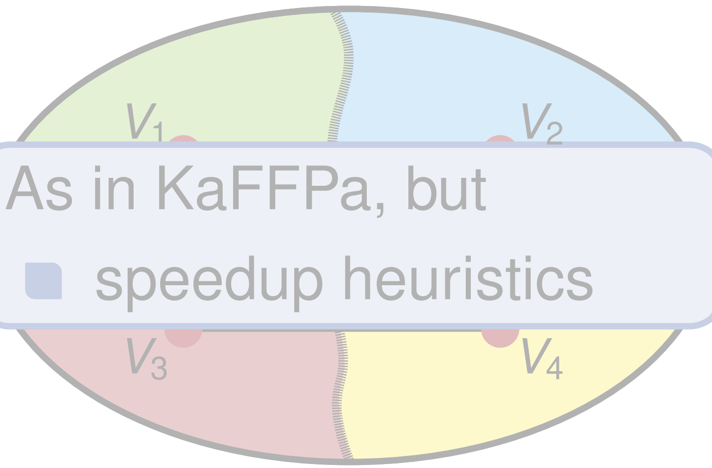
$\Rightarrow$ remove hypernodes with $d(v) \leq 3$

Hypergraph Flow Networks: Our Network

⇒ **combine** low degree hypernode removal with Liu-Wong transformation

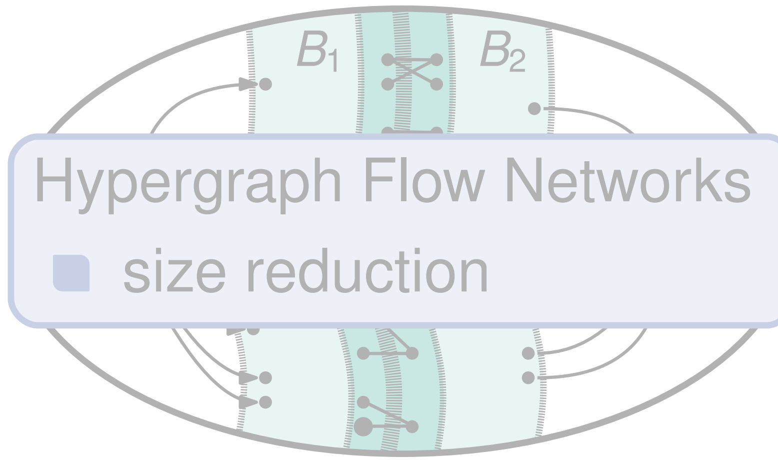


I am going to talk about ...



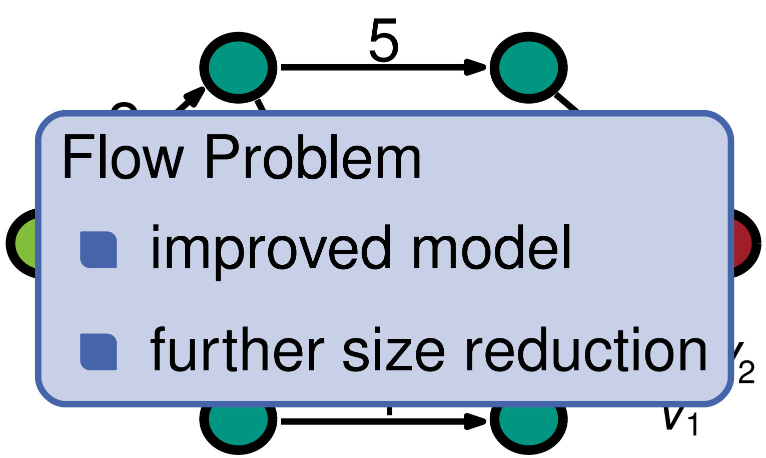
As in KaFFPa, but
- speedup heuristics

select two adjacent blocks for refinement



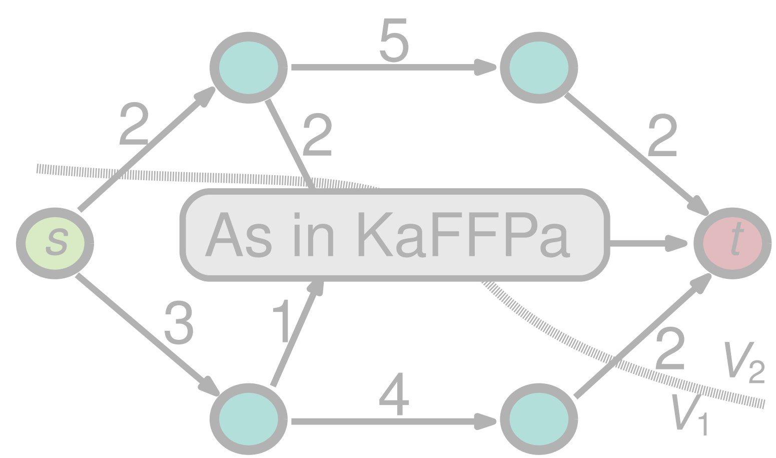
Hypergraph Flow Networks
- size reduction

build flow network



Flow Problem
- improved model
- further size reduction

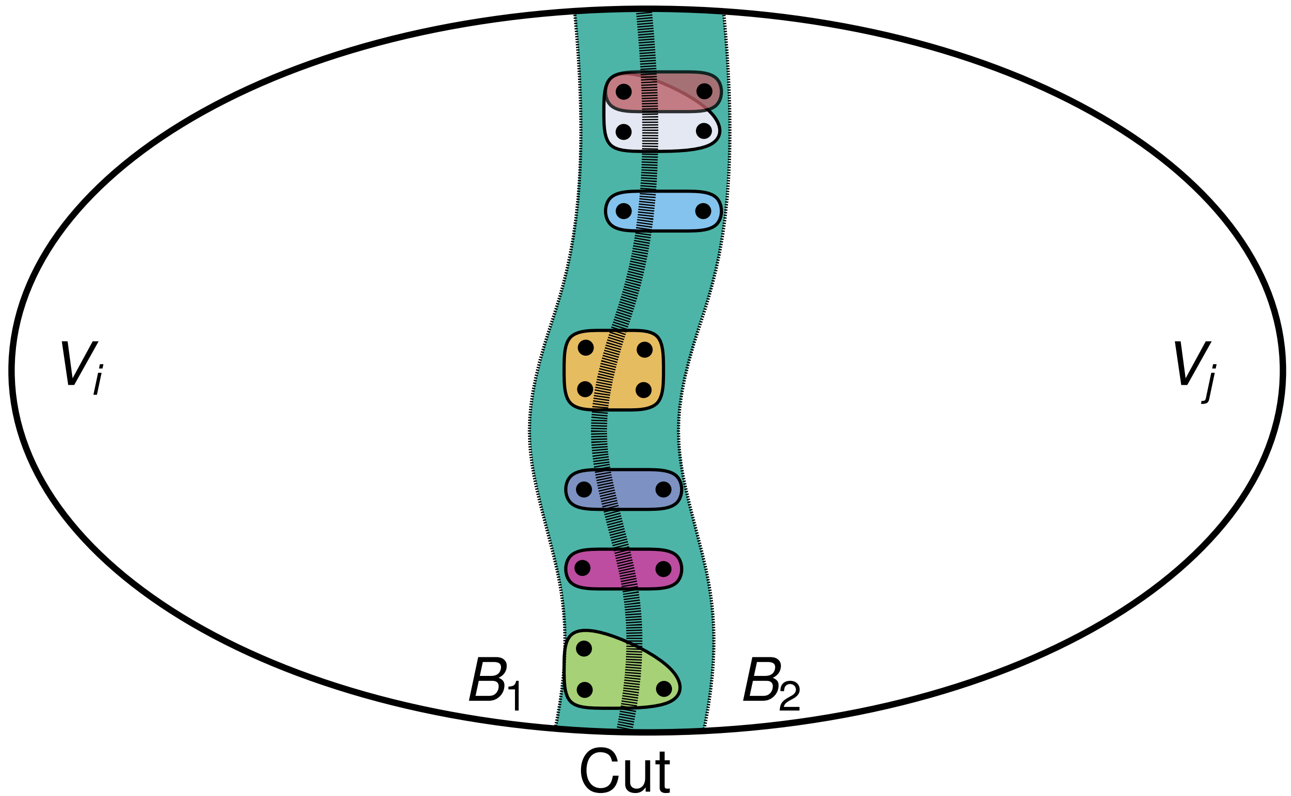
solve flow problem



As in KaFFPa

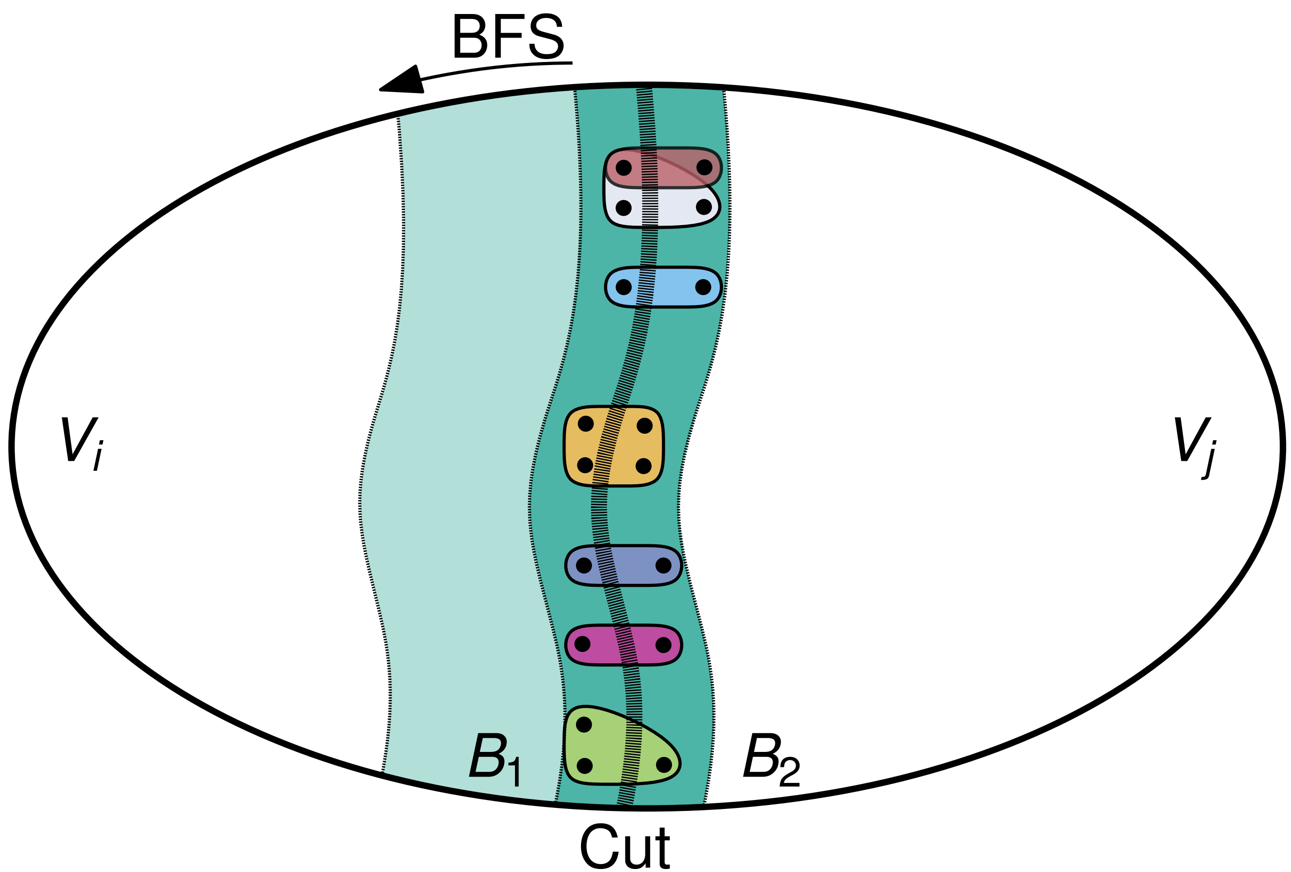
find most-balanced minimum cut

KaFFPa's Flow-Based Refinement for hypergraphs



KaFFPa's Flow-Based Refinement for hypergraphs

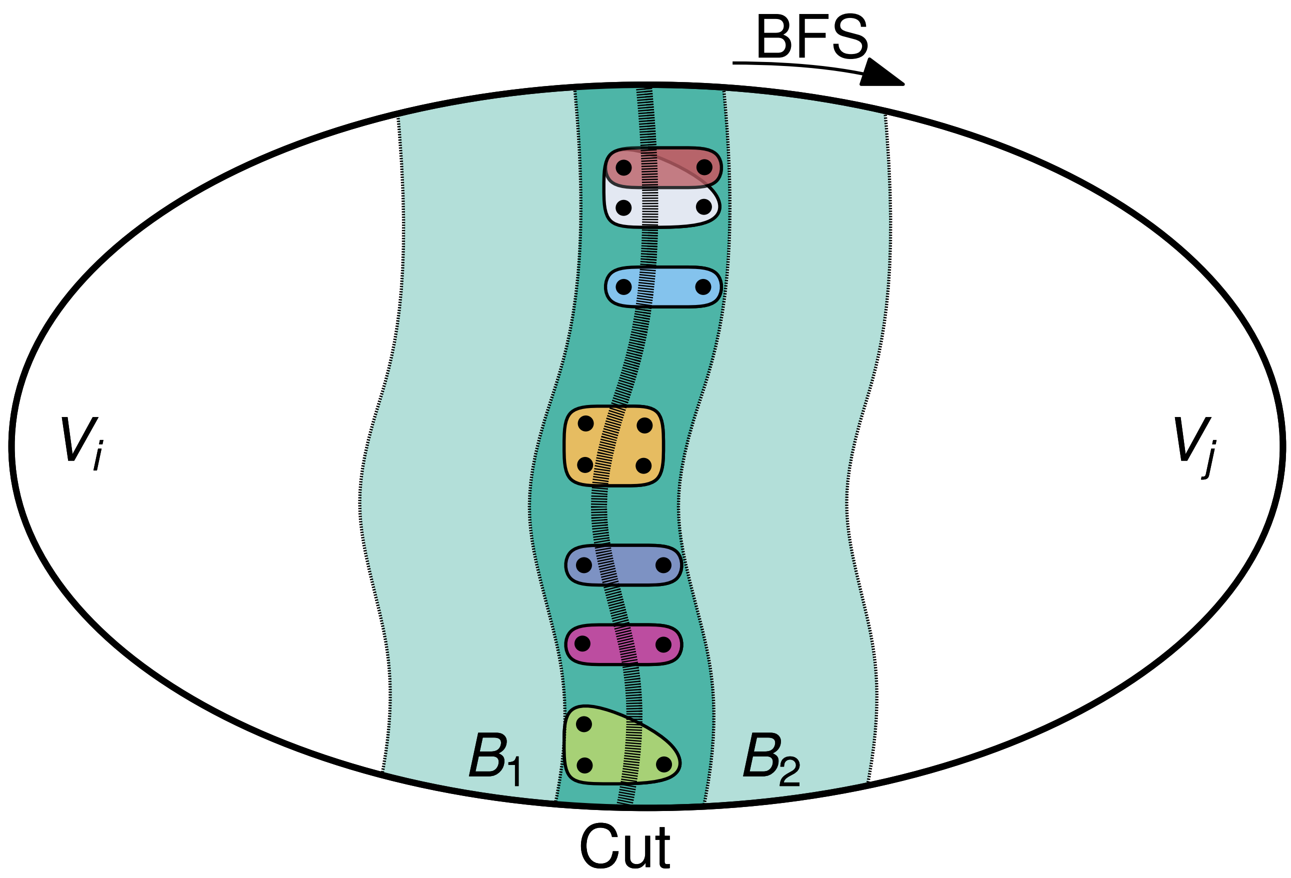
construct area $B = B_1 \cup B_2$ s.t. **every** (s,t)-cut is ε-**balanced** in H



$$c(B_1) \leq (1+\epsilon)\left\lceil \frac{c(V)}{k} \right\rceil - c(V_j) \Leftarrow$$

KaFFPa's Flow-Based Refinement for hypergraphs

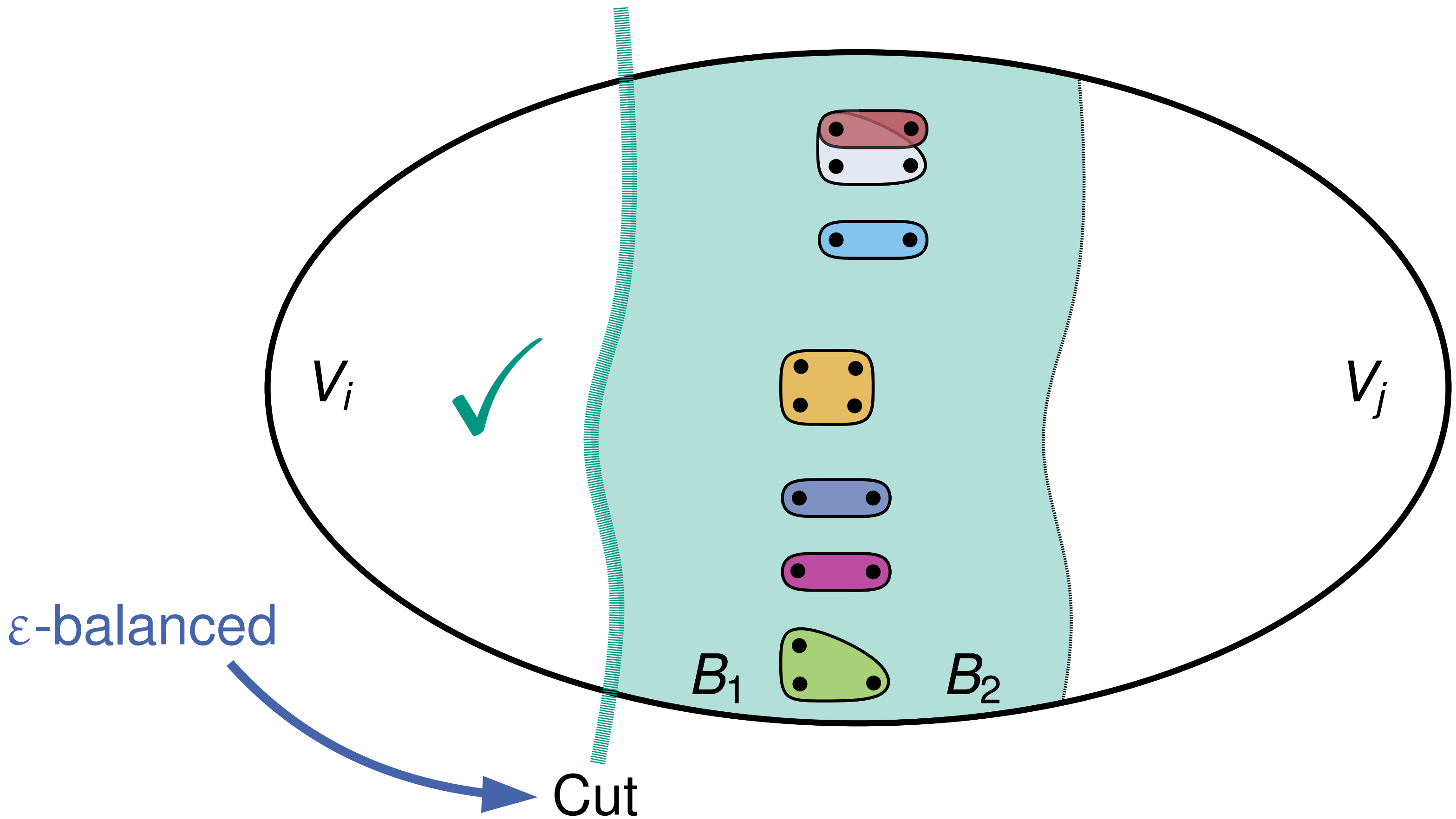
construct area $B = B_1 \cup B_2$ s.t. **every** (s,t)-cut is ε-**balanced** in H



$$c(B_1) \leq (1+\epsilon)\left\lceil \frac{c(V)}{k} \right\rceil - c(V_j) \Leftarrow \qquad \Rightarrow c(B_2) \leq (1+\epsilon)\left\lceil \frac{c(V)}{k} \right\rceil - c(V_i)$$

KaFFPa's Flow-Based Refinement for hypergraphs

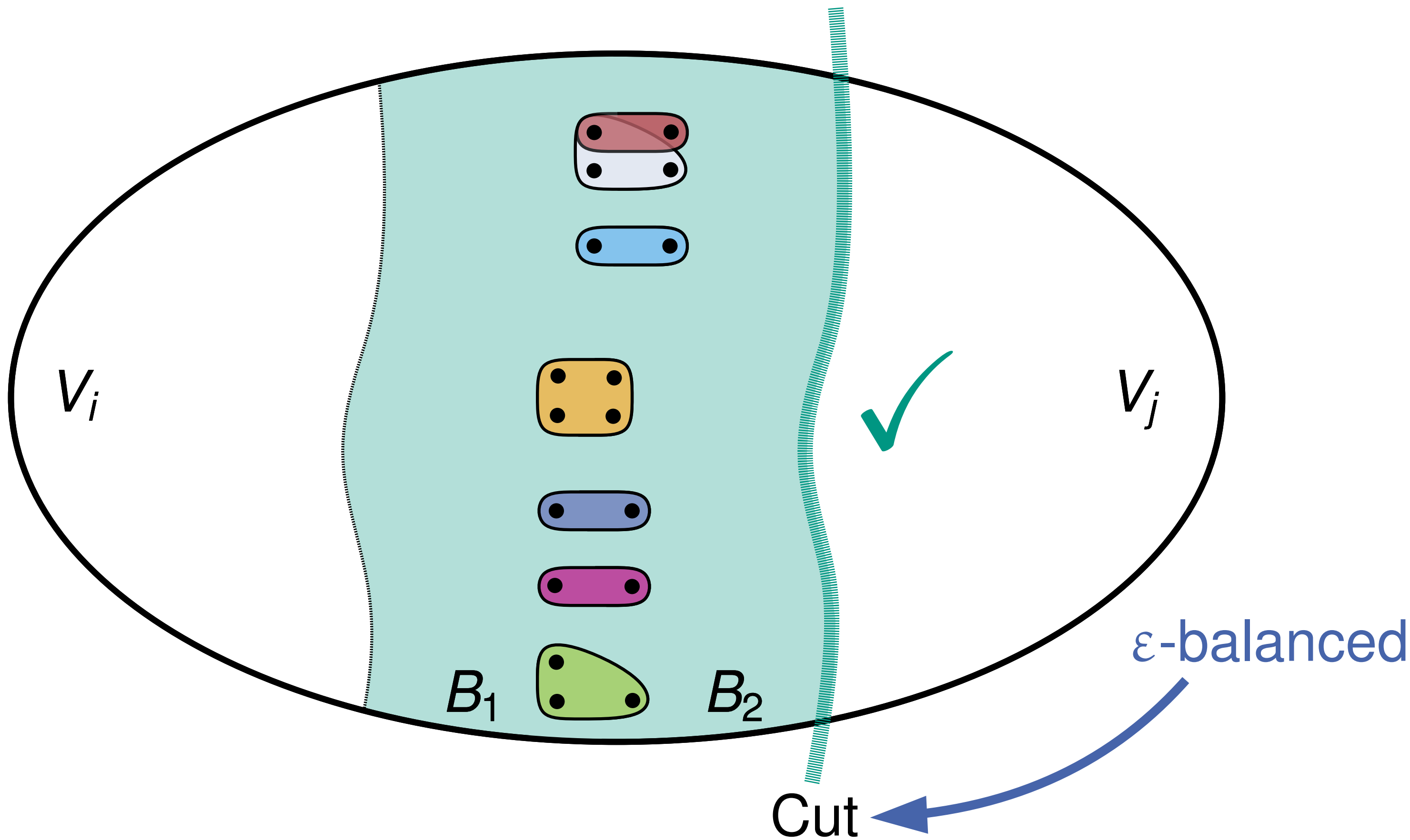
construct area $B = B_1 \cup B_2$ s.t. **every** (s,t)-cut is ε-**balanced** in H



$$c(B_1) \leq (1+\epsilon)\lceil \tfrac{c(V)}{k} \rceil - c(V_j) \Leftarrow \qquad \Rightarrow c(B_2) \leq (1+\epsilon)\lceil \tfrac{c(V)}{k} \rceil - c(V_i)$$

KaFFPa's Flow-Based Refinement for hypergraphs

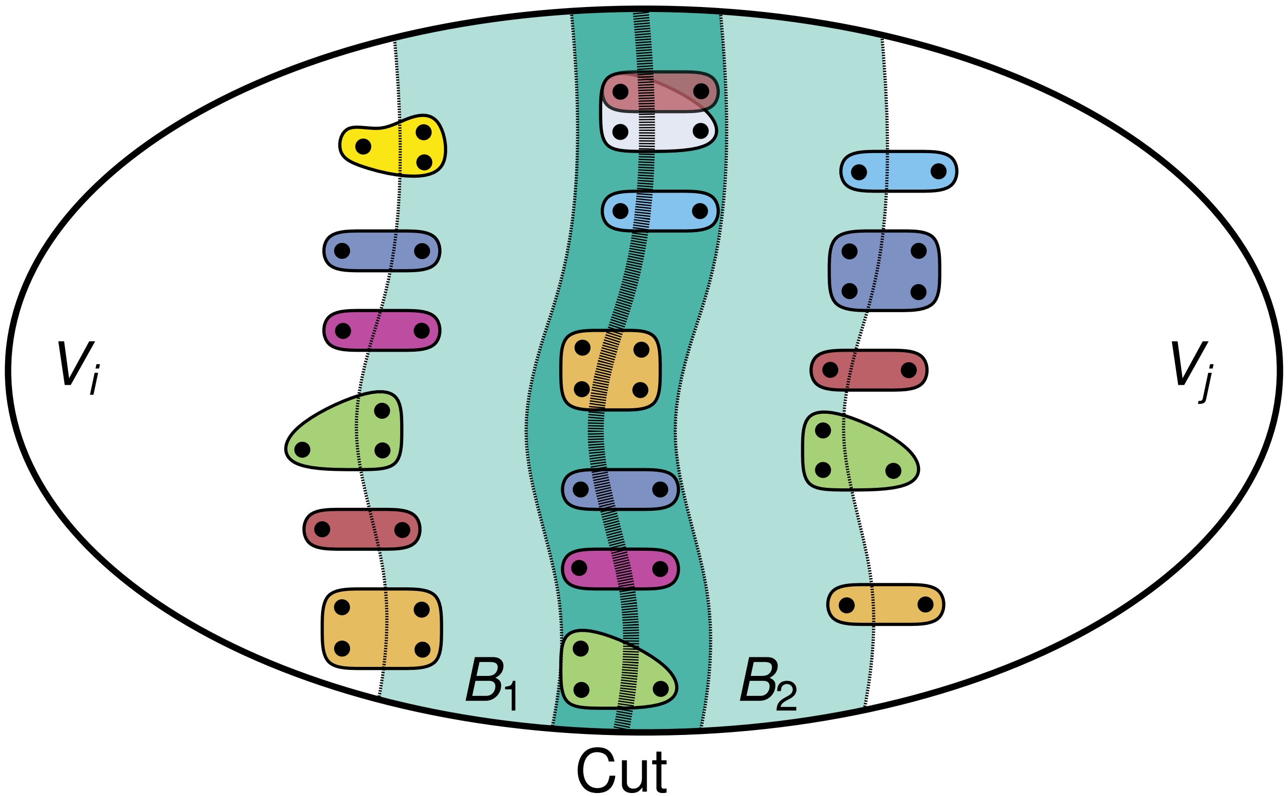
construct area $B = B_1 \cup B_2$ s.t. **every** (s,t)-cut is ε-**balanced** in H



$$c(B_1) \leq (1+\epsilon)\lceil \frac{c(V)}{k} \rceil - c(V_j) \Leftarrow \qquad \Rightarrow c(B_2) \leq (1+\epsilon)\lceil \frac{c(V)}{k} \rceil - c(V_i)$$

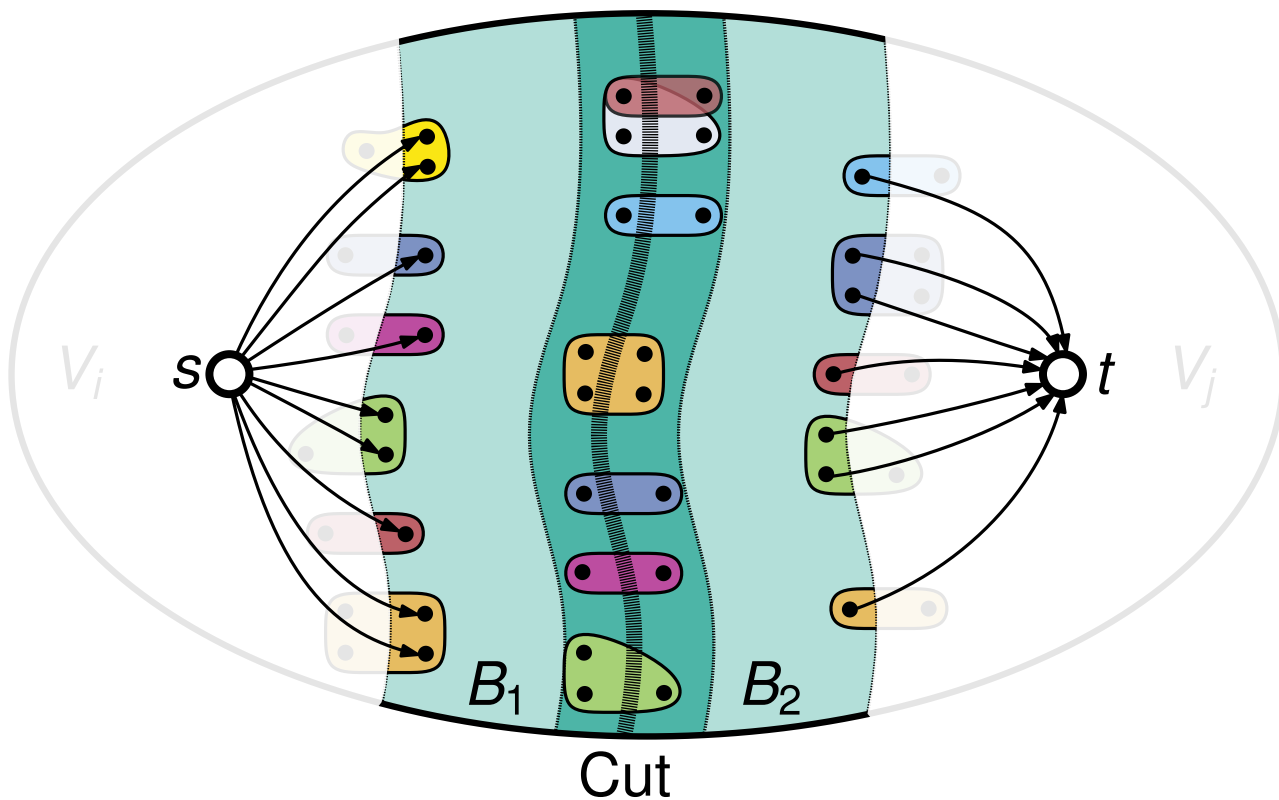
KaFFPa's Flow-Based Refinement for hypergraphs

build and solve flow problem



KaFFPa's Flow-Based Refinement for hypergraphs

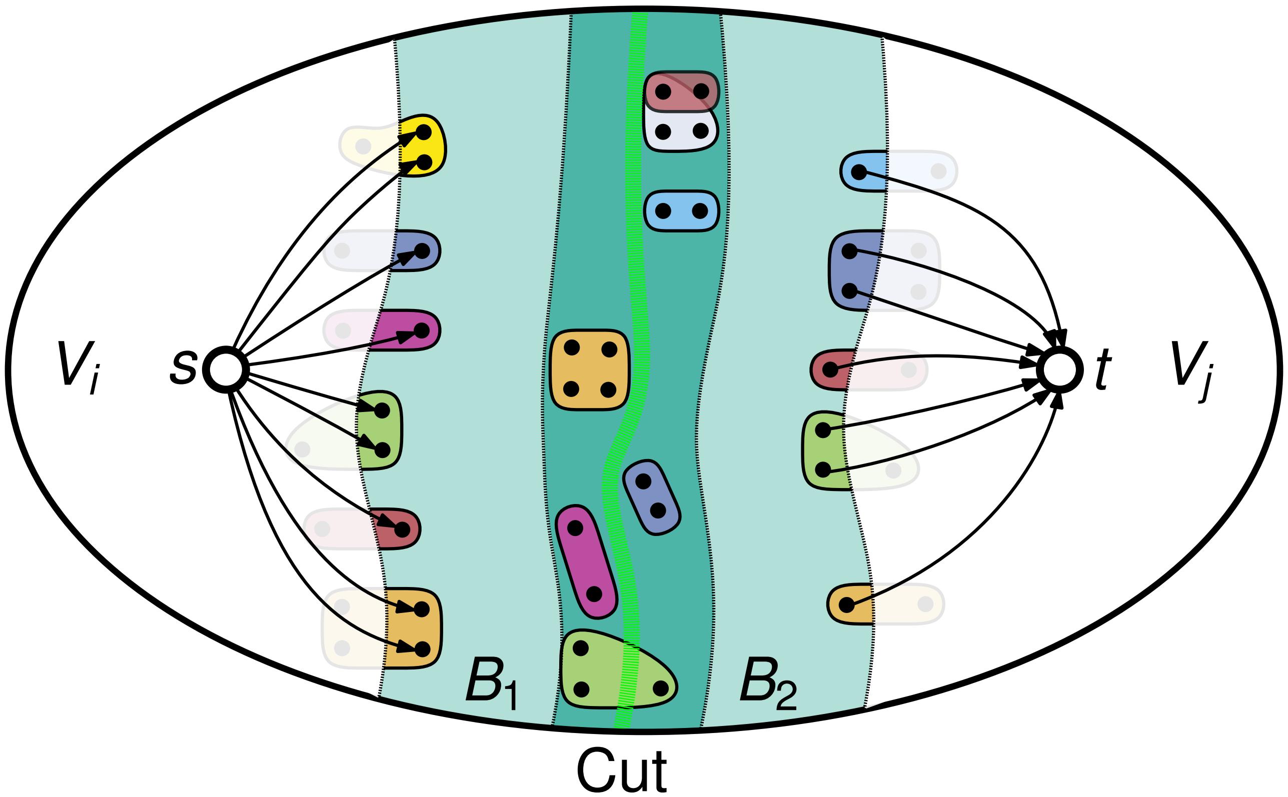
build and solve flow problem



KaFFPa's Flow-Based Refinement for hypergraphs

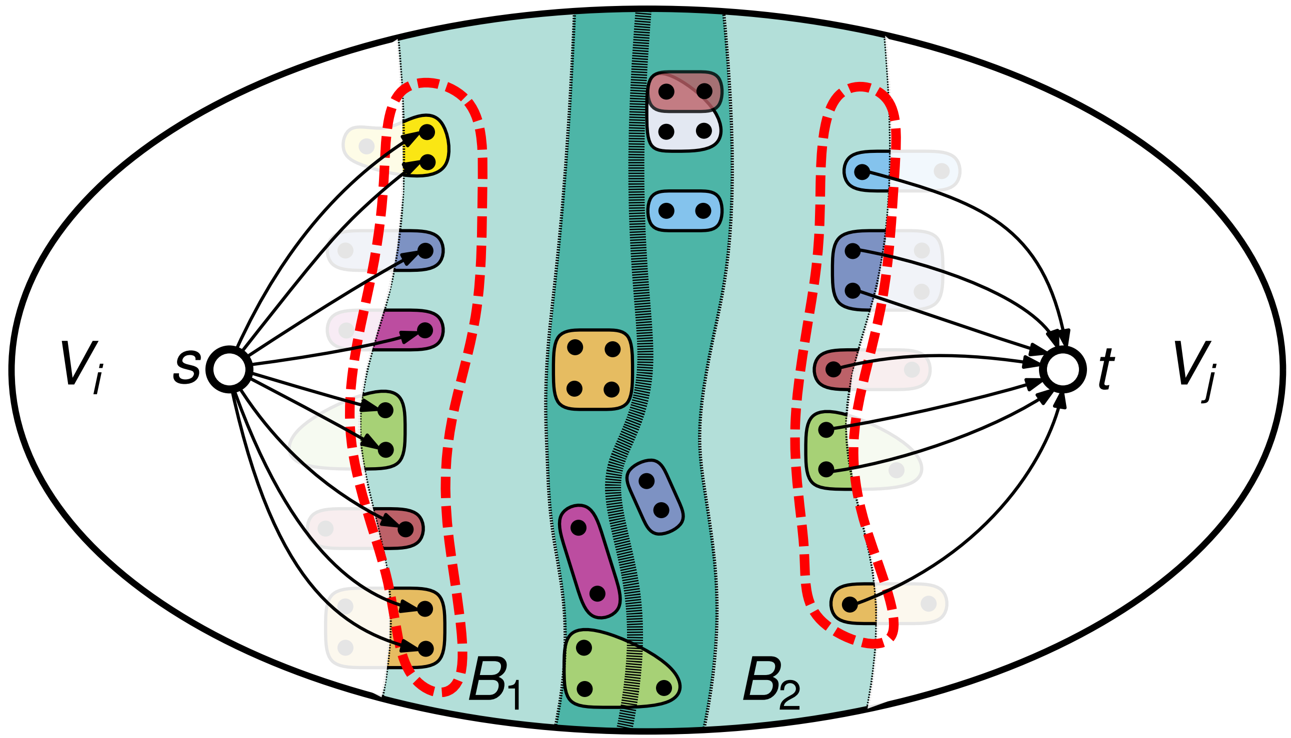
Karlsruhe Institute of Technology

build and solve flow problem



$\Rightarrow$ **optimal cut** in subhypergraph $\rightsquigarrow$ **improved** ε-balanced cut in H

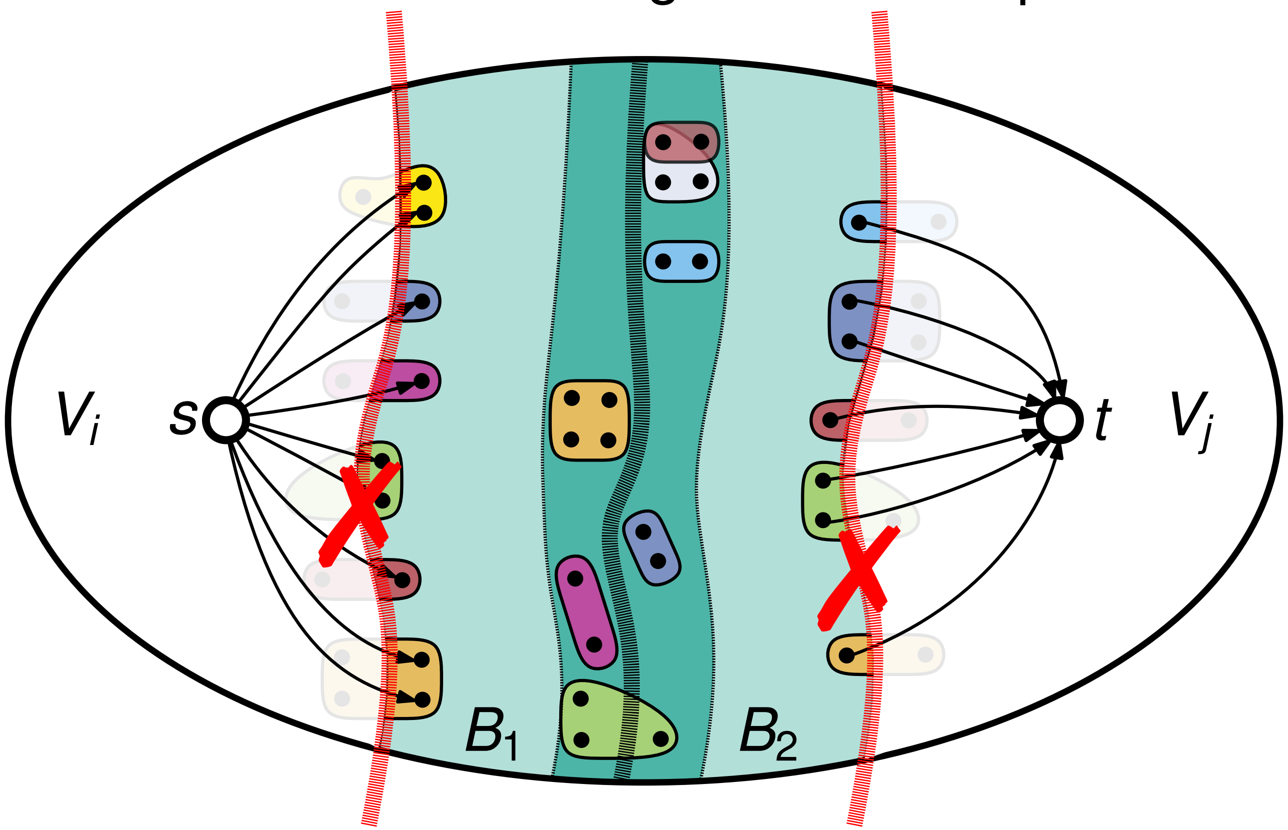
Shortcomings of the KaFFPa Approach



Shortcomings of the KaFFPa Approach

↯ border nodes **cannot move**! ↯

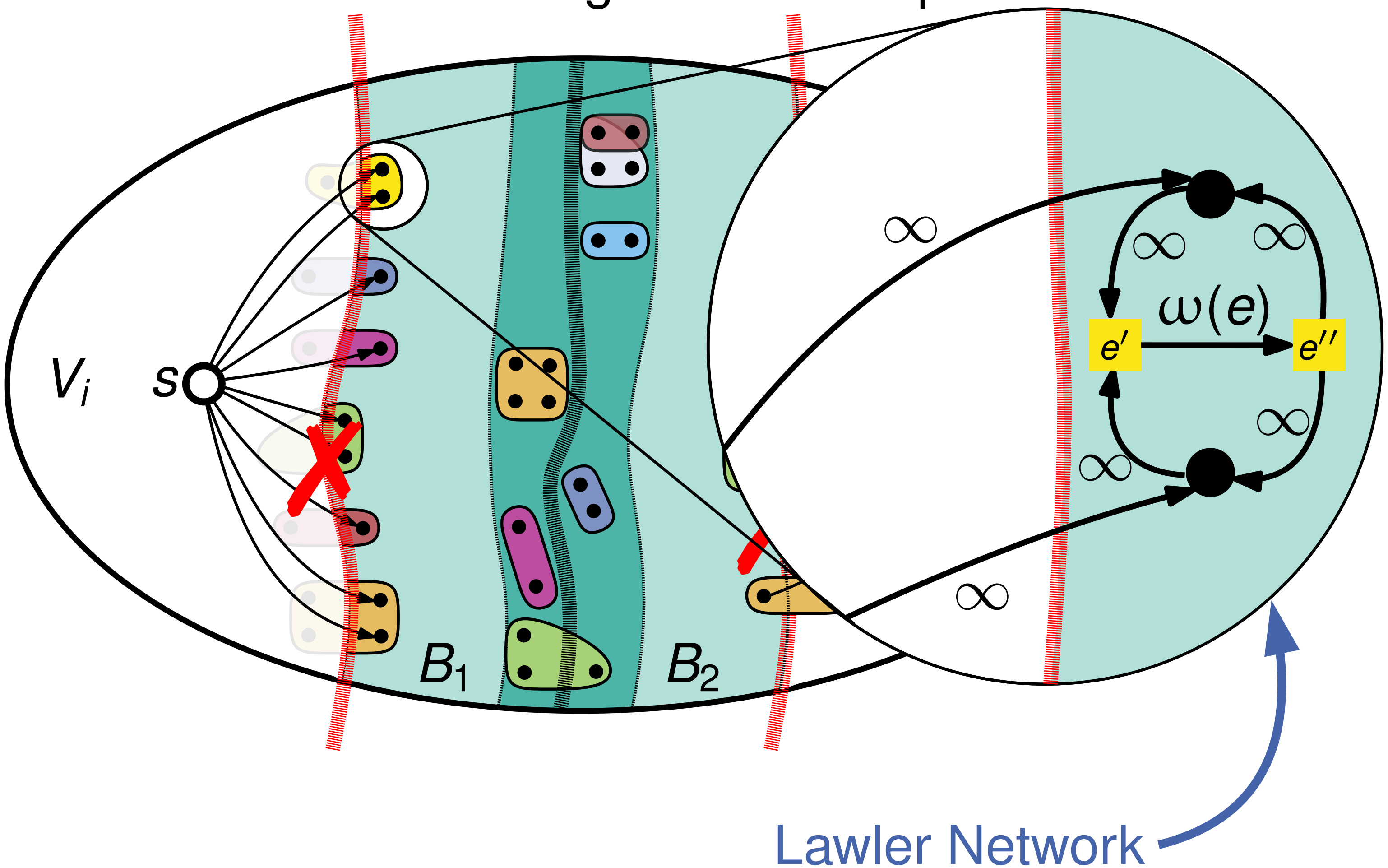
⇒ **no** min-cuts involving border nets possible!



Shortcomings of the KaFFPa Approach

↯ border nodes **cannot move**! ↯

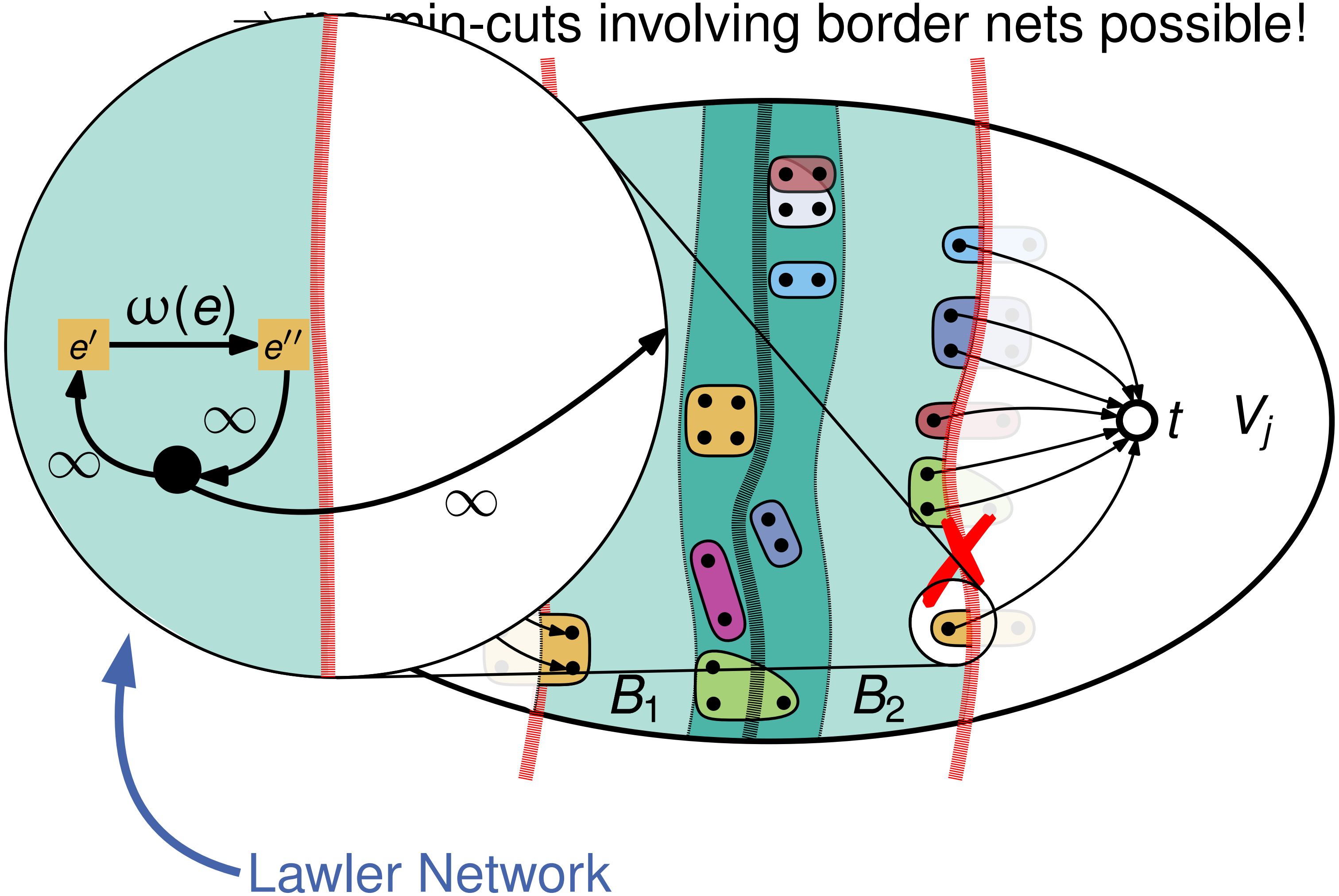
⇒ **no** min-cuts involving border nets possible!



Shortcomings of the KaFFPa Approach

border nodes **cannot move**!

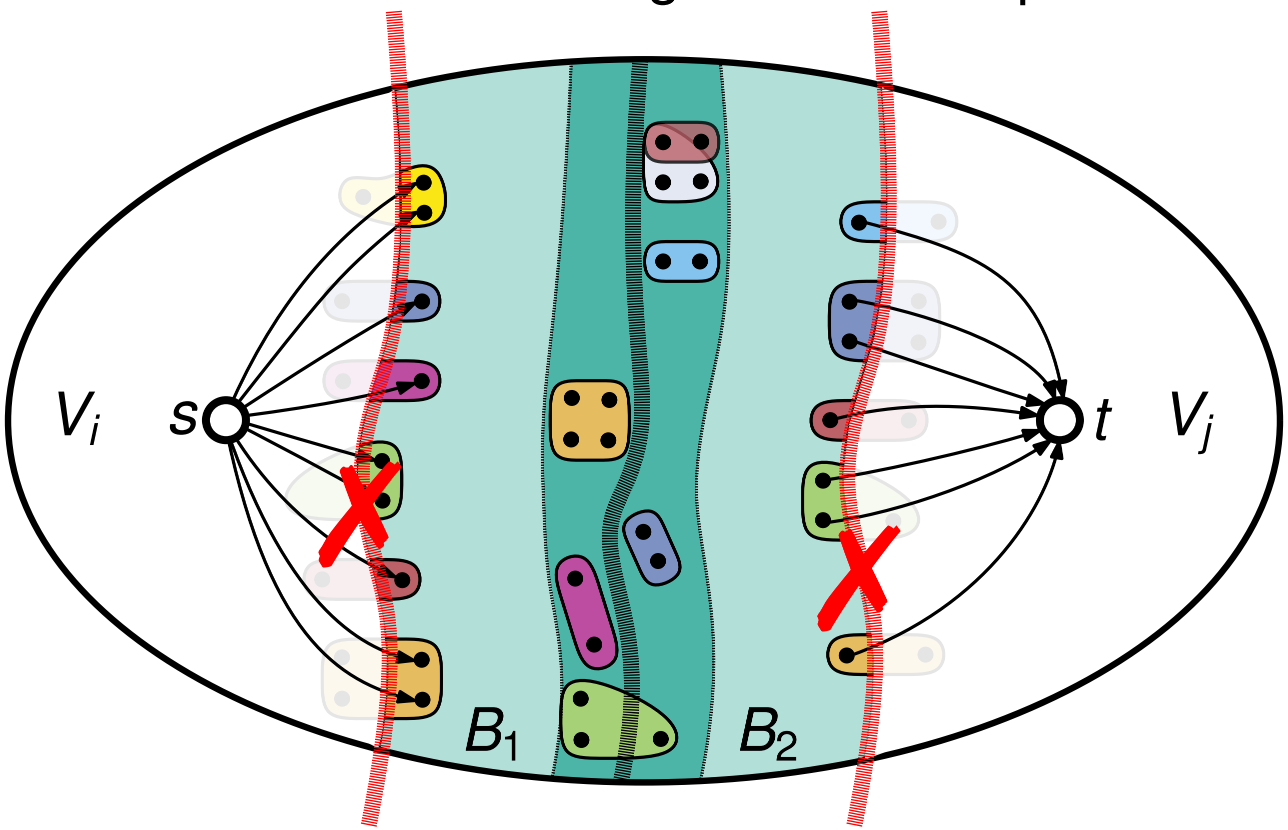
→ no min-cuts involving border nets possible!



Shortcomings of the KaFFPa Approach

↯ border nodes **cannot move**! ↯

$\Rightarrow$ **no** min-cuts involving border nets possible!



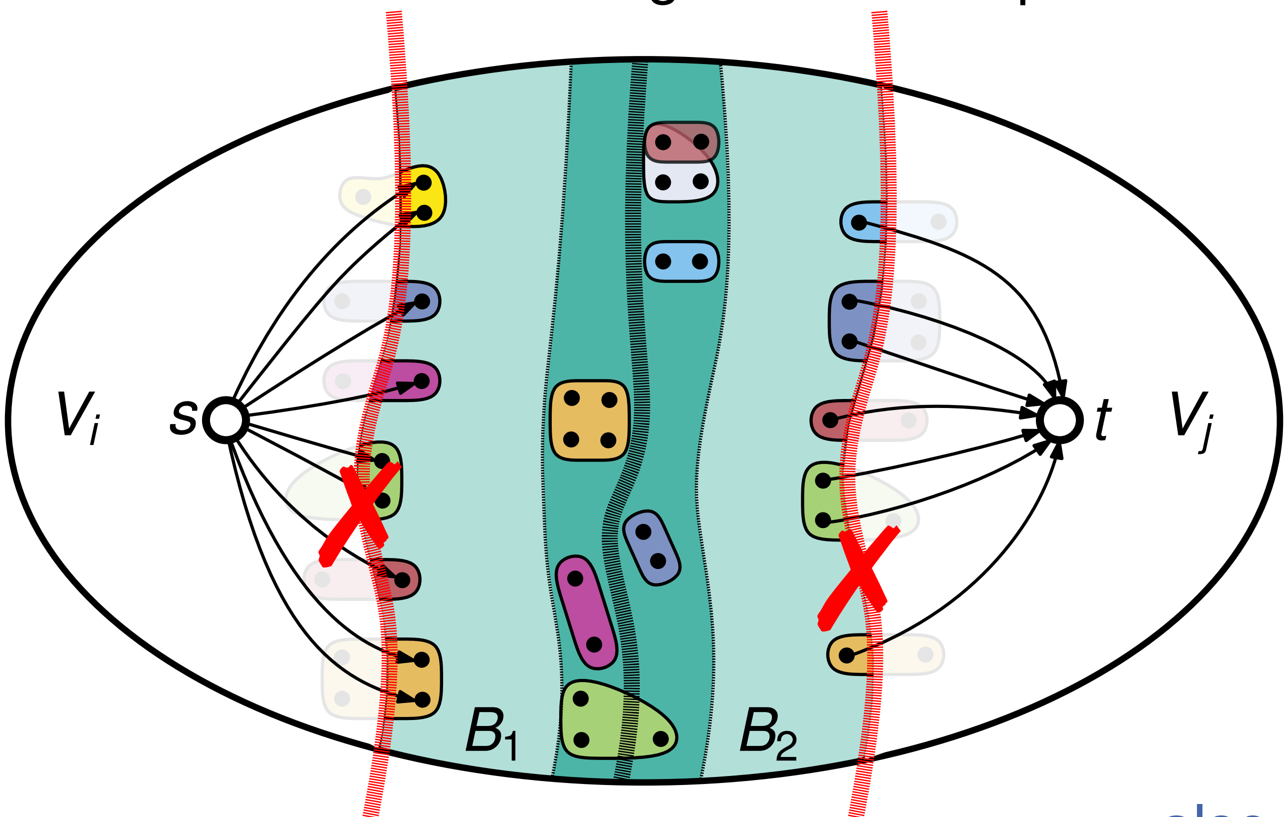
particularly bad for

- **large** nets $\rightsquigarrow$ likely to be border nets
- **small** imbalance ε $\rightsquigarrow$ small area B

Shortcomings of the KaFFPa Approach

↯ border nodes **cannot move**! ↯

$\Rightarrow$ **no** min-cuts involving border nets possible!

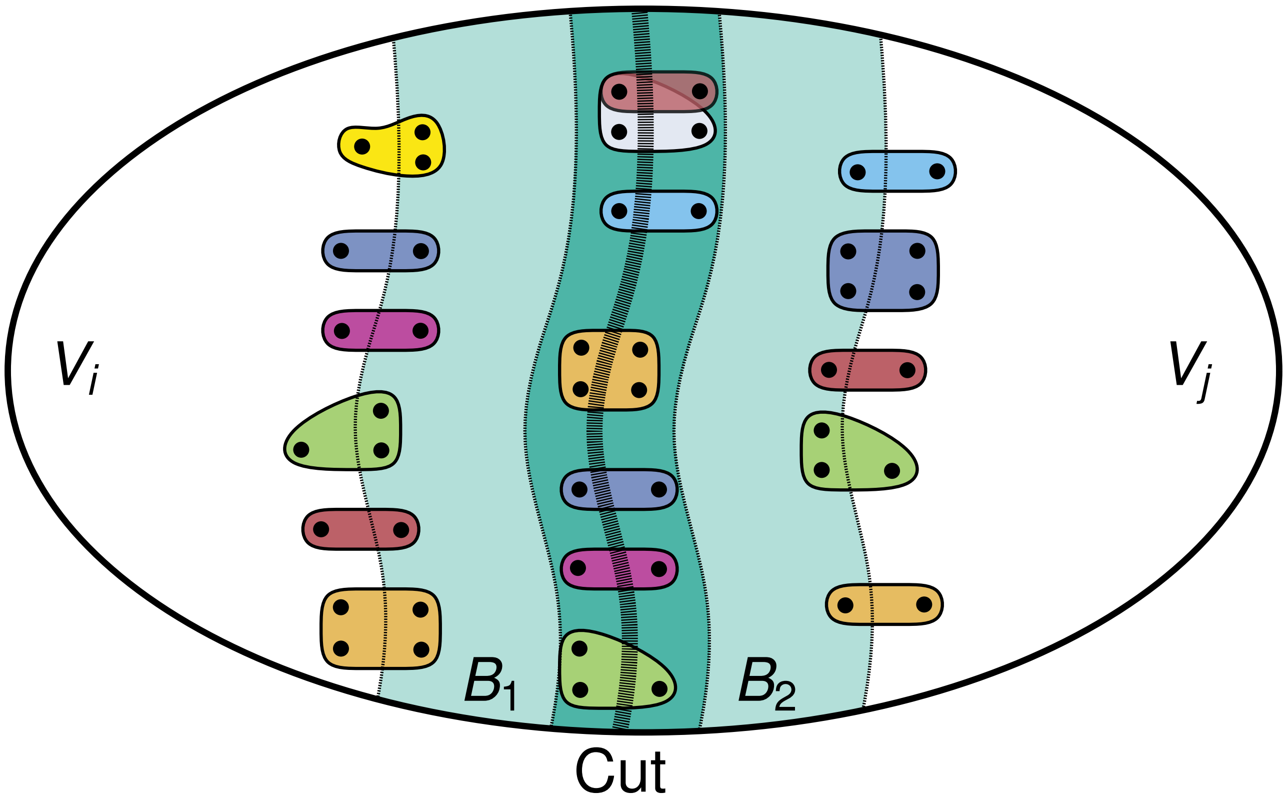


particularly bad for

- **large** nets $\rightsquigarrow$ likely to be border nets
- **small** imbalance ε $\rightsquigarrow$ small area B

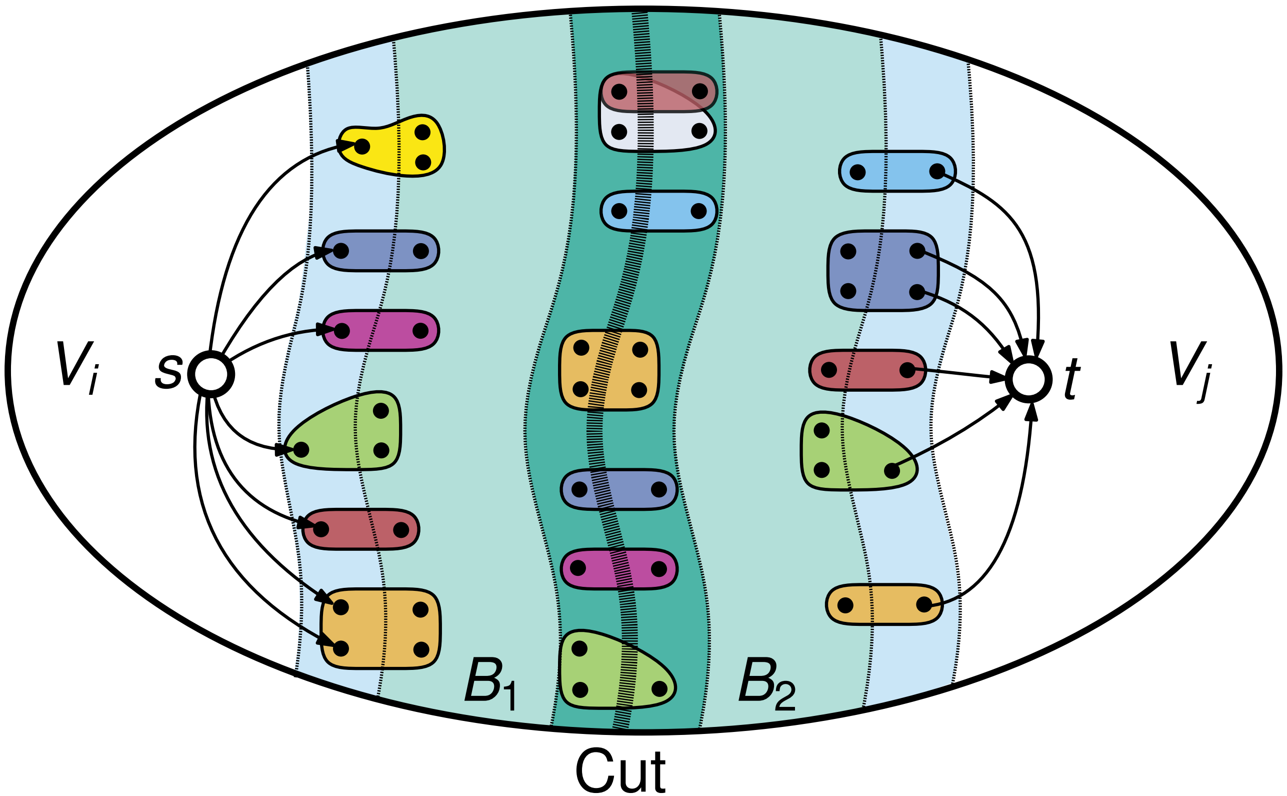
also applies to **GP**

Solution: A more flexible Model



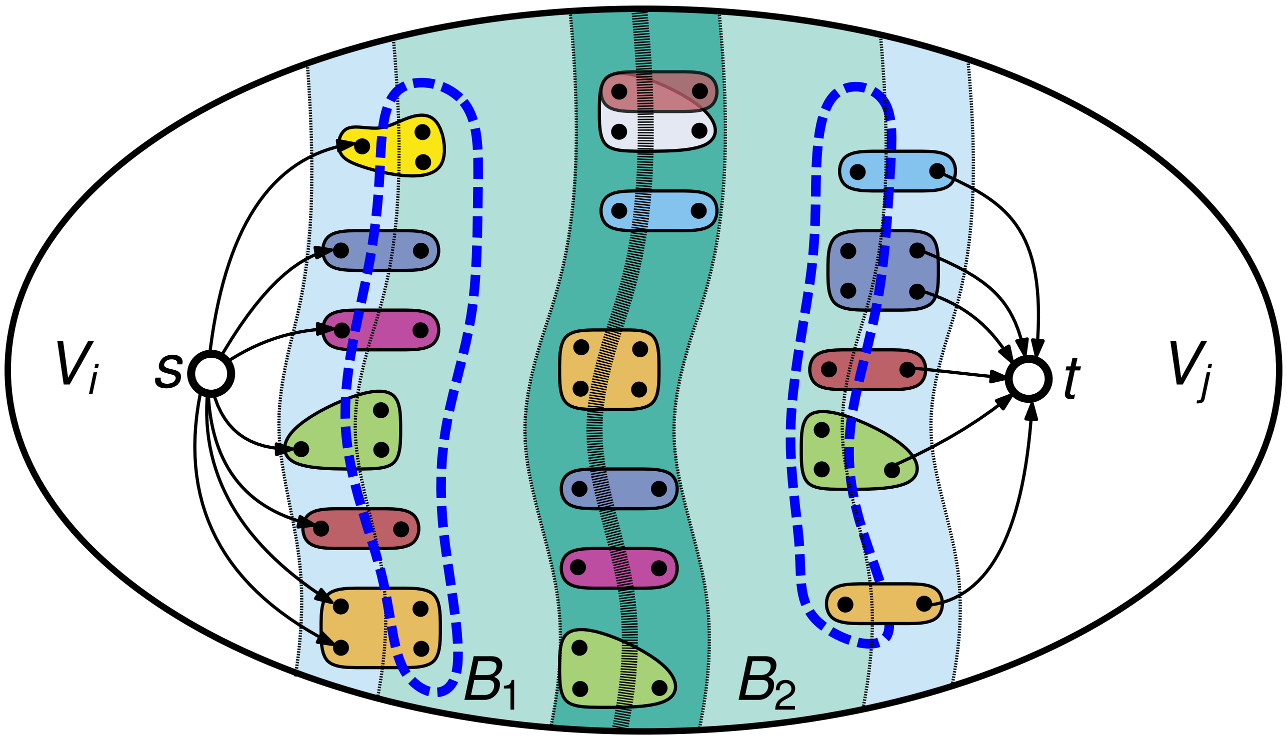
Solution: A more flexible Model

extend flow problem to include border nets



Solution: A more flexible Model

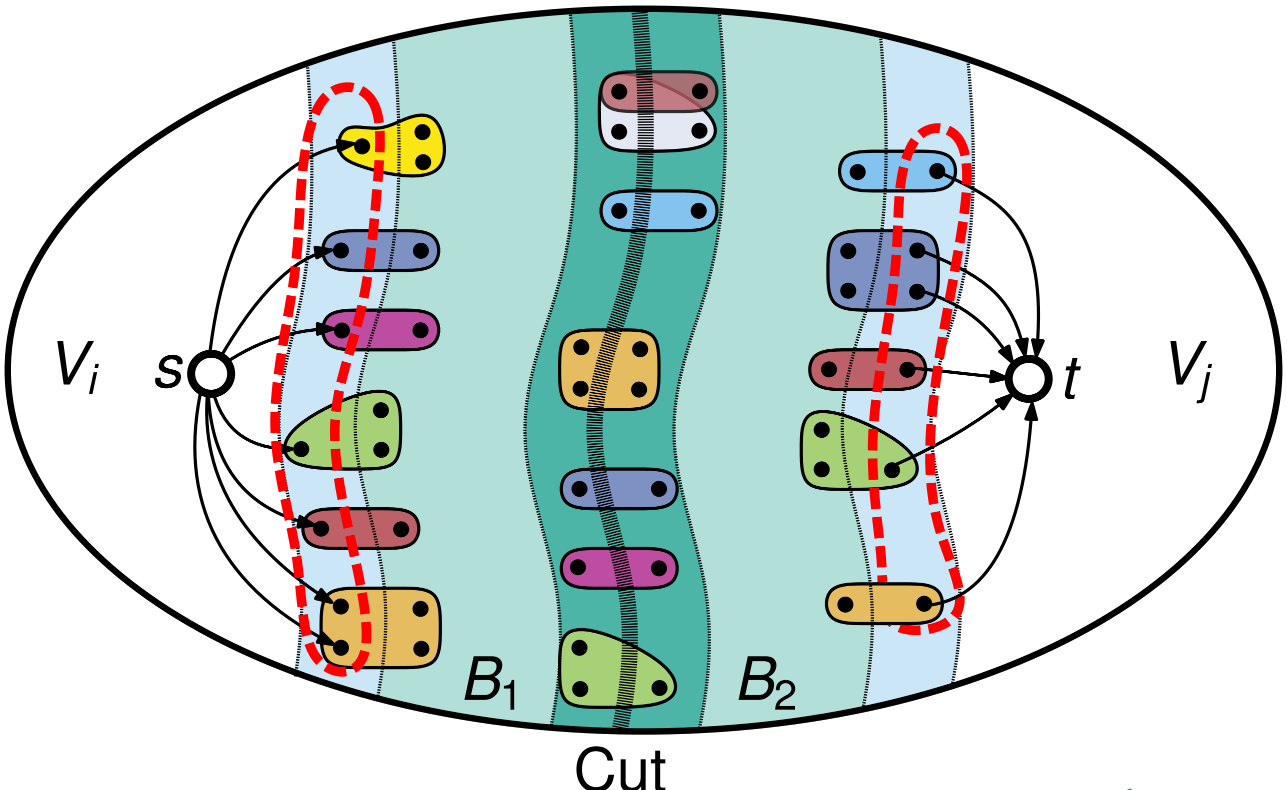
extend flow problem to include border nets



$\Rightarrow$ border nodes become **movable** ✓

Solution: A more flexible Model

extend flow problem to include border nets

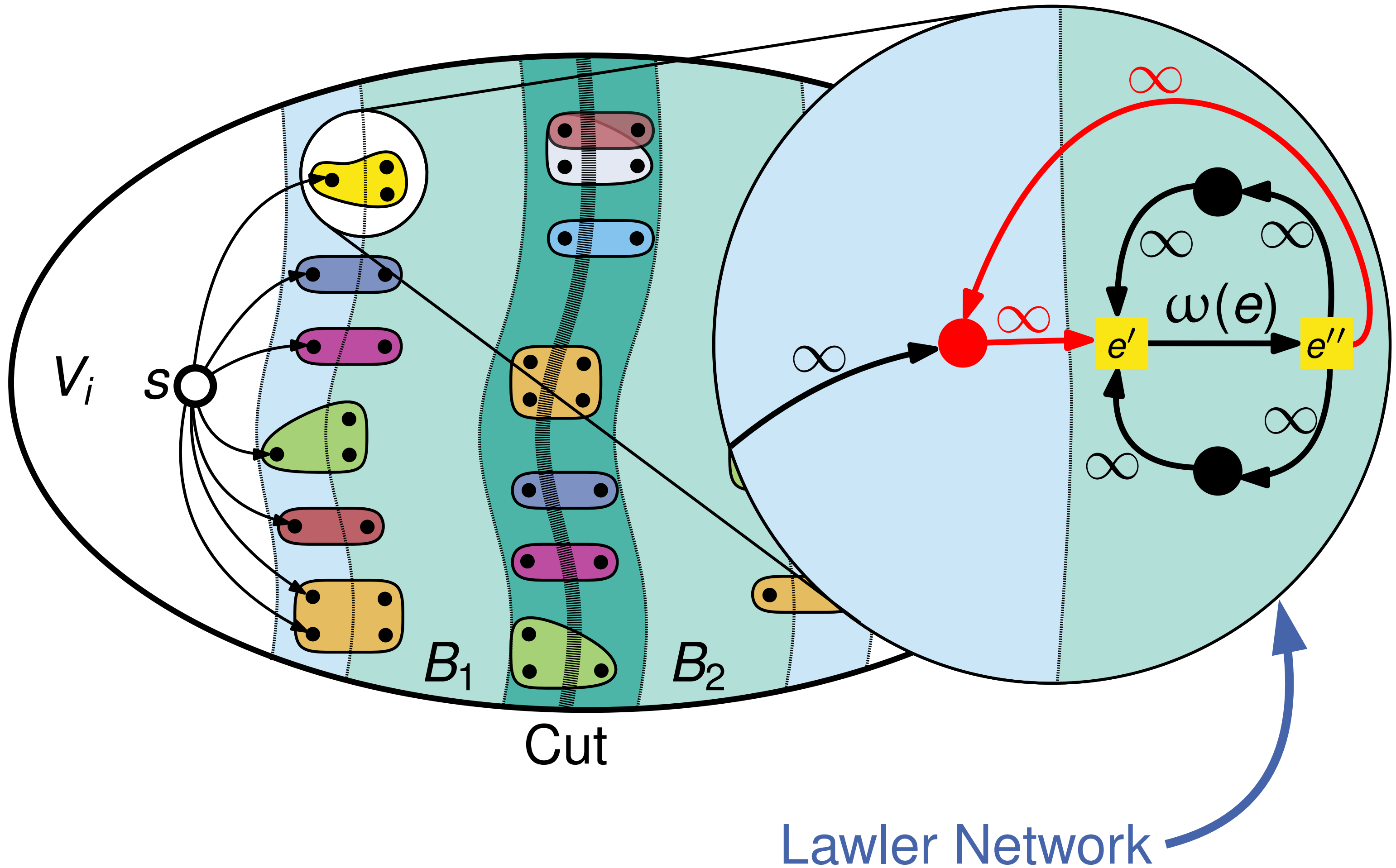


$\Rightarrow$ border nodes become **movable** ✓

$\Rightarrow$ **but** flow problem becomes **larger** ↯

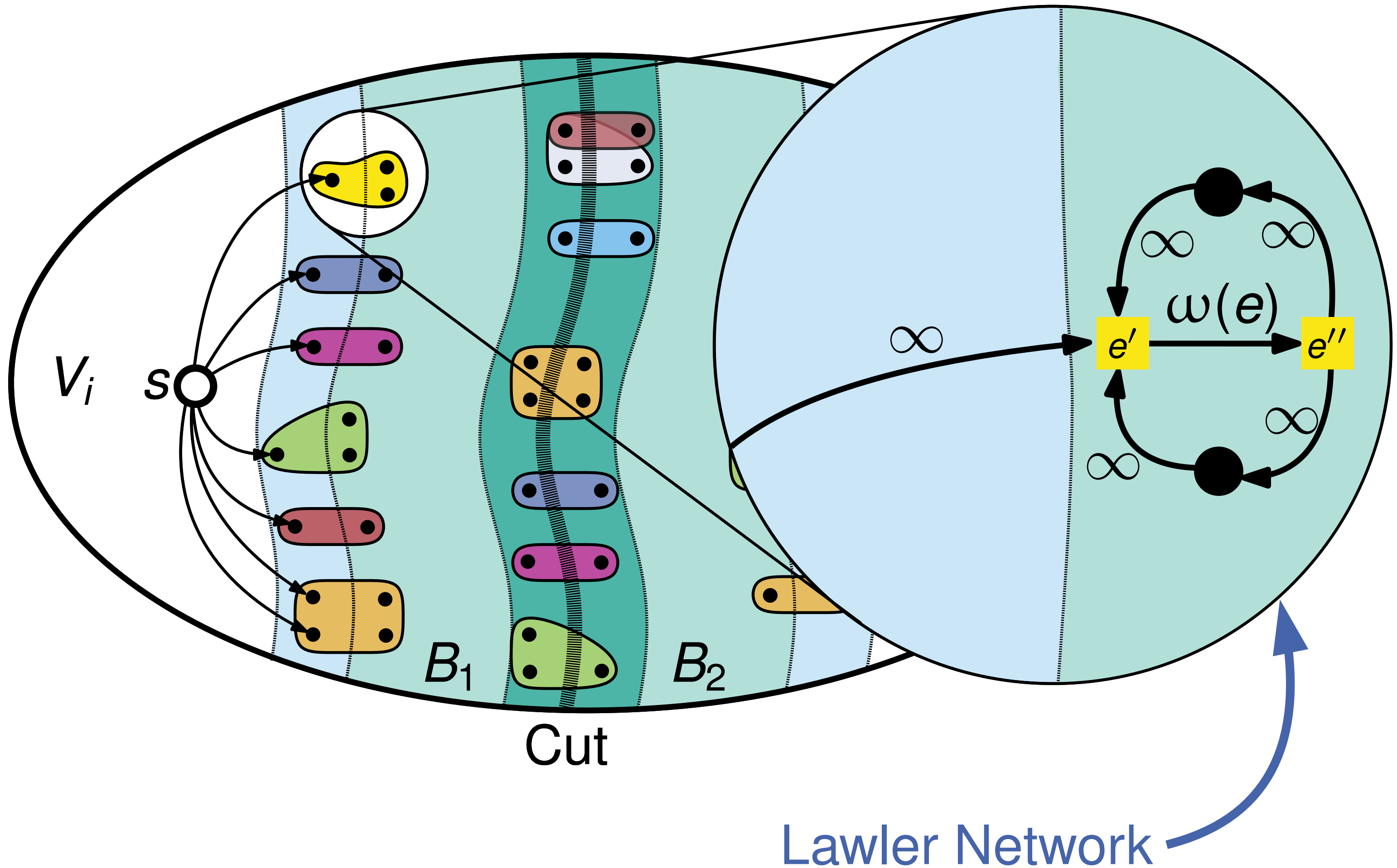
Solution: A more flexible Model

remove border pins with help of e', e'' nodes



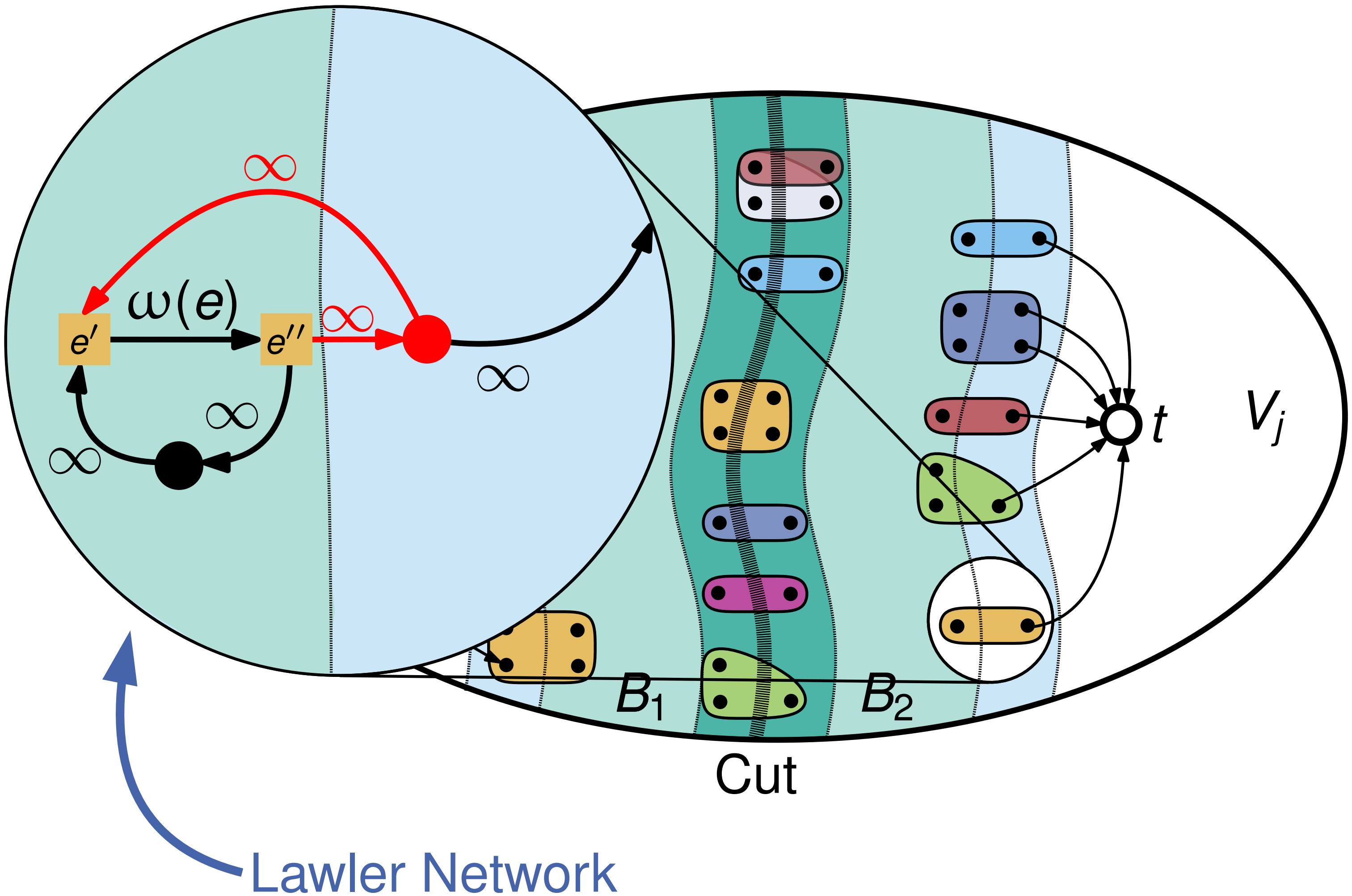
Solution: A more flexible Model

remove border pins with help of e', e'' nodes



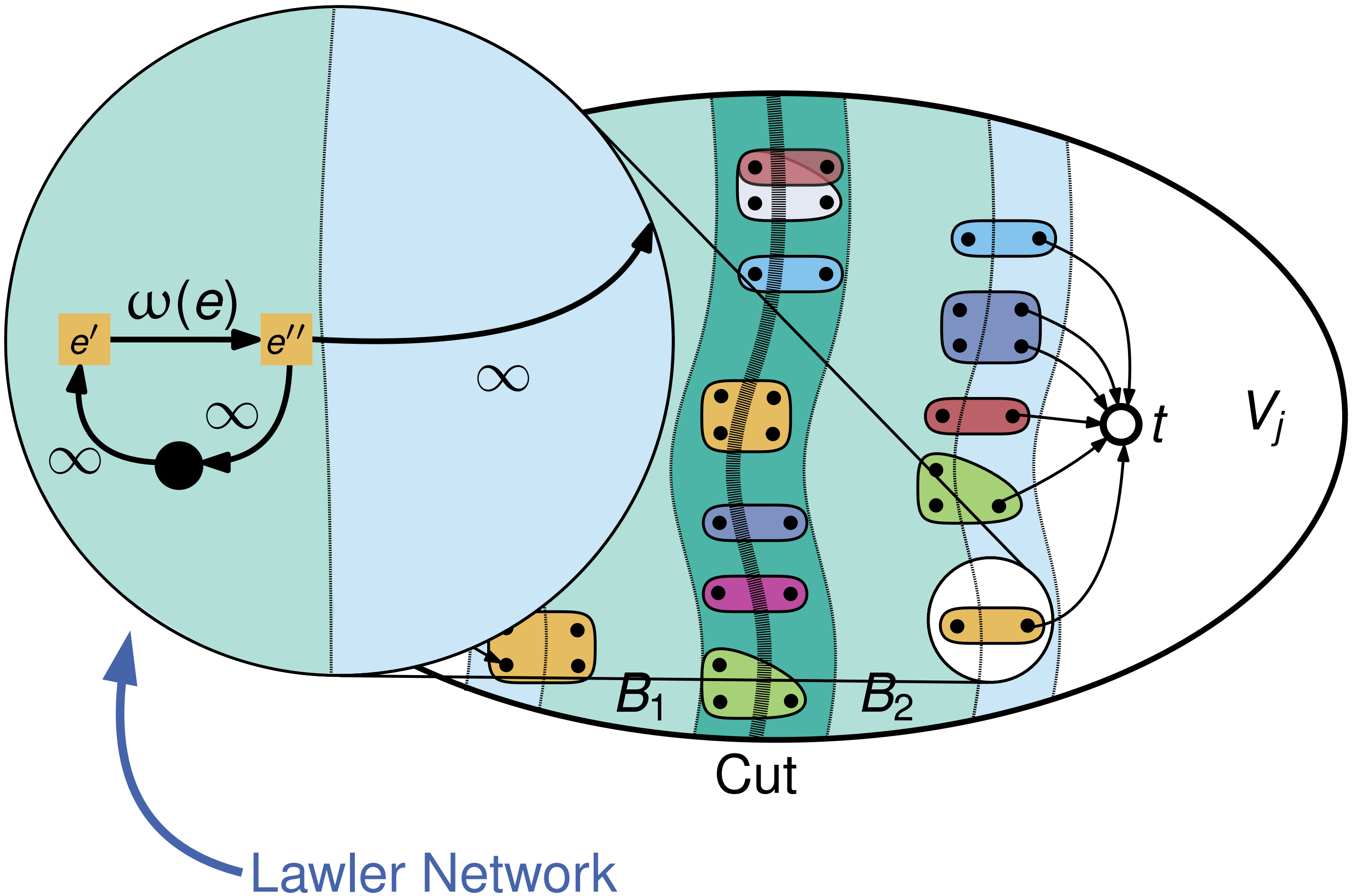
Solution: A more flexible Model

remove border pins with help of e', e'' nodes



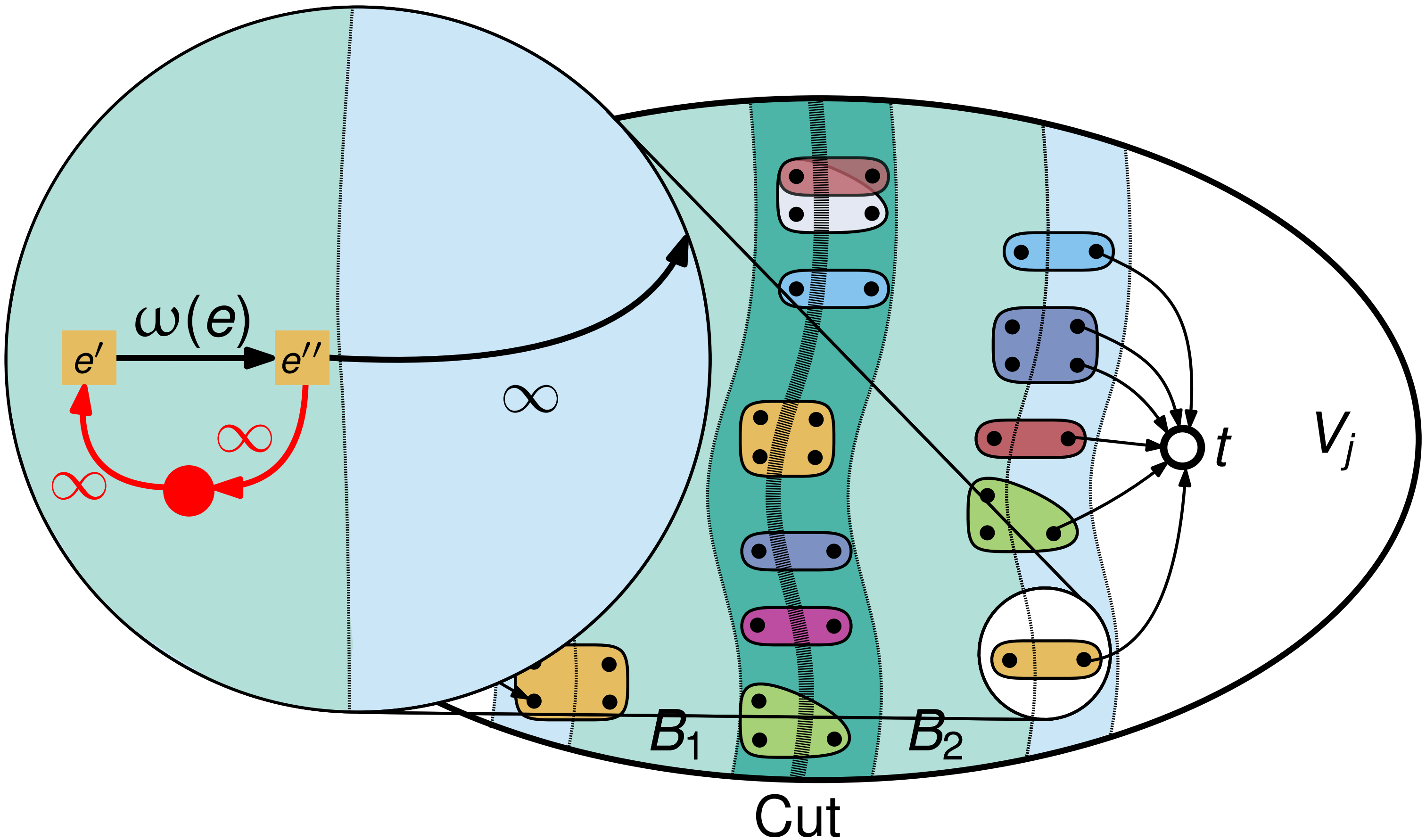
Solution: A more flexible Model

remove border pins with help of e', e'' nodes



Solution: A more flexible Model

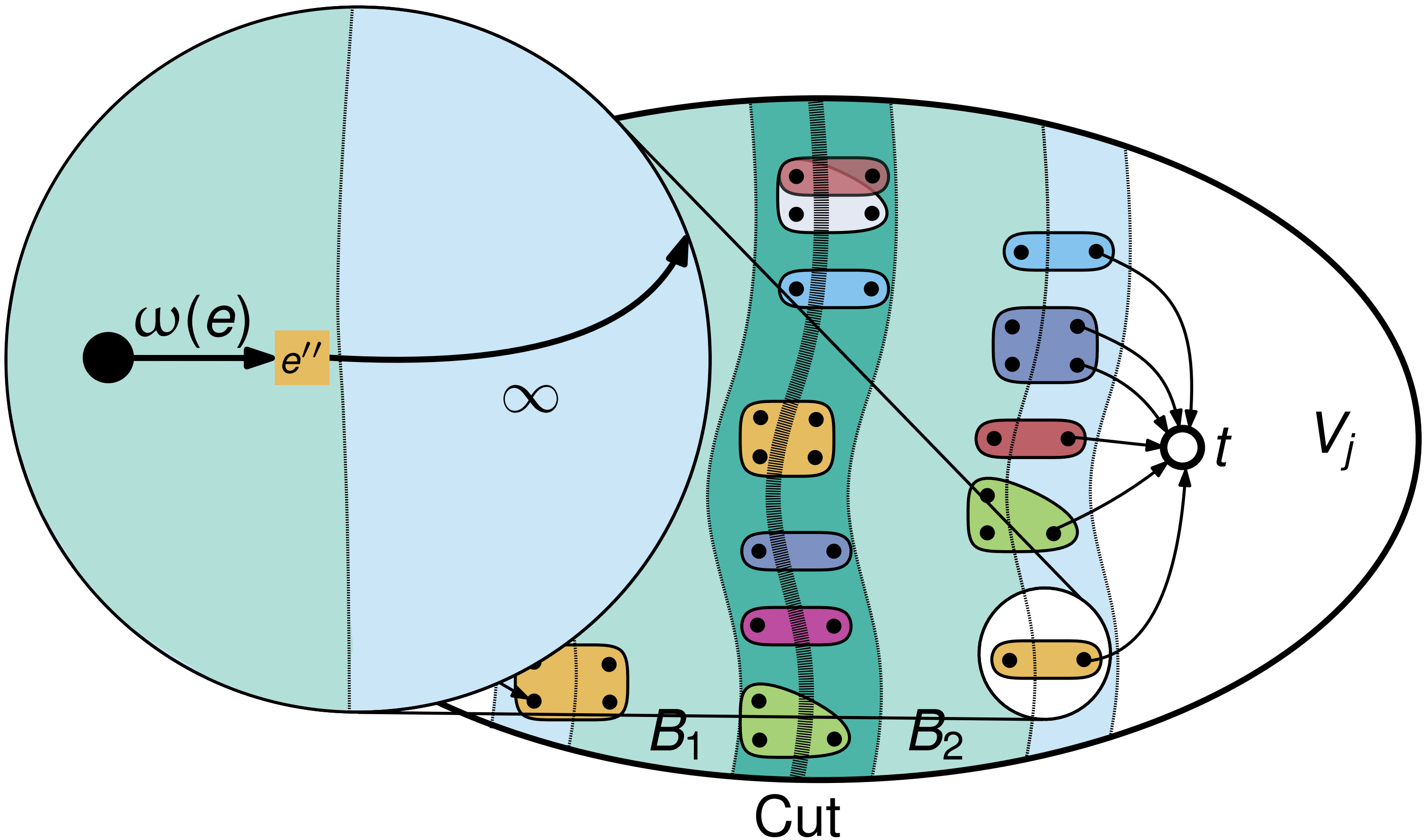
remove border pins with help of e', e'' nodes



special case: single-pin border nets

Solution: A more flexible Model

remove border pins with help of e', e'' nodes



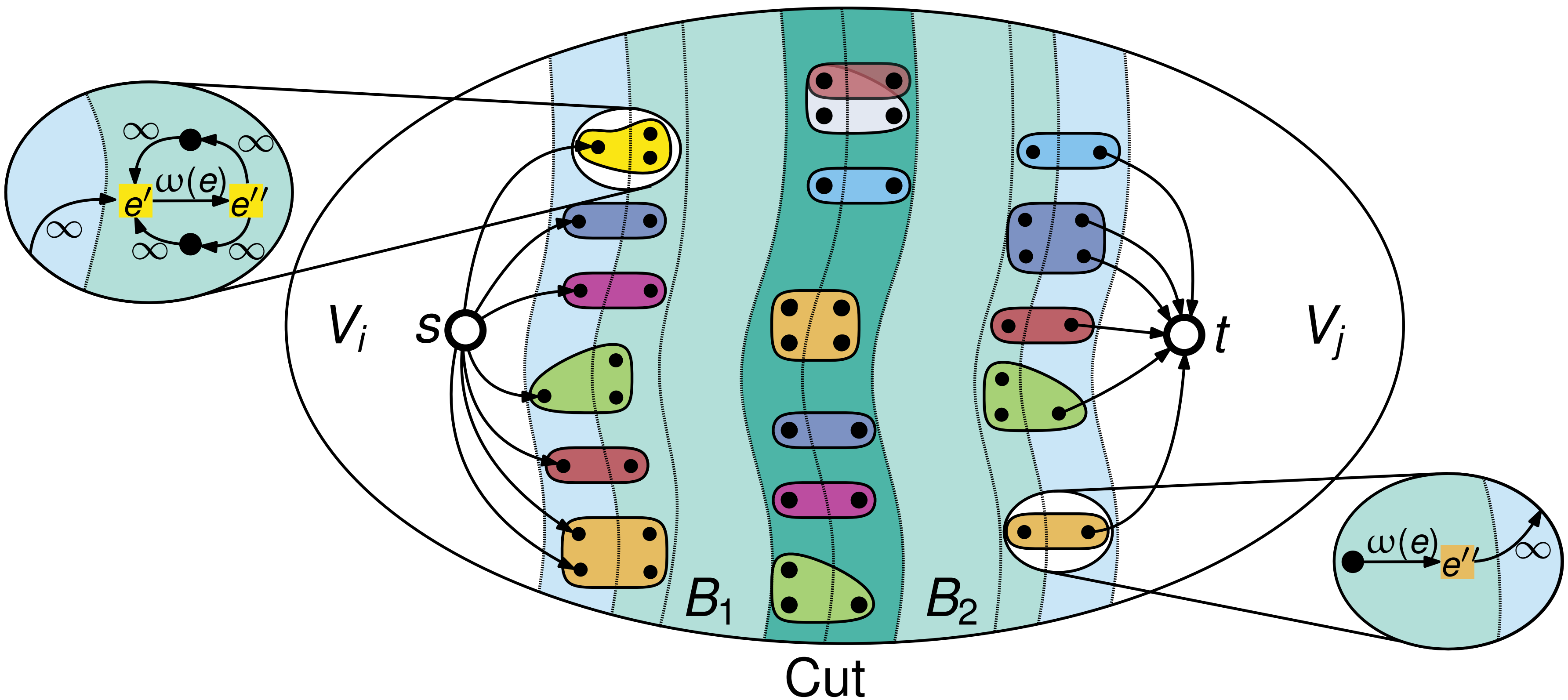
special case: single-pin border nets

A more flexible Model – Summary

✓ **movable** border nodes ⇝ **all** cuts are feasible

✓ **no** increase in problem size

✓ further size **reduction** through $|e| = 1$ border nets



Experiments – Benchmark Setup

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

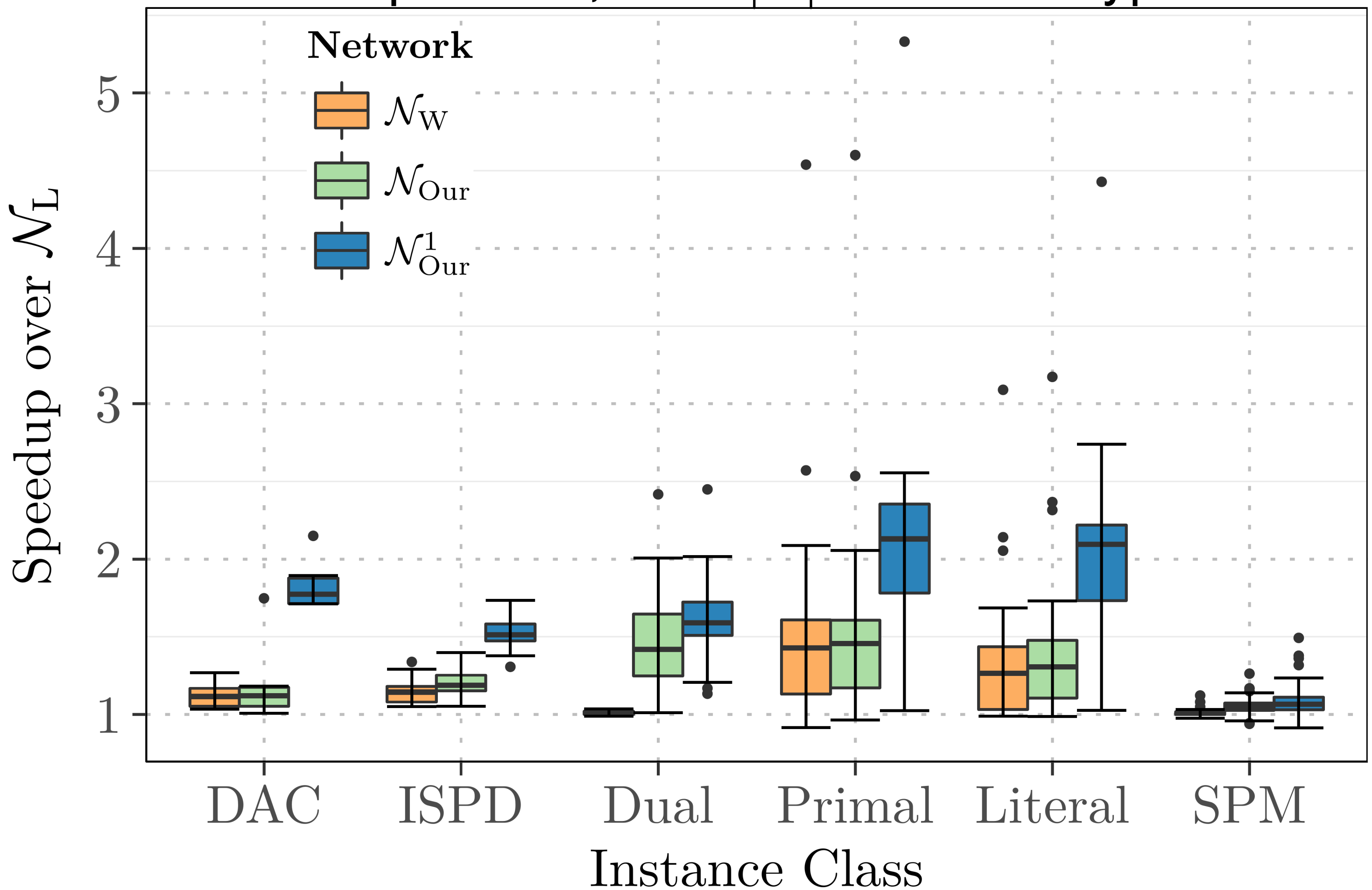
- # (Hyper)graphs: [publicly available]
  - SuiteSparse Matrix Collection 184
  - SAT Competition 2014 (3 representations) 92·3
  - ISPD98 & DAC2012 VLSI Circuits 28
  - DIMACS Graphs [flow model experiments] 15

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

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

Size Reduction Of Hypergraph Flow Networks

Max-Flow Computation, area $|B| = 25.000$ hypernodes



$\mathcal{N}_L$: Lawler Network

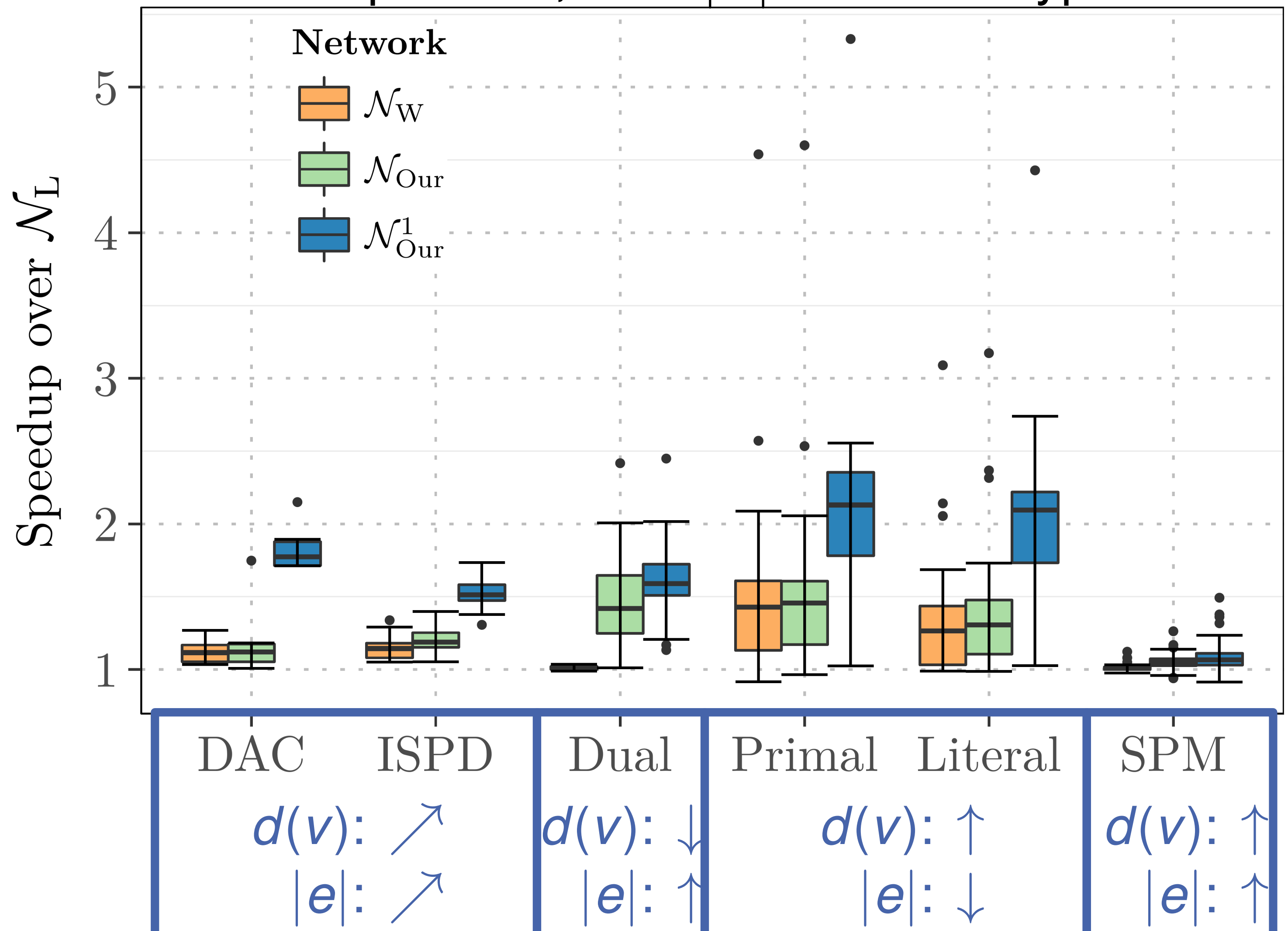
$\mathcal{N}_W$: Liu-Wong Network

$\mathcal{N}_{Our}$: Our Network

$\mathcal{N}^1_{Our}$: Our Network with $|e| = 1$ opt.

Size Reduction Of Hypergraph Flow Networks

Max-Flow Computation, area $|B| = 25.000$ hypernodes



$\mathcal{N}_L$: Lawler Network

$\mathcal{N}_W$: Liu-Wong Network

$\mathcal{N}_{Our}$: Our Network

$\mathcal{N}^1_{Our}$: Our Network with $|e| = 1$ opt.

Impact of KaHyPar's Flow Model

Average Improvement [%] over the KaFFPa Approach

	Hypergraphs			DIMACS Graphs		
α'	ε = 1%	ε = 3%	ε = 5%	ε = 1%	ε = 3%	ε = 5%
1	7.7	8.1	7.6	11.7	11.3	10.5
2	7.9	6.6	4.8	11.0	9.1	7.8
4	6.9	3.9	2.7	9.9	7.3	5.4
8	5.1	2.3	1.5	8.6	5.3	3.9
16	3.4	1.3	1.2	7.0	4.1	3.5

$\Rightarrow$ performs **better** on **all** problem sizes and imbalances
$\Rightarrow$ most pronounced for **small** flow problems & imbalances
$\Rightarrow$ effects also visible for **graphs**

Impact of KaHyPar's Flow Model

Average Improvement [%] over the KaFFPa Approach

	Hypergraphs			DIMACS Graphs		
α'	ε = 1%	ε = 3%	ε = 5%	ε = 1%	ε = 3%	ε = 5%
1	7.7	8.1	7.6	11.7	11.3	10.5
2	7.9	6.6	4.8	11.0	9.1	7.8
4	6.9	3.9	2.7	9.9	7.3	5.4
8	5.1	2.3	1.5	8.6	5.3	3.9
16	3.4	1.3	1.2	7.0	4.1	3.5

$\Rightarrow$ performs **better** on **all** problem sizes and imbalances
$\Rightarrow$ most pronounced for **small** flow problems & imbalances
$\Rightarrow$ effects also visible for **graphs**

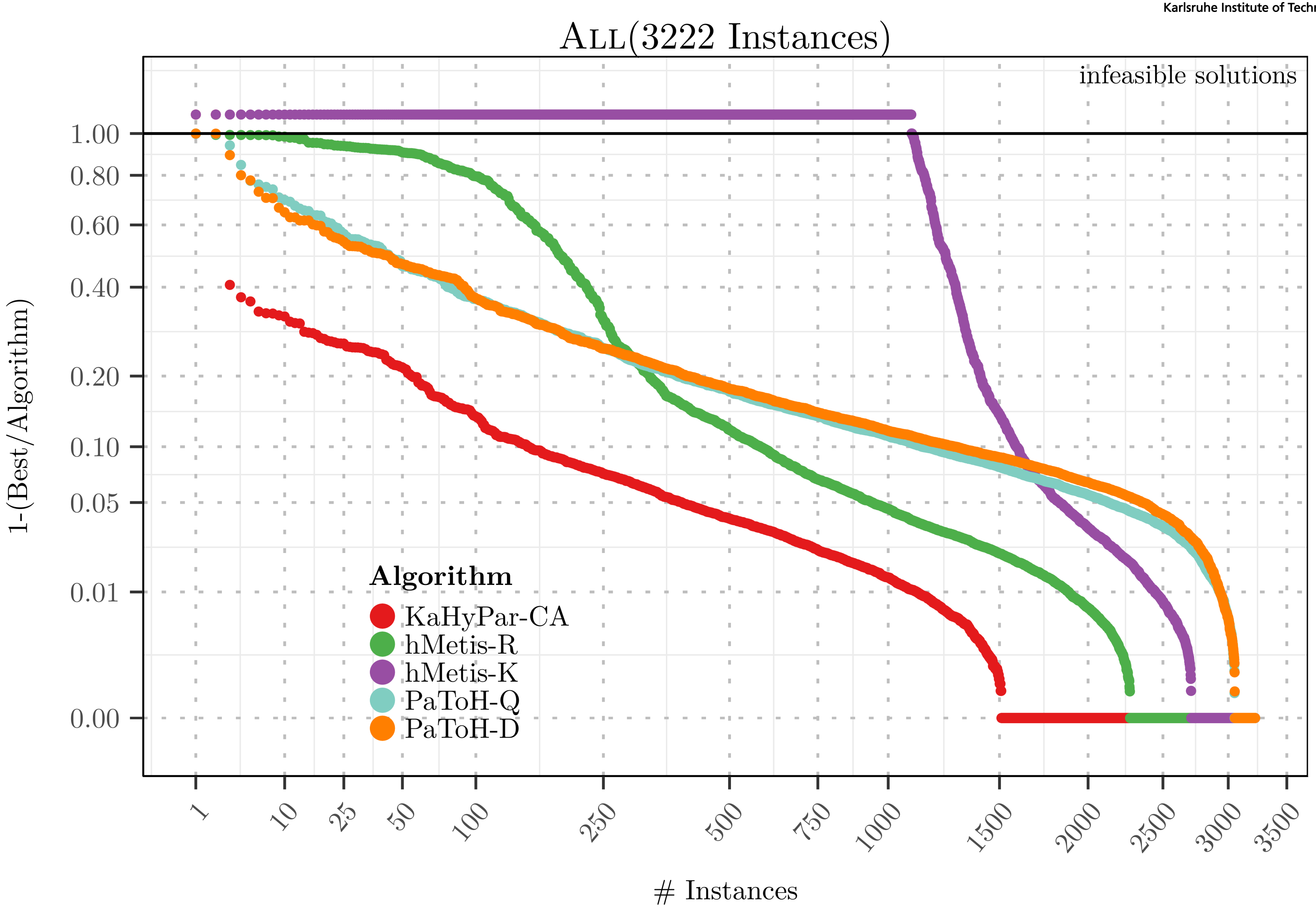
Impact of KaHyPar's Flow Model

Average Improvement [%] over the KaFFPa Approach

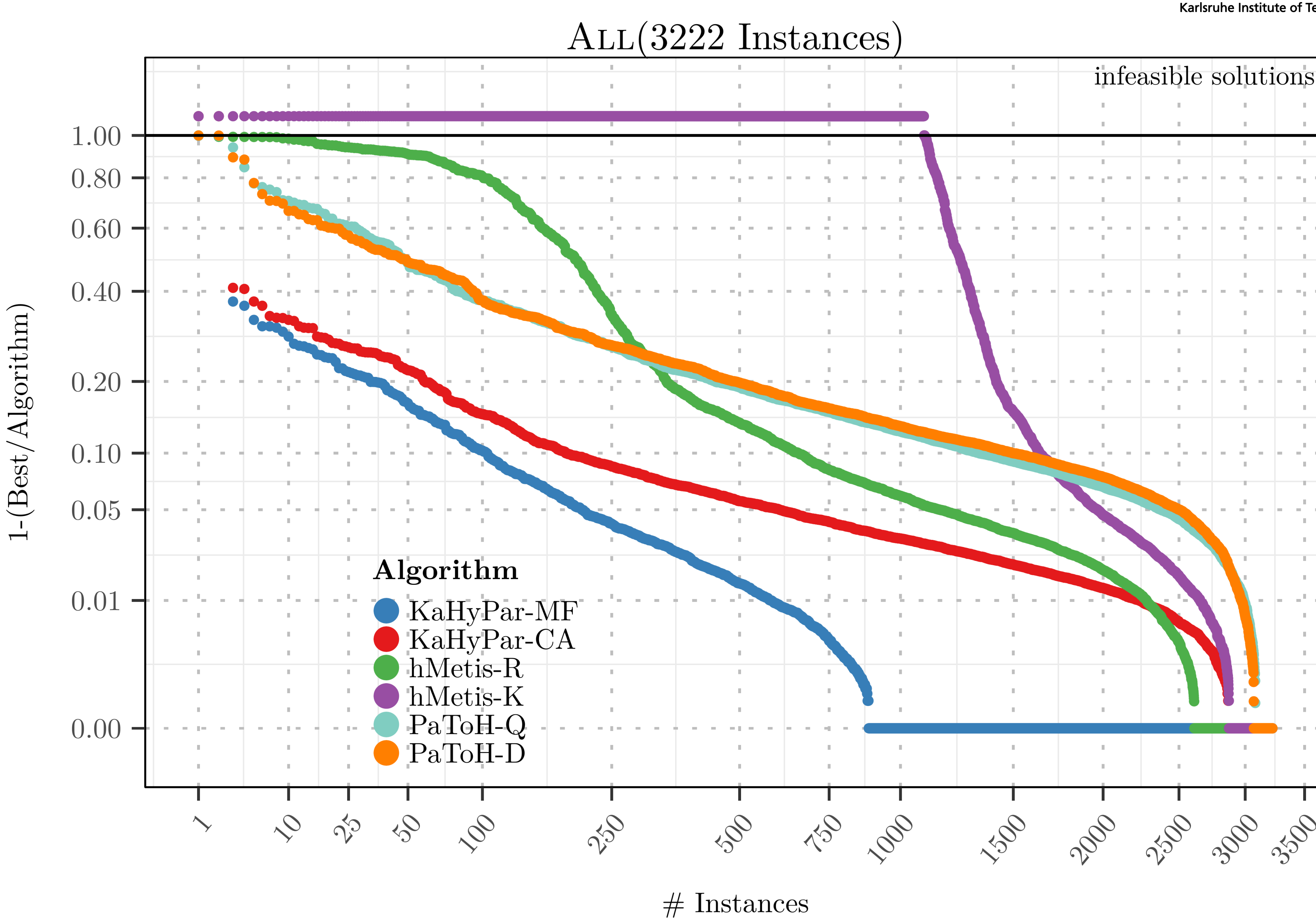
	Hypergraphs			DIMACS Graphs		
α'	$\varepsilon = 1\%$	$\varepsilon = 3\%$	$\varepsilon = 5\%$	$\varepsilon = 1\%$	$\varepsilon = 3\%$	$\varepsilon = 5\%$
1	7.7	8.1	7.6	11.7	11.3	10.5
2	7.9	6.6	4.8	11.0	9.1	7.8
4	6.9	3.9	2.7	9.9	7.3	5.4
8	5.1	2.3	1.5	8.6	5.3	3.9
16	3.4	1.3	1.2	7.0	4.1	3.5

$\Rightarrow$ performs **better** on **all** problem sizes and imbalances
$\Rightarrow$ most pronounced for **small** flow problems & imbalances
$\Rightarrow$ effects also visible for **graphs**

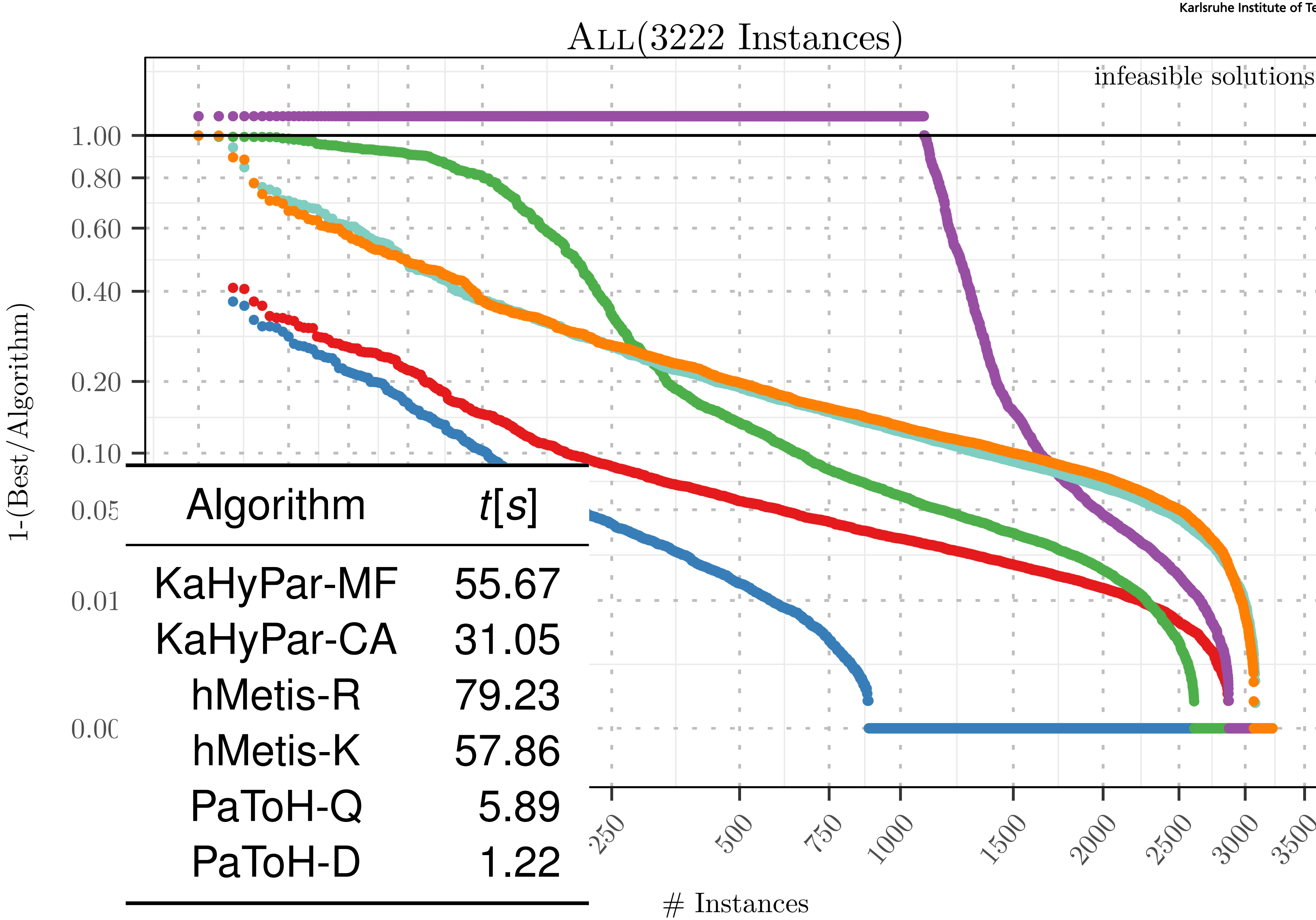
State-of-the-Art: HGP Quality



KaHyPar-MF: HGP Quality



KaHyPar-MF: HGP Quality & Running Time



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

Conclusion & Discussion

KaHyPar-MF – direct k-way HGP with **flow-based** refinement

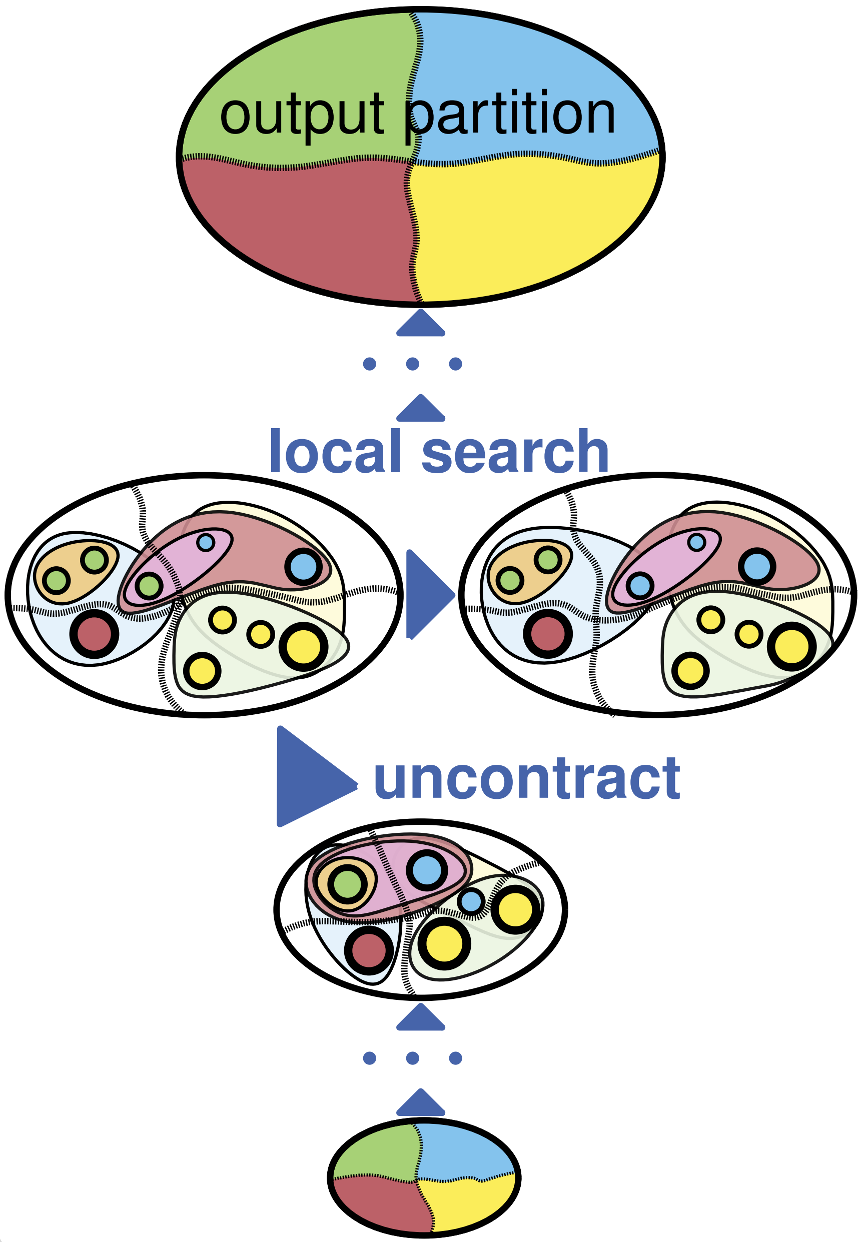
- generalizes KaFFPa's flow refinement to hypergraphs
- sparsified hypergraph flow networks
- improved flow model

In the paper / technical report:

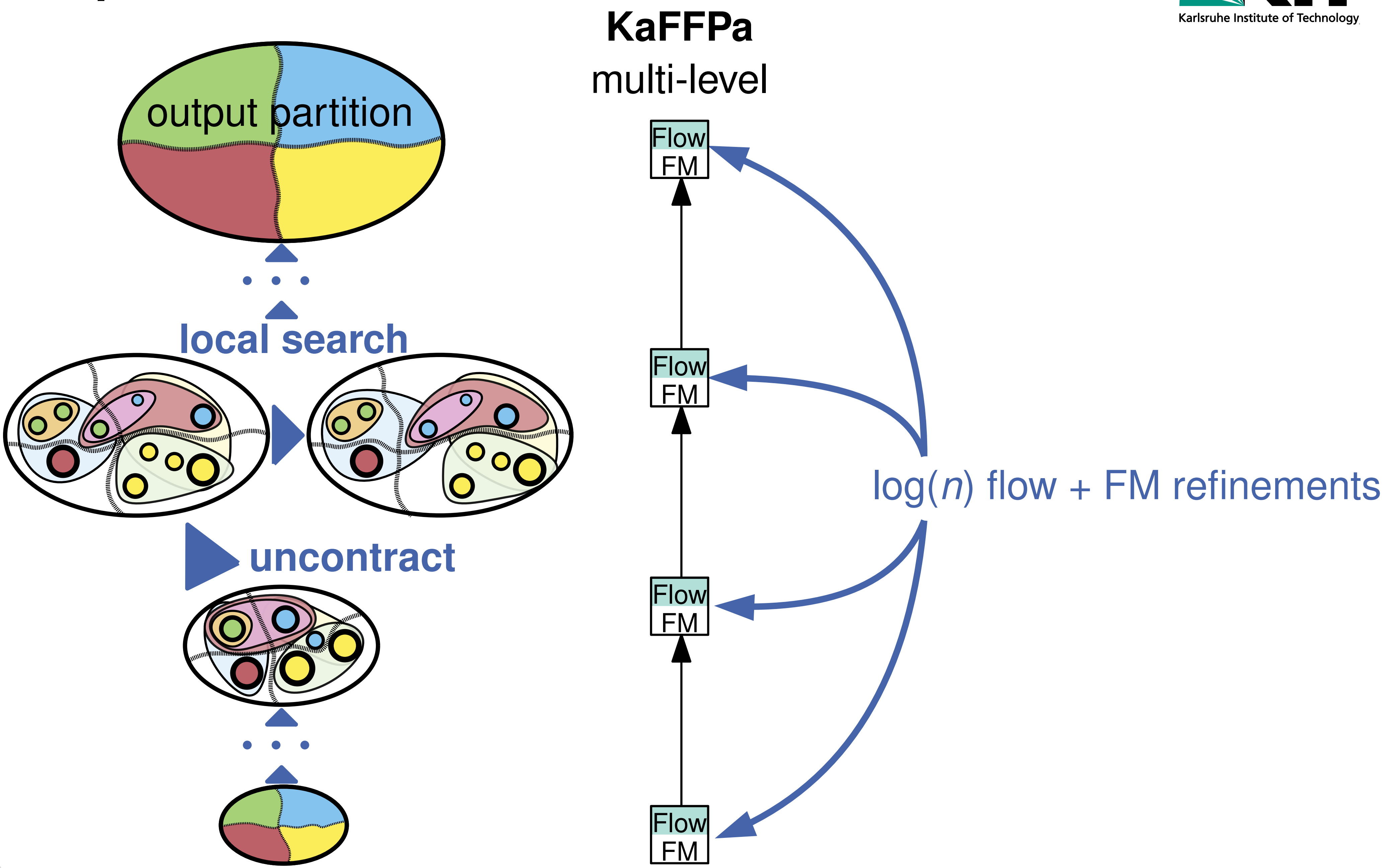
- speedup heuristics $\rightsquigarrow$ factor 2 faster
- min-cut reconstruction
- more experimental results:
  - size of flow networks
  - different algorithm configurations
  - quality & running times per instance class

KaHyPar-Framework
Open-Source:
`http://kahypar.org`

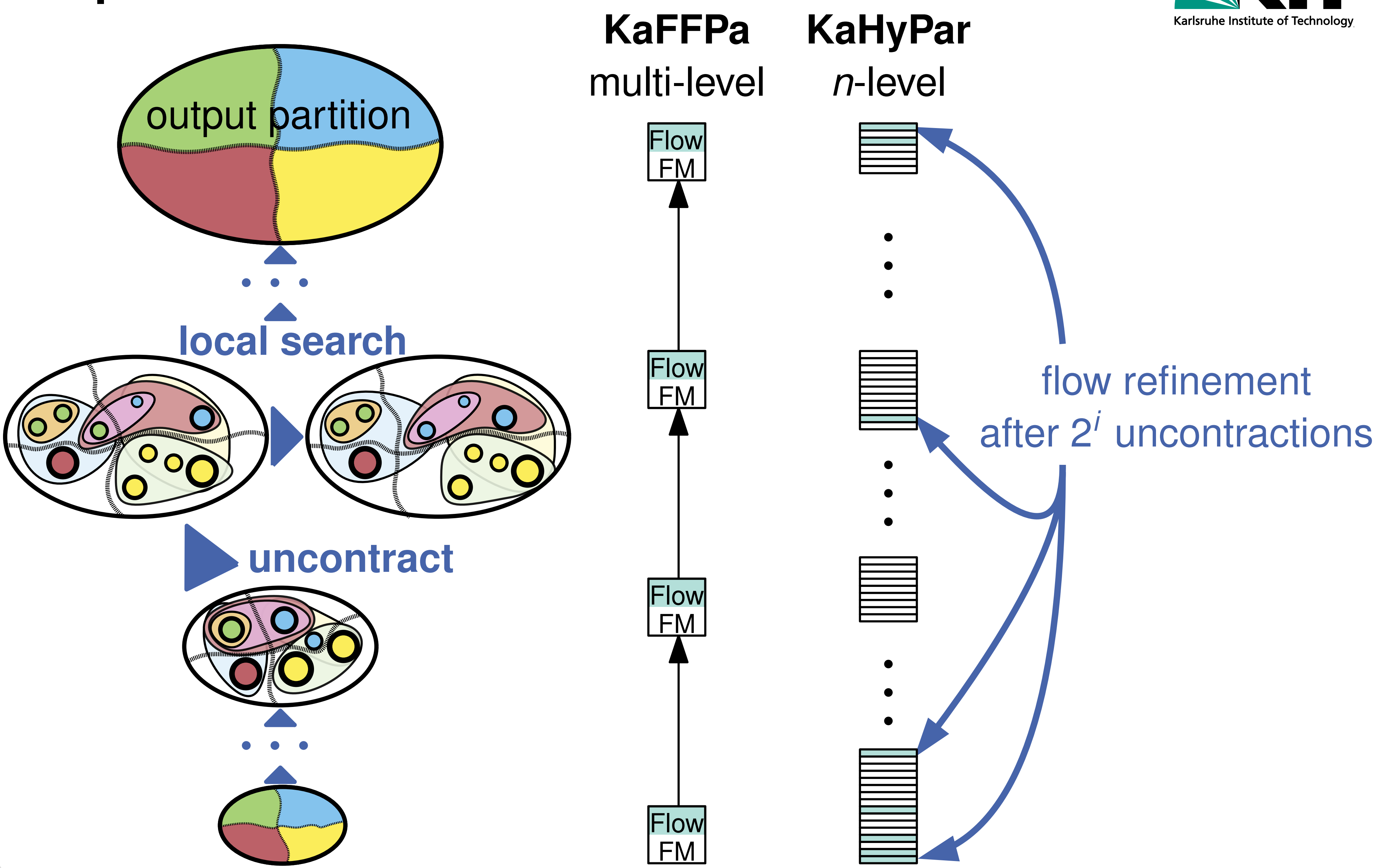
Implementation Details



Implementation Details



Implementation Details



Implementation Details