Karlsruhe Institute of Technology

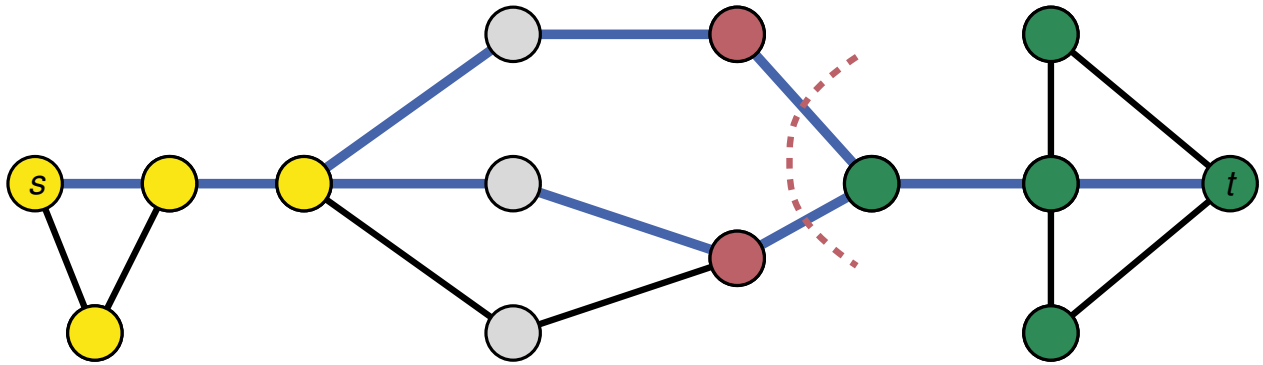
Advanced Flow-Based Multilevel Hypergraph Partitioning

SEA 2020

June 5, 2020

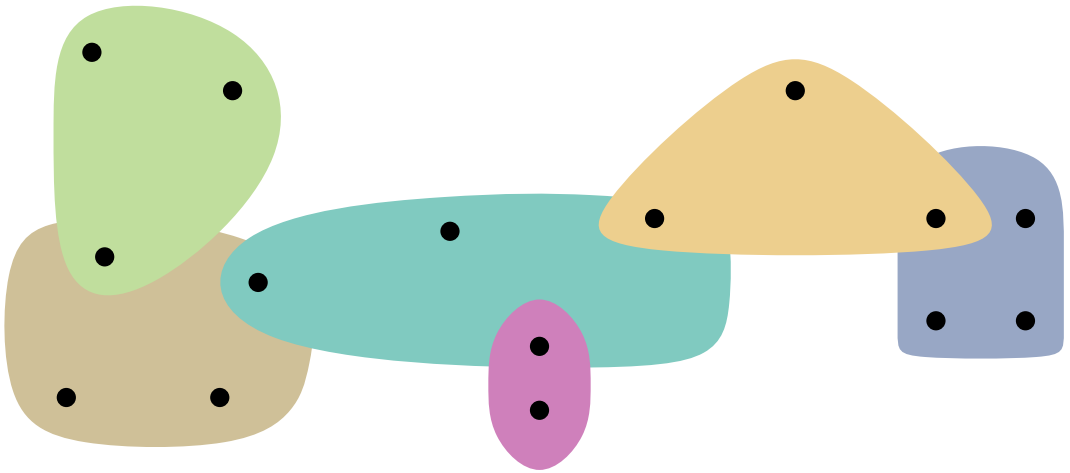
Lars Gottesbüren, Michael Hamann, Sebastian Schlag, Dorothea Wagner

INSTITUTE OF THEORETICAL INFORMATICS · ALGORITHMICS GROUP



Hypergraph Partitioning

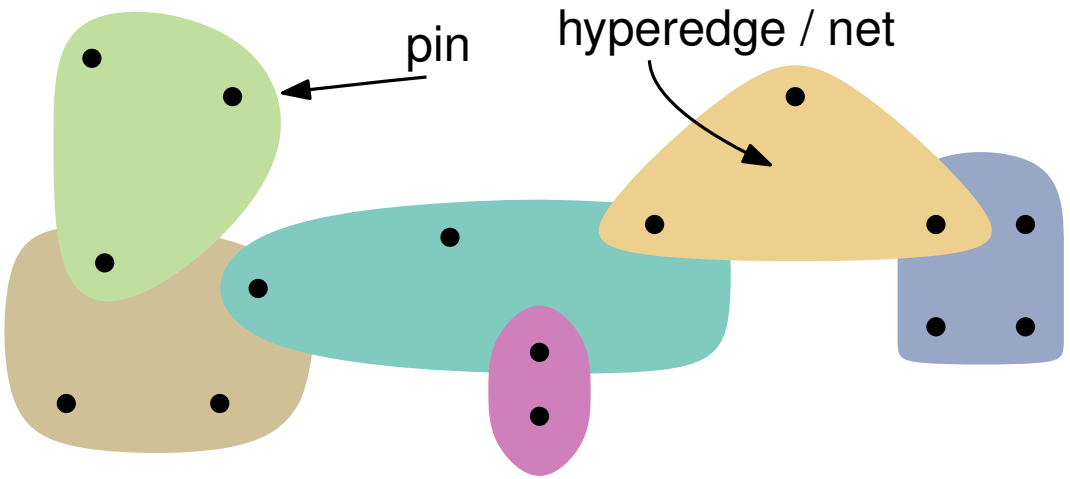
- Hypergraph $H = (V, E, c, \omega)$
  - vertex set $V = \{1, ..., n\}$
  - edge set $E \subseteq \mathcal{P}(V) \setminus \emptyset$
  - incident edges $\Gamma(u) = \{e \in E \mid u \in e\}$
  - vertex weights $\varphi : V \to \mathbb{R}_{\geq 1}$
  - edge weights $\omega : E \to \mathbb{R}_{\geq 1}$



Institute of Theoretical Informatics
Algorithmics Group

Hypergraph Partitioning

- Hypergraph $H = (V, E, c, \omega)$
  - vertex set $V = \{1, ..., n\}$
  - edge set $E \subseteq \mathcal{P}(V) \setminus \emptyset$
  - incident edges $\Gamma(u) = \{e \in E \mid u \in e\}$
  - vertex weights $\varphi : V \to \mathbb{R}_{\geq 1}$
  - edge weights $\omega : E \to \mathbb{R}_{\geq 1}$



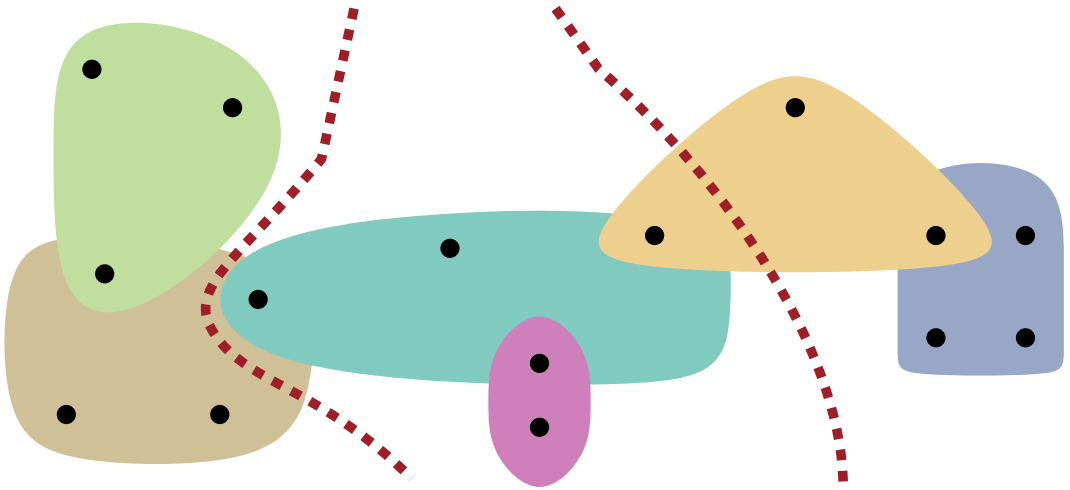
Institute of Theoretical Informatics
Algorithmics Group

Hypergraph Partitioning

Partition into $\mathbf{k}$ disjoint blocks $\Pi = \{V_1, \ldots, V_k\}$

- blocks V_i have **roughly equal weight**:

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



Institute of Theoretical Informatics
Algorithmics Group

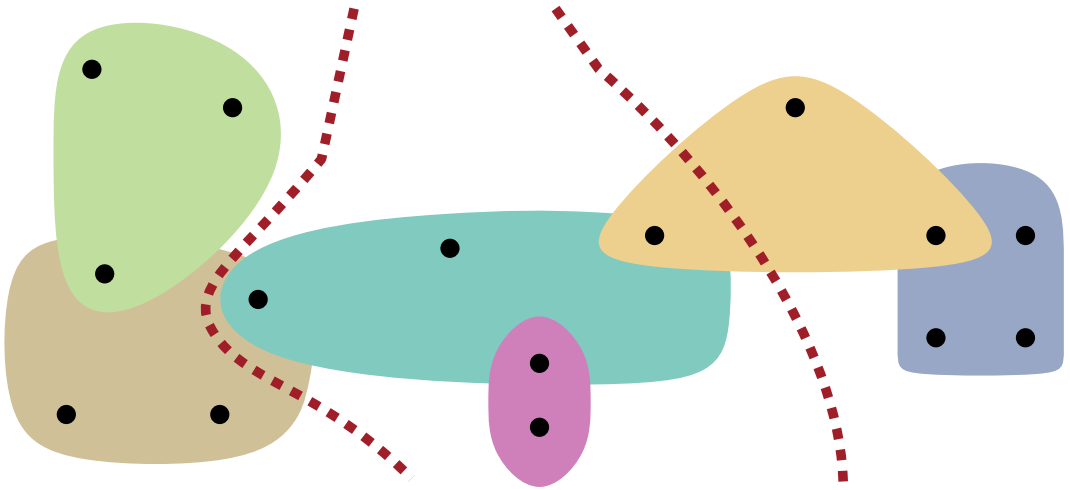
Hypergraph Partitioning

Partition into $\mathbf{k}$ disjoint blocks $\Pi = \{V_1, \ldots, V_k\}$

- blocks V_i have **roughly equal weight**:

imbalance

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



Institute of Theoretical Informatics
Algorithmics Group

Hypergraph Partitioning

Partition into $\mathbf{k}$ disjoint blocks $\Pi = \{V_1, \ldots, V_k\}$

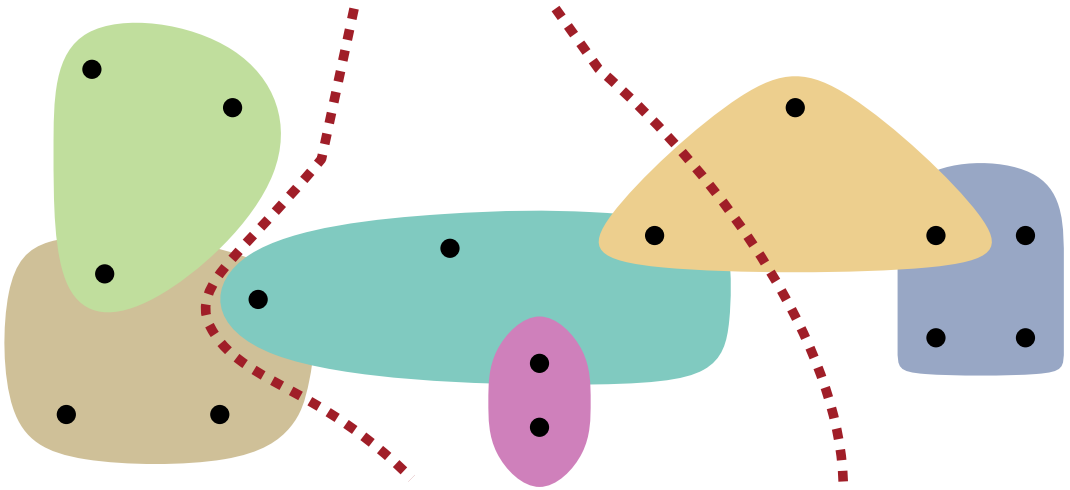
- blocks V_i have **roughly equal weight**:

imbalance

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

- **minimize connectivity** objective:

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



Institute of Theoretical Informatics
Algorithmics Group

Hypergraph Partitioning

Partition into $\mathbf{k}$ disjoint blocks $\Pi = \{V_1, \ldots, V_k\}$

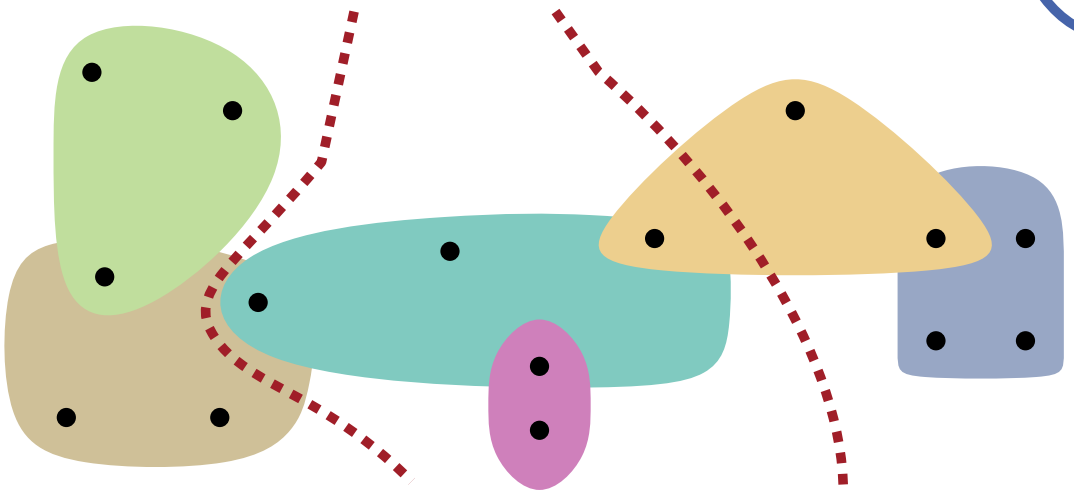
- blocks V_i have **roughly equal weight**:

imbalance

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

- **minimize connectivity** objective:

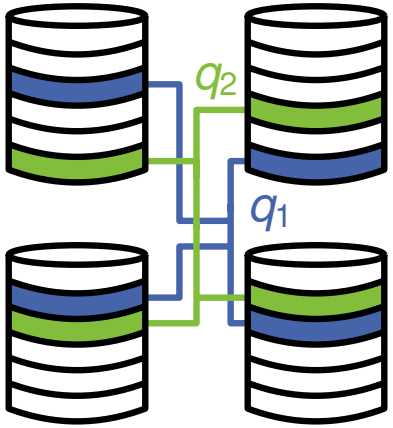
$$\text{con} = \sum_{e \in E} (\lambda(e) - 1)\, \omega(e)$$



$\lambda(e) = |\{V_i \mid V_i \cap e \neq \emptyset\}|$
blocks overlapping with e

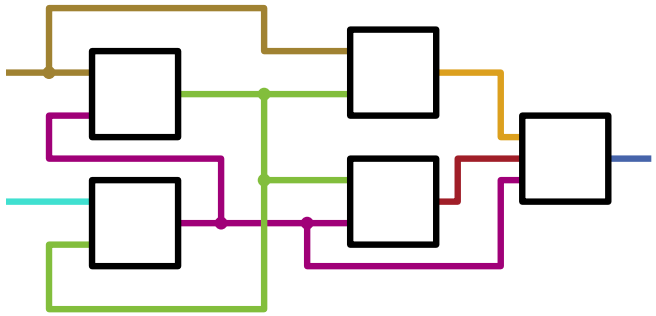
Institute of Theoretical Informatics
Algorithmics Group

Applications



Distributed Databases

Route Planning

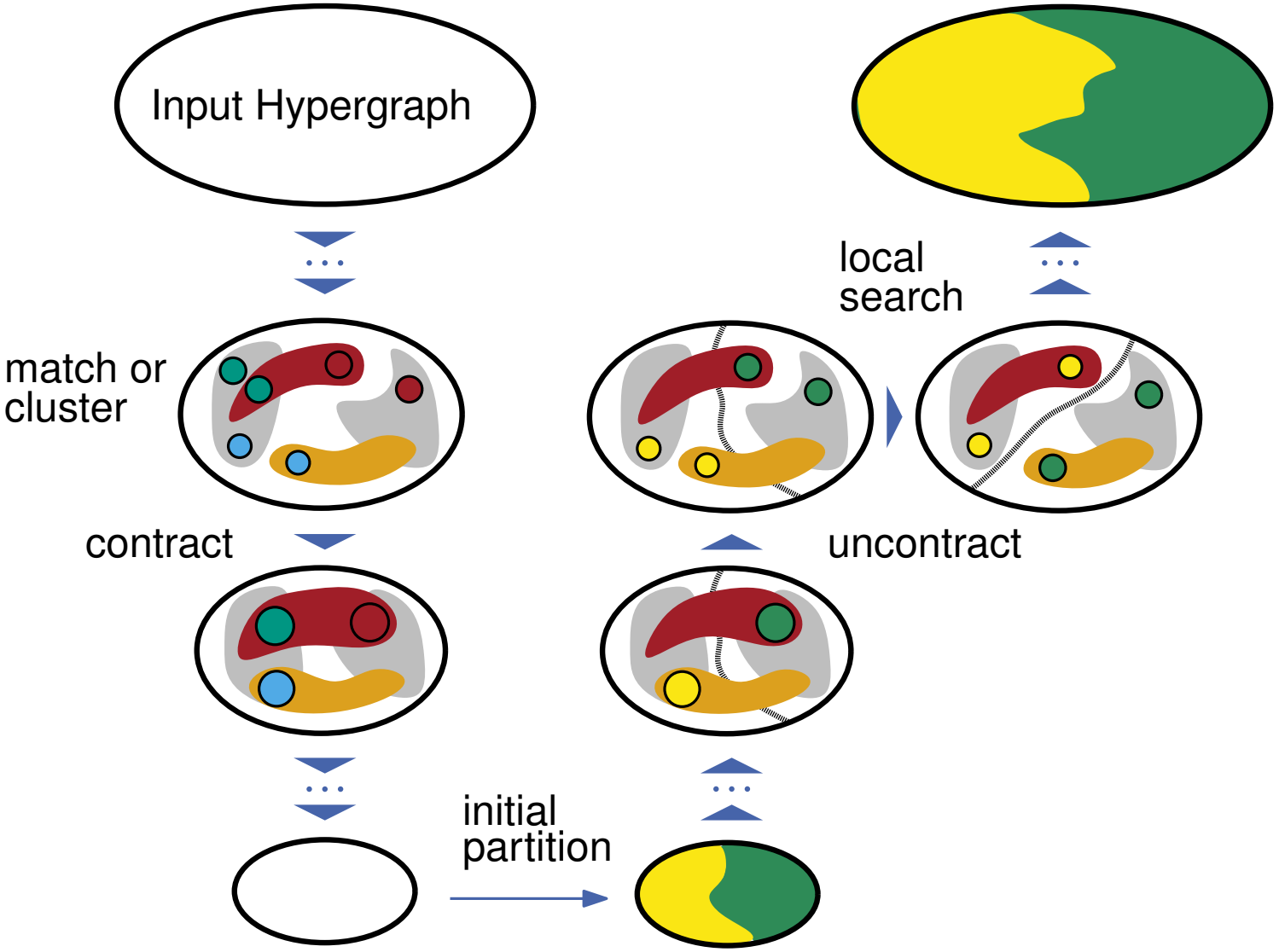


VLSI Design

HPC

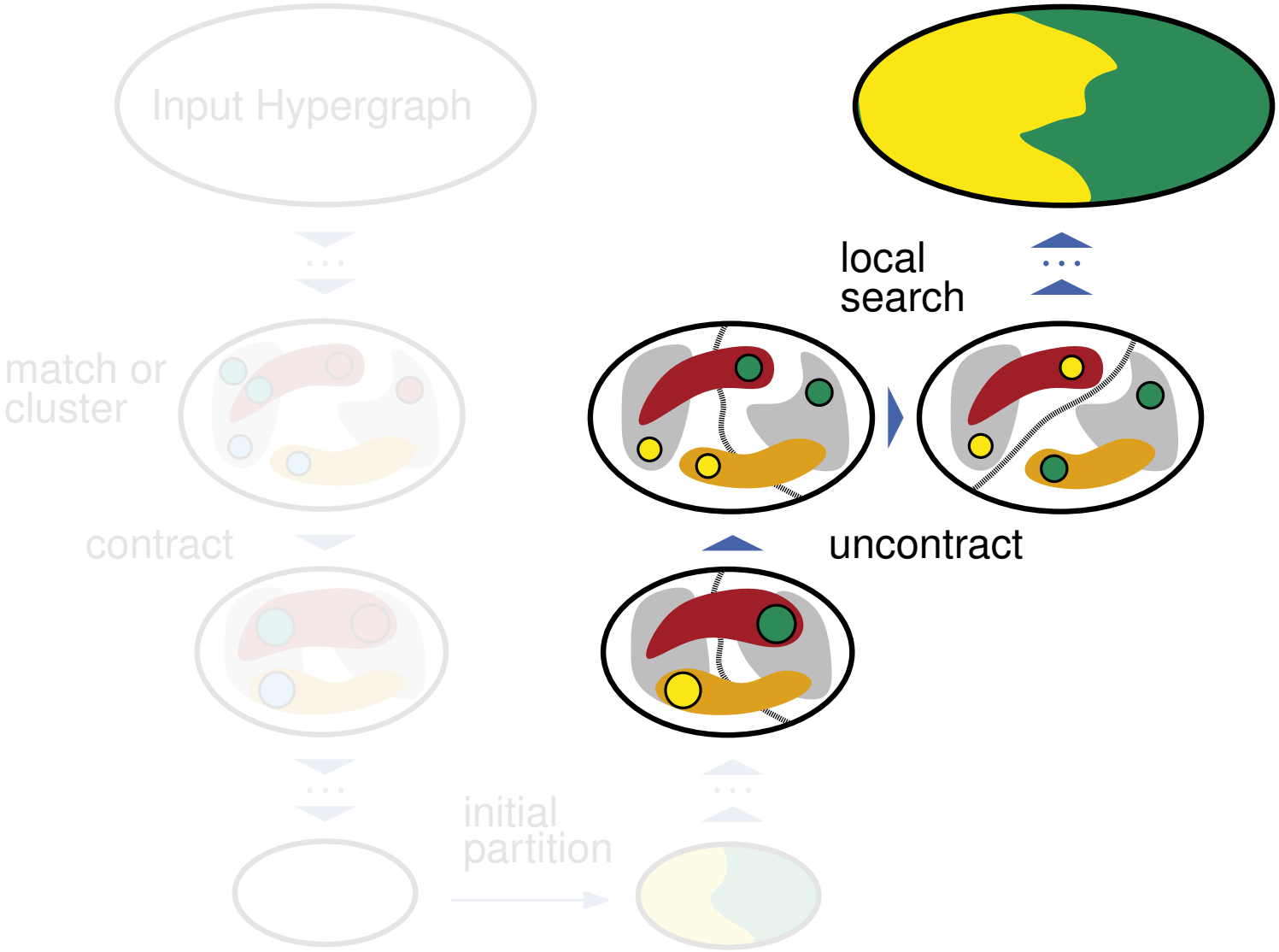
Institute of Theoretical Informatics
Algorithmics Group

Multilevel Algorithms



Institute of Theoretical Informatics
Algorithmics Group

Multilevel Algorithms

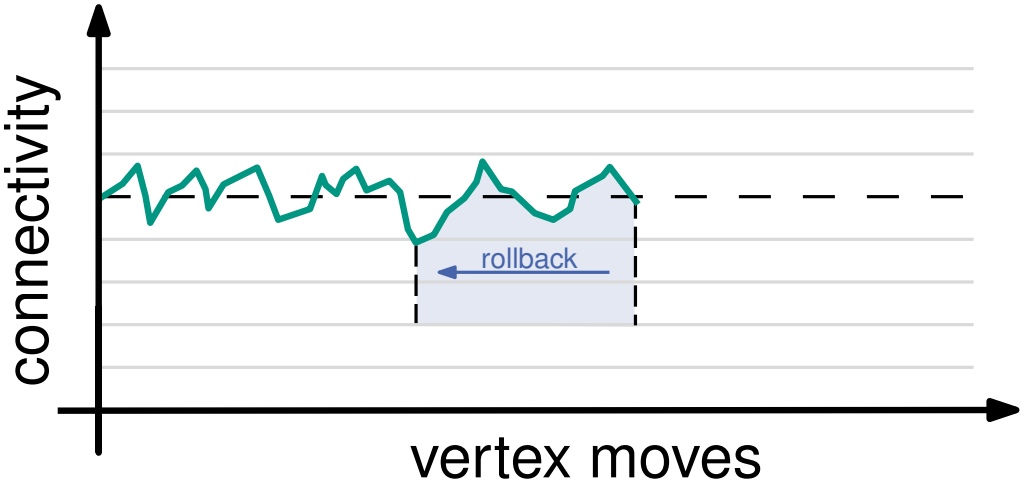


Classic Fiduccia-Mattheyses

Algorithm 1: FM Local Search

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

pass



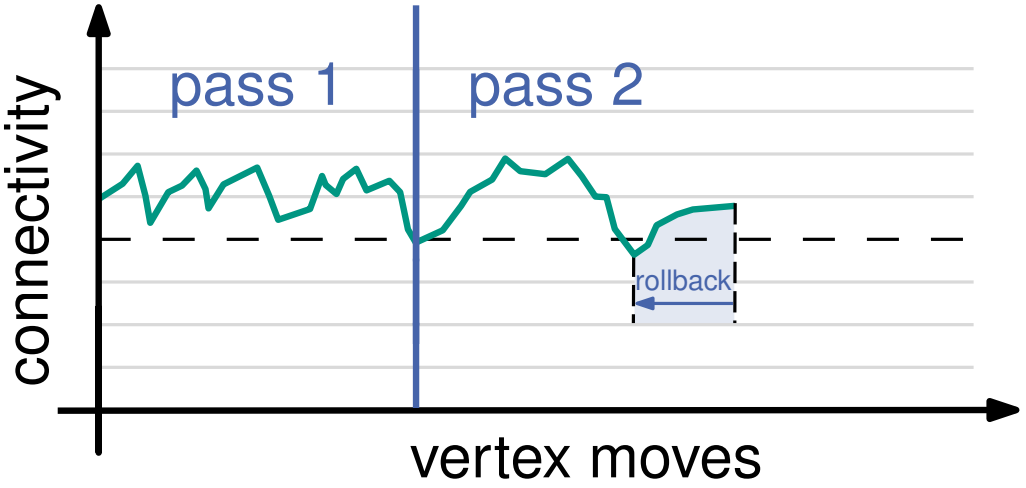
slide kindly provided by Sebastian Schlag

Classic Fiduccia-Mattheyses

Algorithm 1: FM Local Search

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

pass



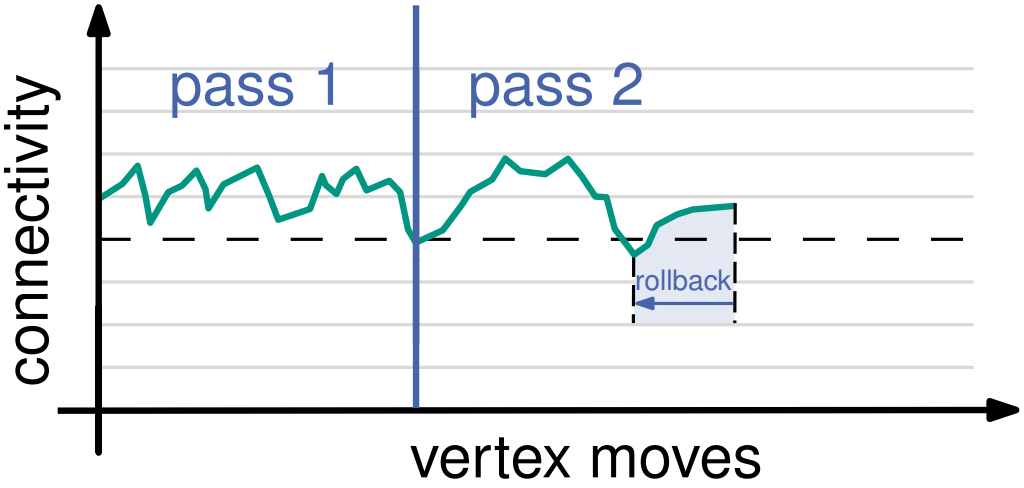
slide kindly provided by Sebastian Schlag

Classic Fiduccia-Mattheyses

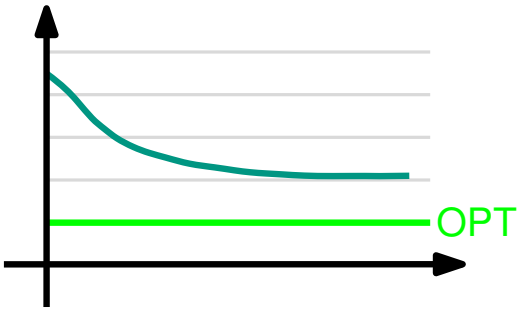
Algorithm 1: FM Local Search

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

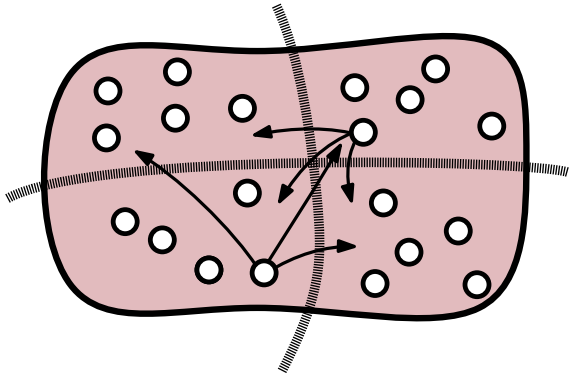
pass



✗ get **stuck** in local optima



✗ large edges ⇝ **zero** gain moves



slide kindly provided by Sebastian Schlag

Institute of Theoretical Informatics
Algorithmics Group

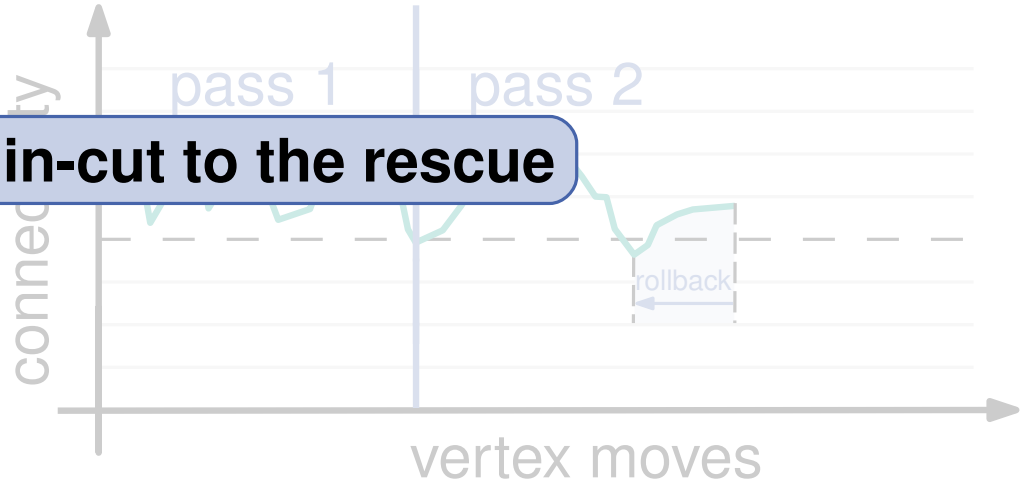
Classic Fiduccia-Mattheyses

Algorithm 1: FM Local Search

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

pass

max-flow-min-cut to the rescue



✗ get **stuck** in local optima

✗ large edges ⇝ **zero** gain moves

slide kindly provided by Sebastian Schlag

Institute of Theoretical Informatics
Algorithmics Group

Classic Fiduccia-Mattheyses

max-flow-min-cut to the rescue

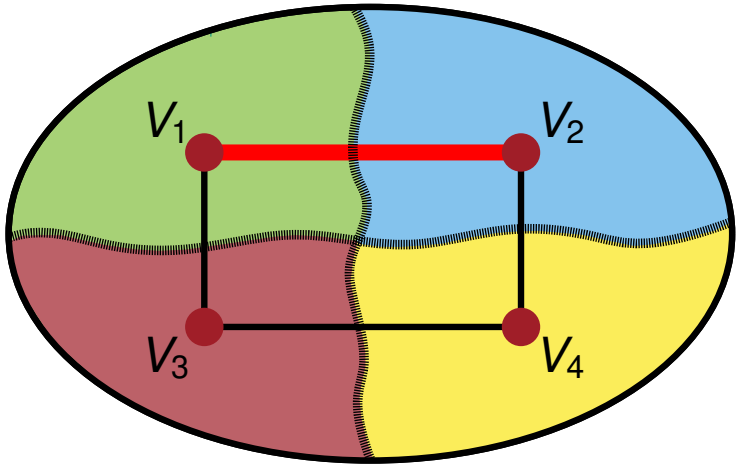
Issues?

- only 2-way
- what are flows on hypergraphs?
- not balanced

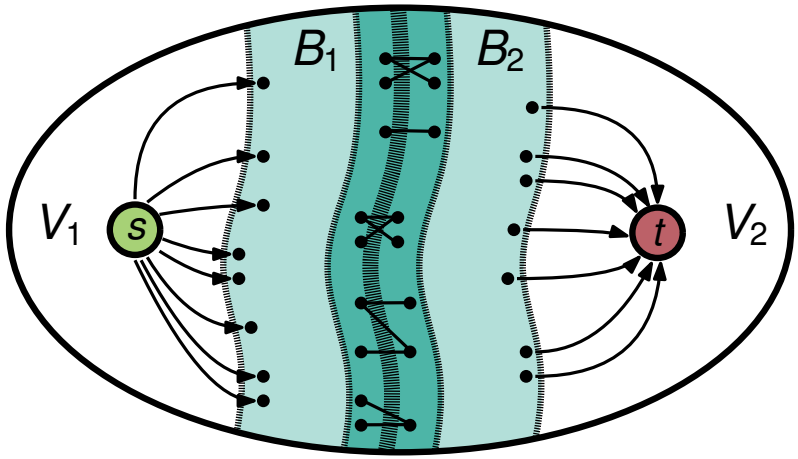
slide kindly provided by Sebastian Schlag

Institute of Theoretical Informatics
Algorithmics Group

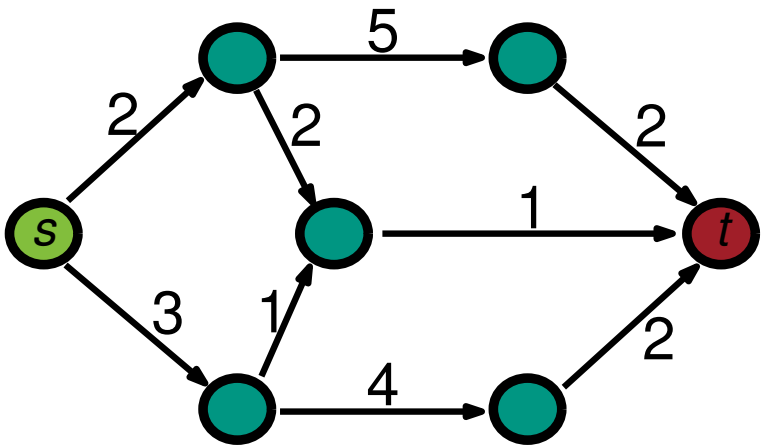
Flow-Based Refinement in KaHyPar



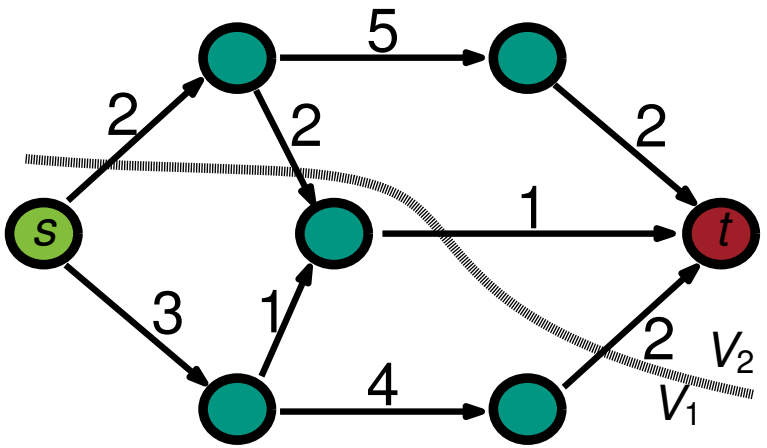
select two adjacent blocks for refinement



build graph-based flow model



solve flow problem

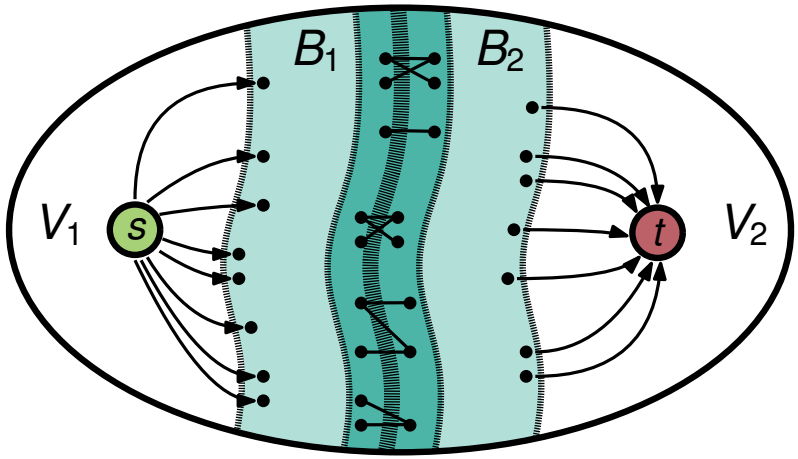


find more balanced minimum cut

slide kindly provided by Sebastian Schlag

Institute of Theoretical Informatics
Algorithmics Group

Flow-Based Refinement in KaHyPar

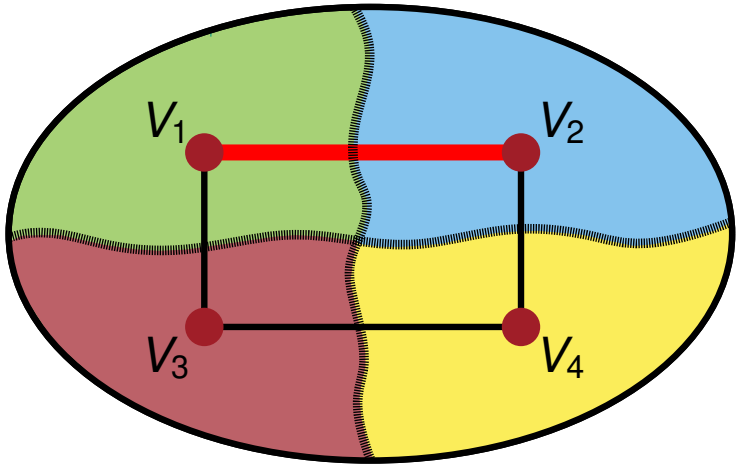


build graph-based flow model

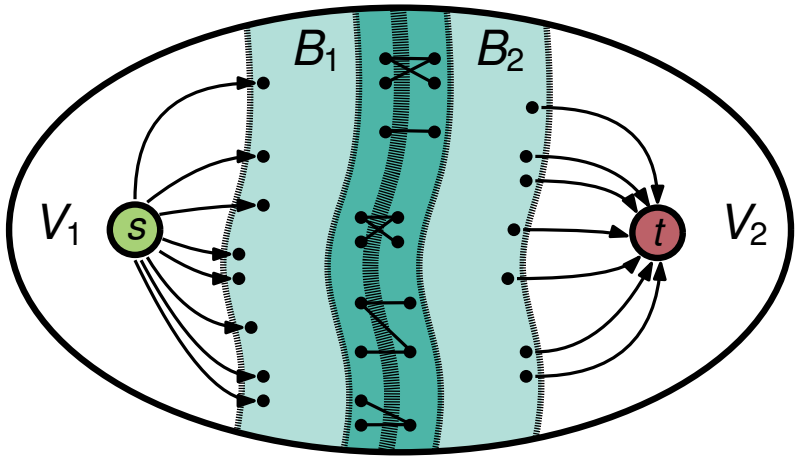
- either: restrict flow model size so that balance is guaranteed
- or: make it a little larger, hope for balance. if not $\rightsquigarrow$ scale down again

slide kindly provided by Sebastian Schlag

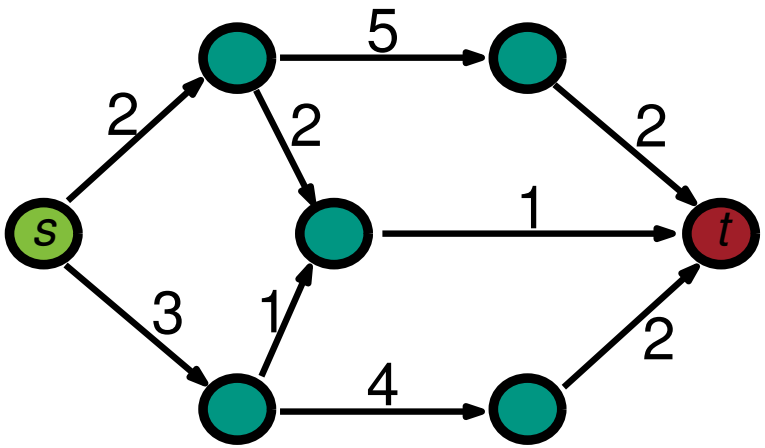
Flow-Based Refinement in KaHyPar



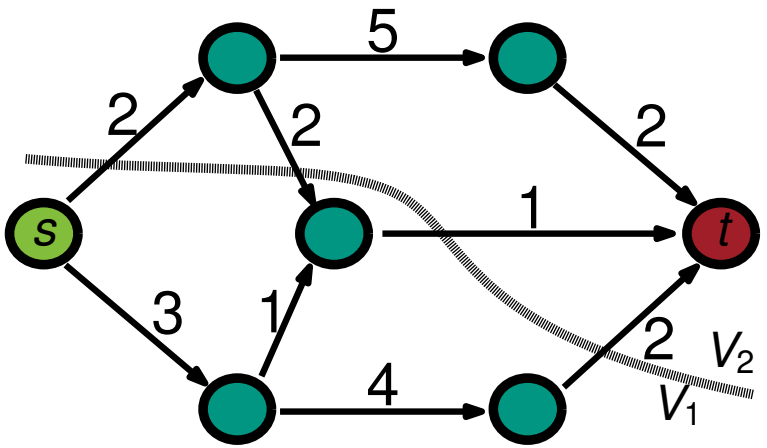
select two adjacent blocks for refinement



build graph-based flow model



solve flow problem

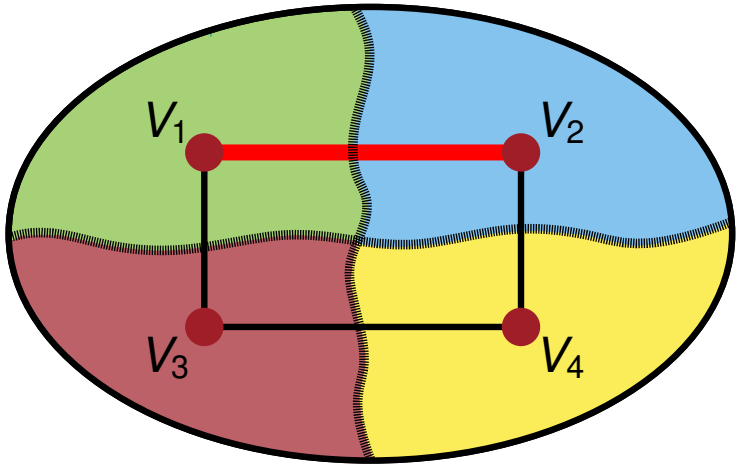


find more balanced minimum cut

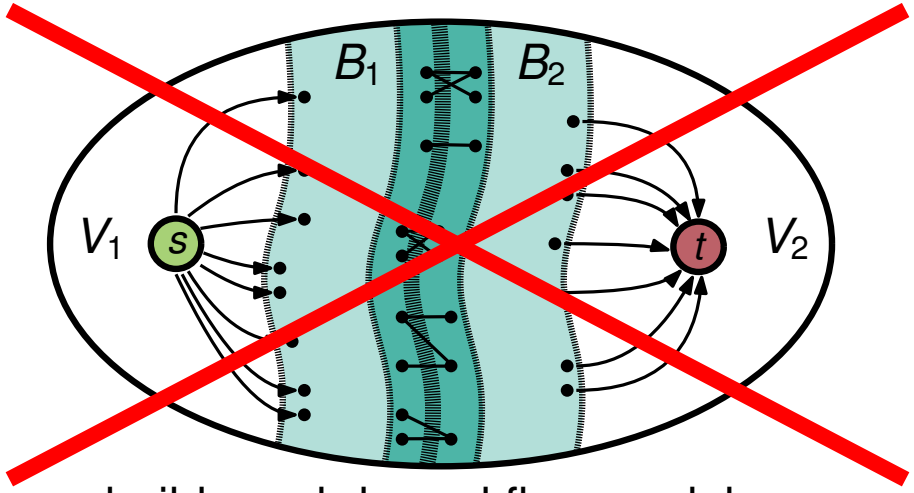
slide kindly provided by Sebastian Schlag

Institute of Theoretical Informatics
Algorithmics Group

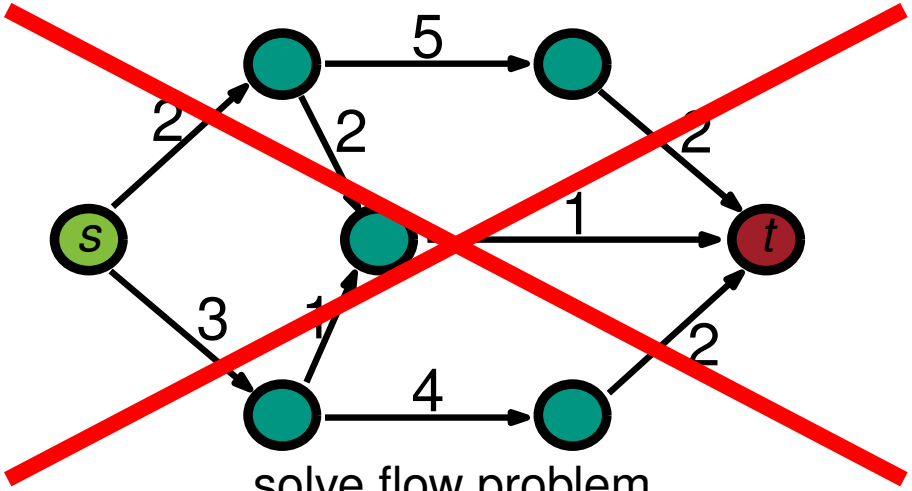
Flow-Based Refinement in KaHyPar



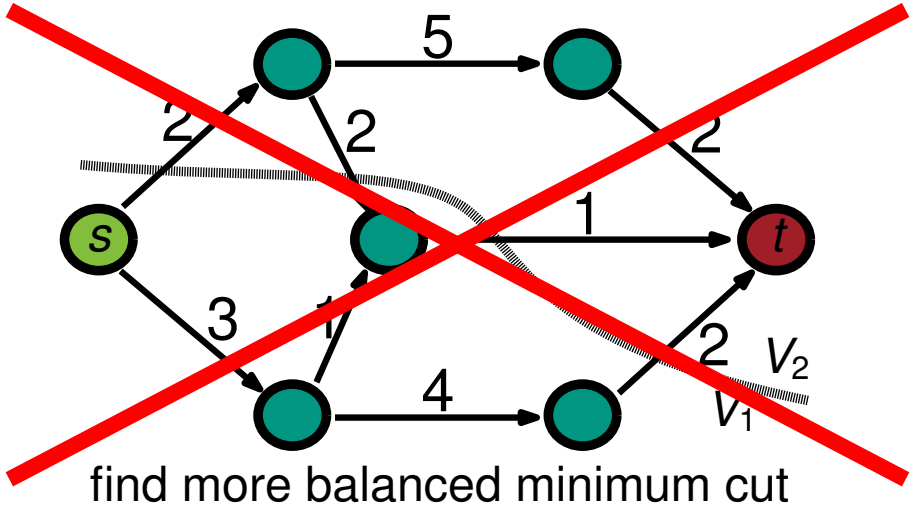
select two adjacent blocks for refinement



build graph-based flow model



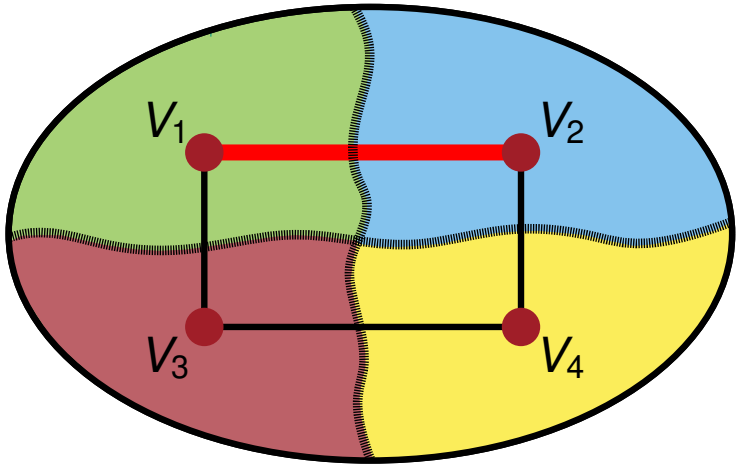
solve flow problem



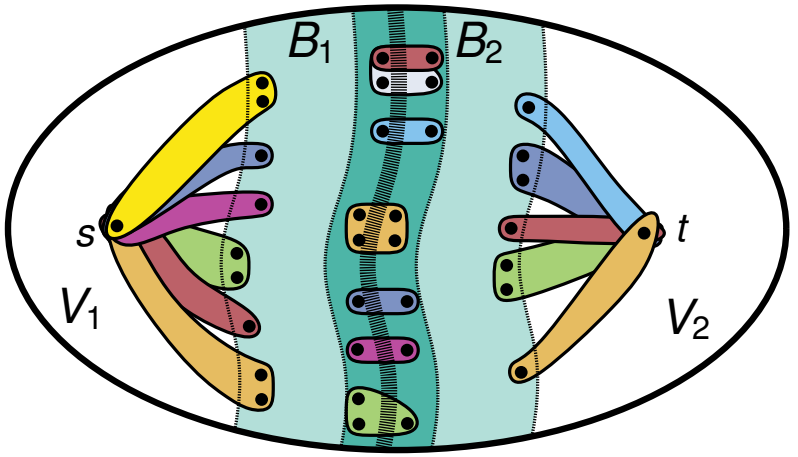
find more balanced minimum cut

slide kindly provided by Sebastian Schlag

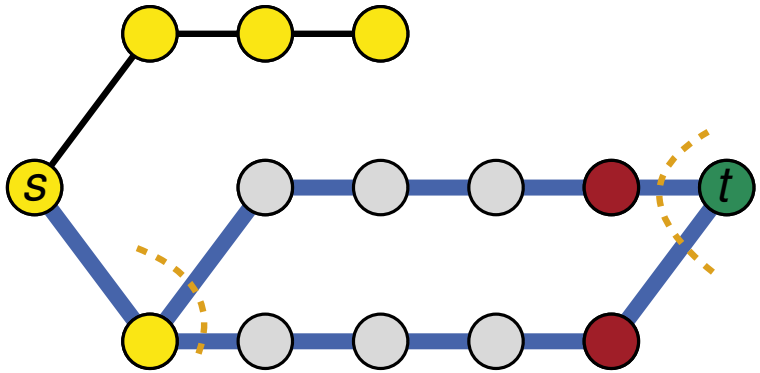
The new KaHyPar-HFC



select two adjacent blocks for refinement



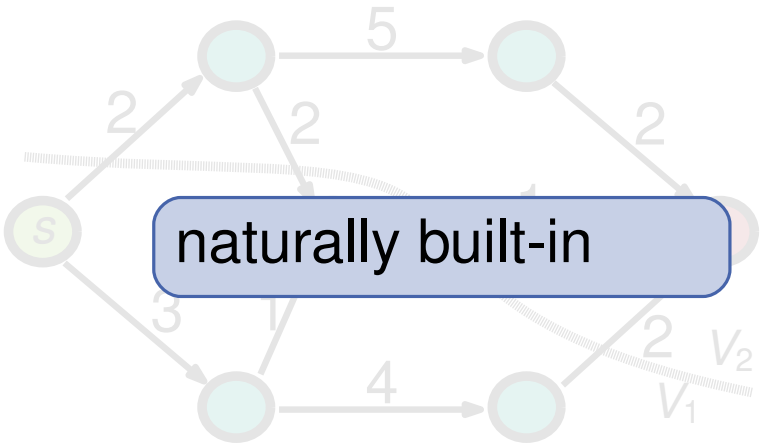
flows directly on hypergraph



use FlowCutter

[Hamann, Strasser JEA18]

[Gottesbüren, Hamann, Wagner ESA19]

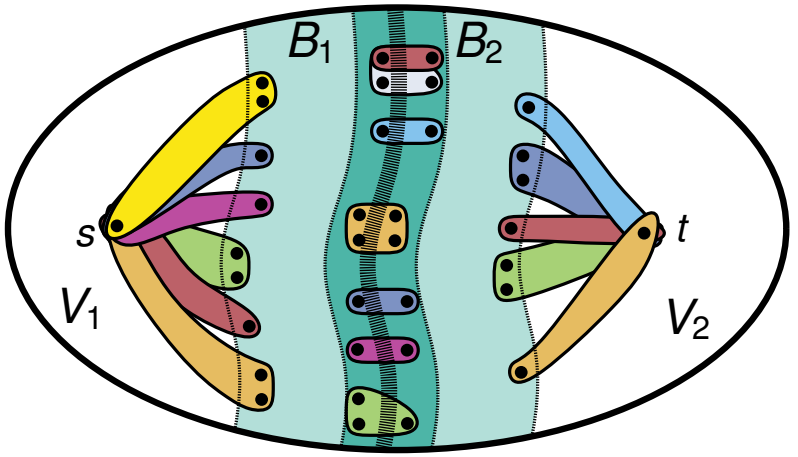


find more balanced minimum cut

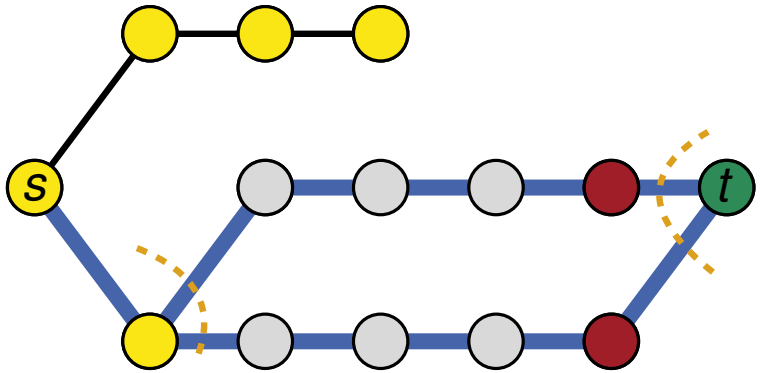
Institute of Theoretical Informatics
Algorithmics Group

The new KaHyPar-HFC

select two adjacent blocks for refinement



flows directly on hypergraph



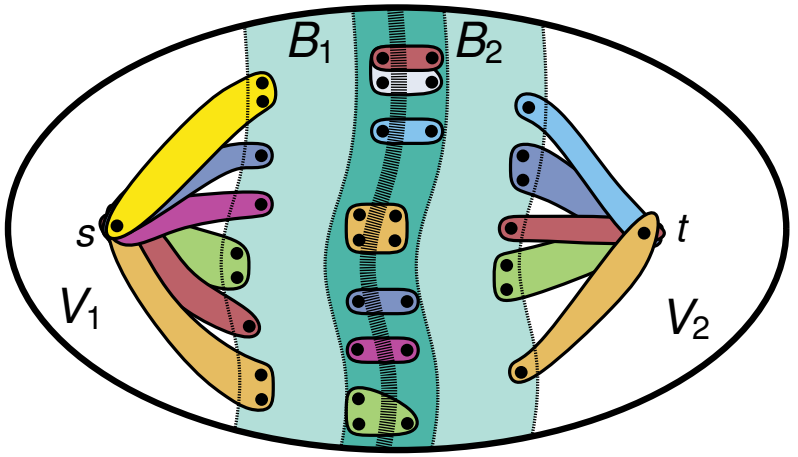
use FlowCutter
[Hamann, Strasser JEA18]
[Gottesbüren, Hamann, Wagner ESA19]

find more balanced minimum cut

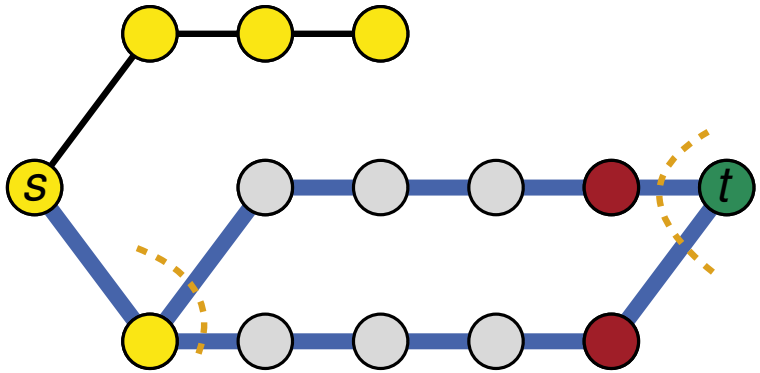
Institute of Theoretical Informatics
Algorithmics Group

The new KaHyPar-HFC

select two adjacent blocks for refinement



flows directly on hypergraph



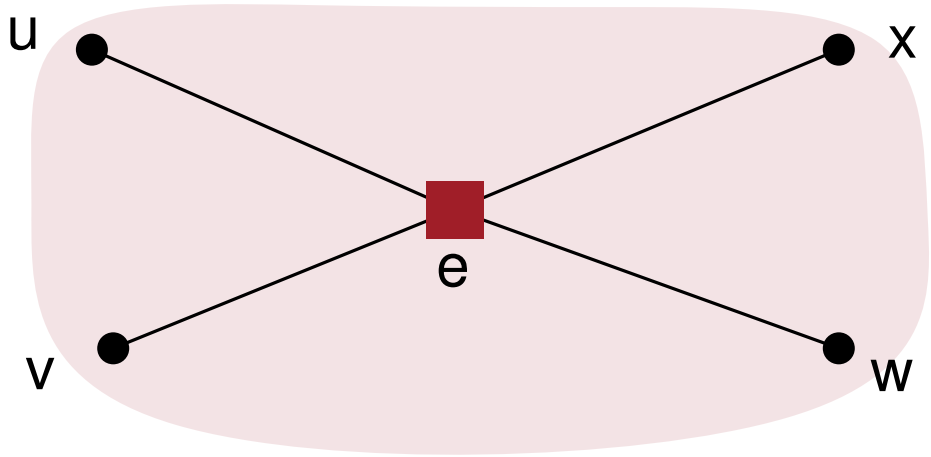
use FlowCutter
[Hamann, Strasser JEA18]
[Gottesbüren, Hamann, Wagner ESA19]

what's new for FlowCutter?

- weighted instances
- new guidance

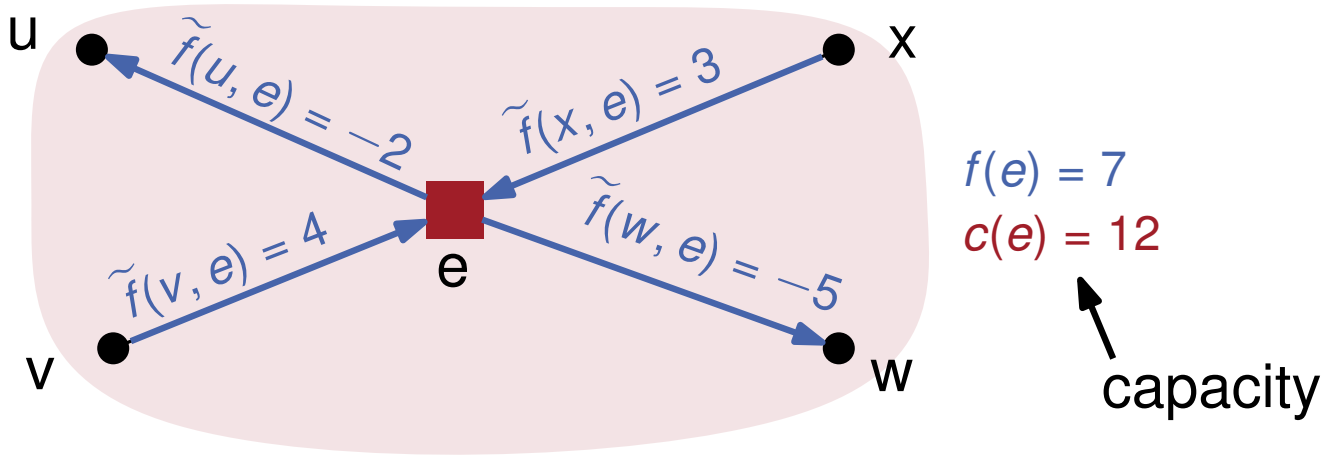
Institute of Theoretical Informatics
Algorithmics Group

Flows on Hypergraphs



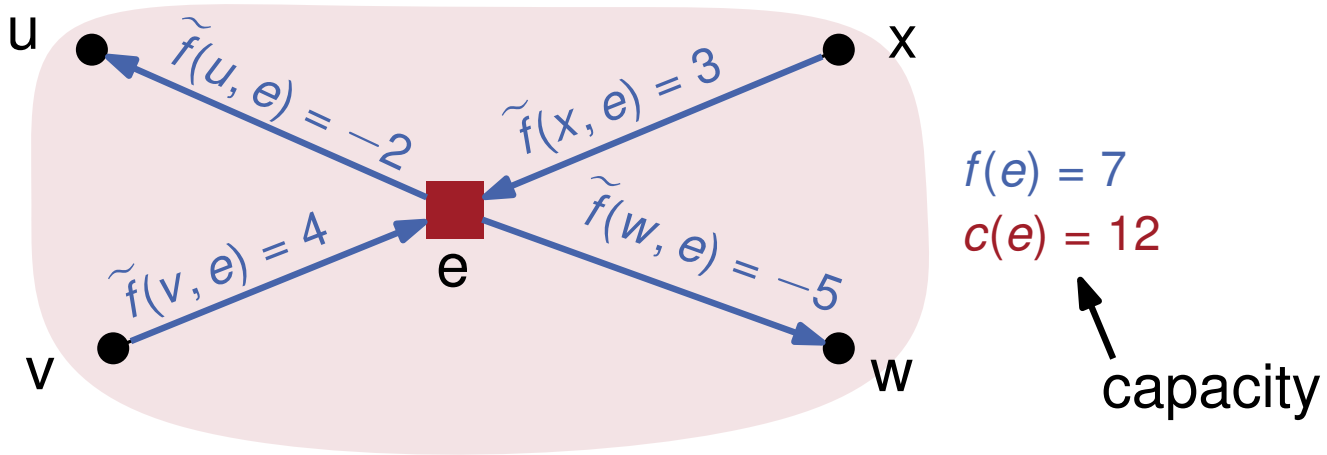
Institute of Theoretical Informatics
Algorithmics Group

Flows on Hypergraphs



- $\tilde{f}(u, e) > 0 \Rightarrow u$ sends flow into e
- $\tilde{f}(u, e) < 0 \Rightarrow u$ receives flow from e

Flows on Hypergraphs

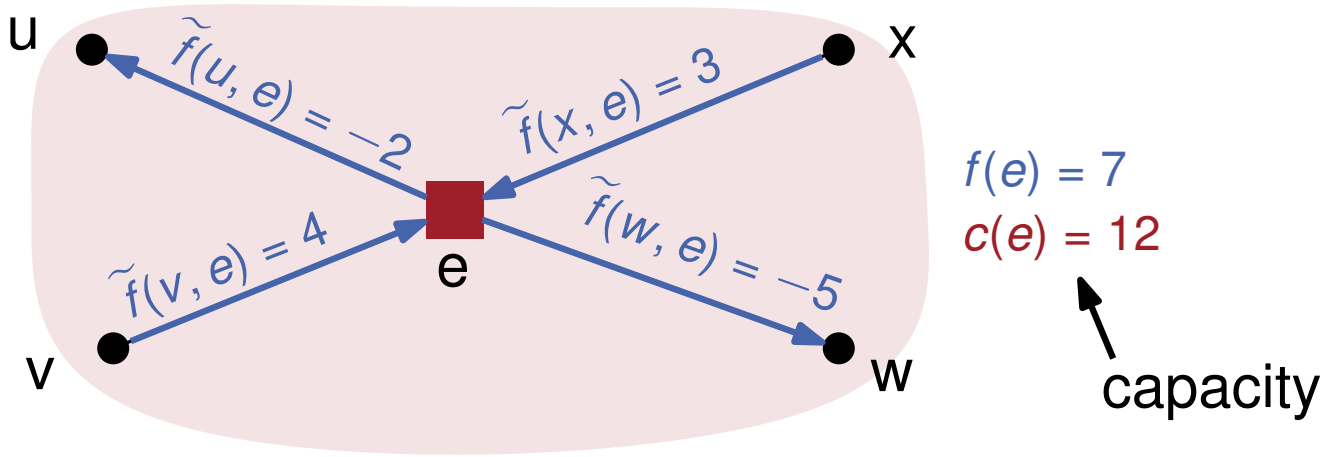


- $\widetilde{f}(u,e) > 0 \Rightarrow u$ sends flow into e
- $\widetilde{f}(u,e) < 0 \Rightarrow u$ receives flow from e
- $\text{rcap}(e,u,v) = c(e) - f(e) + \widetilde{f}(u,e)^- + \widetilde{f}(v,e)^+ = (12-7) + 2 + 4 = 11$

  $\max(0, -\widetilde{f}(u,e))$ $\max(0, \widetilde{f}(v,e))$

Institute of Theoretical Informatics
Algorithmics Group

Flows on Hypergraphs

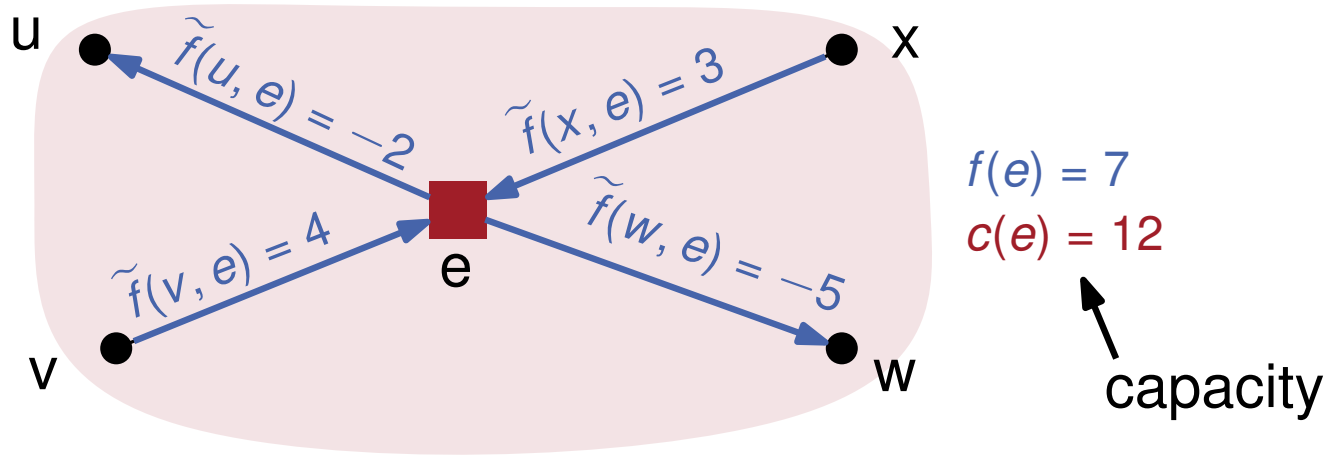


- $\tilde{f}(u, e) > 0 \Rightarrow u$ sends flow into e
- $\tilde{f}(u, e) < 0 \Rightarrow u$ receives flow from e
- $\text{rcap}(e, u, v) = c(e) - f(e) + \tilde{f}(u, e)^- + \tilde{f}(v, e)^+ = (12 - 7) + 2 + 4 = 11$

  $\max(0, -\tilde{f}(u, e))$ $\max(0, \tilde{f}(v, e))$

Institute of Theoretical Informatics
Algorithmics Group

Flows on Hypergraphs

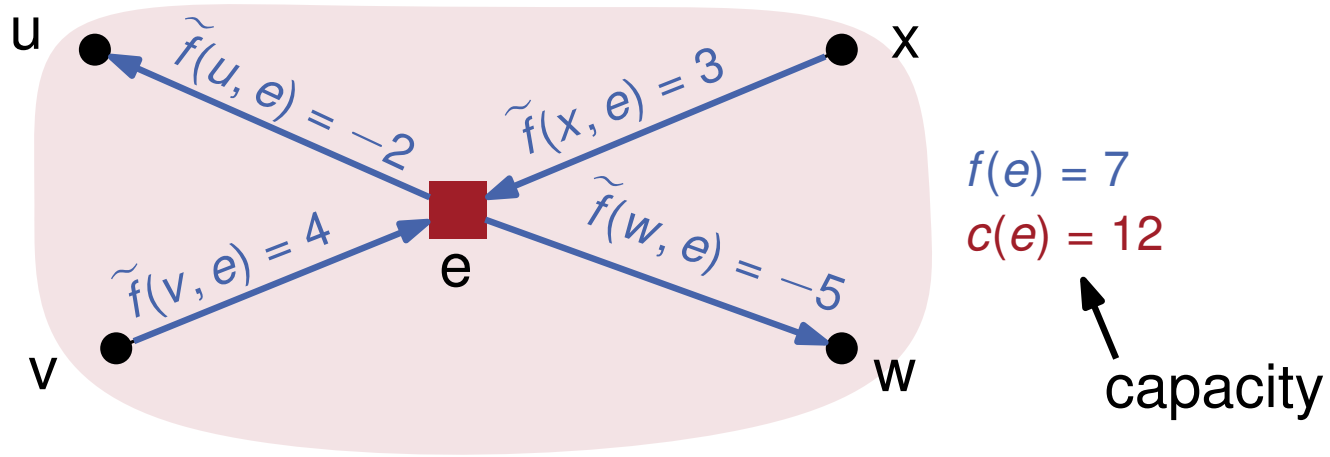


- $\widetilde{f}(u,e) > 0 \Rightarrow u$ sends flow into e
- $\widetilde{f}(u,e) < 0 \Rightarrow u$ receives flow from e
- $\text{rcap}(e,u,v) = c(e) - f(e) + \widetilde{f}(u,e)^- + \widetilde{f}(v,e)^+ = (12-7) + 2 + 4 = 11$

$\max(0, -\widetilde{f}(u,e))$ $\max(0, \widetilde{f}(v,e))$

Institute of Theoretical Informatics
Algorithmics Group

Flows on Hypergraphs

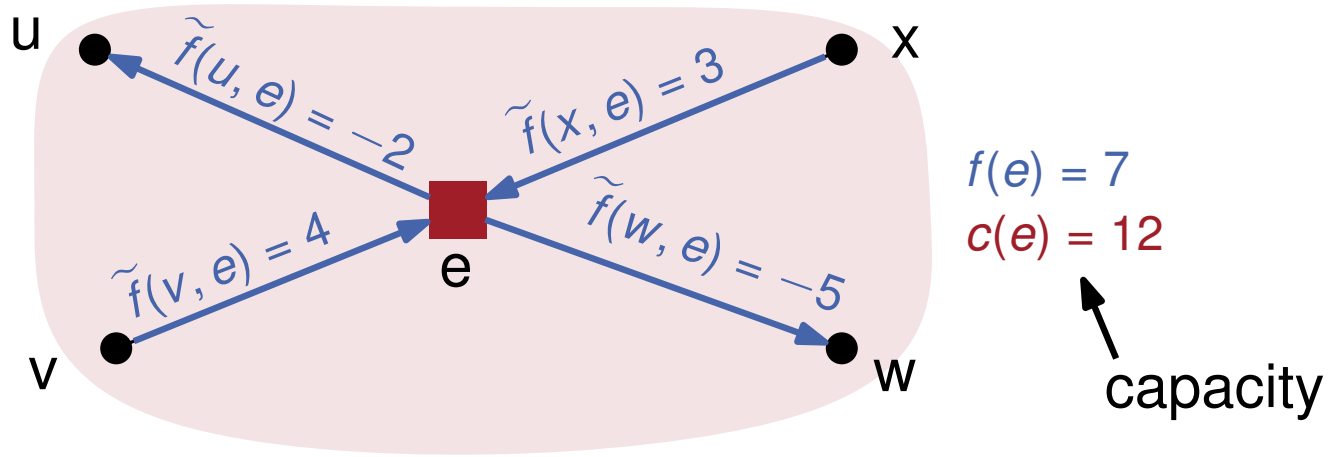


- $\tilde{f}(u,e) > 0 \Rightarrow u$ sends flow into e
- $\tilde{f}(u,e) < 0 \Rightarrow u$ receives flow from e
- $\text{rcap}(e,u,v) = c(e) - f(e) + \tilde{f}(u,e)^- + \tilde{f}(v,e)^+ = (12-7) + 2 + 4 = 11$

  $\max(0, -\tilde{f}(u,e))$ $\max(0, \tilde{f}(v,e))$

Institute of Theoretical Informatics
Algorithmics Group

Flows on Hypergraphs



- $\tilde{f}(u,e) > 0 \Rightarrow u$ sends flow into e
- $\tilde{f}(u,e) < 0 \Rightarrow u$ receives flow from e
- $\text{rcap}(e,u,v) = c(e) - f(e) + \tilde{f}(u,e)^- + \tilde{f}(v,e)^+ = (12-7) + 2 + 4 = 11$

  $\max(0, -\tilde{f}(u,e))$ $\quad$ $\max(0, \tilde{f}(v,e))$
- can implement *any* flow algorithm by treating nets like vertices
- we use Dinic

Institute of Theoretical Informatics
Algorithmics Group

Dinic

- residual BFS to compute distance labels
- residual DFS to find edge-disjoint shortest augmenting paths
- repeat until no flow augmented

Institute of Theoretical Informatics
Algorithmics Group

Dinic

- residual BFS to compute distance labels
- residual DFS to find edge-disjoint shortest augmenting paths
- repeat until no flow augmented

Distance labels for hypergraphs

- $d[u]$ for vertices
- $d_i[e]$ for pushing flow to flow-sending pins $\widetilde{f}(v, e) > 0$
- $d_o[e]$ for pushing flow to all pins. set if $c(e) - f(e) + \widetilde{f}(u, e)^- > 0$

Institute of Theoretical Informatics
Algorithmics Group

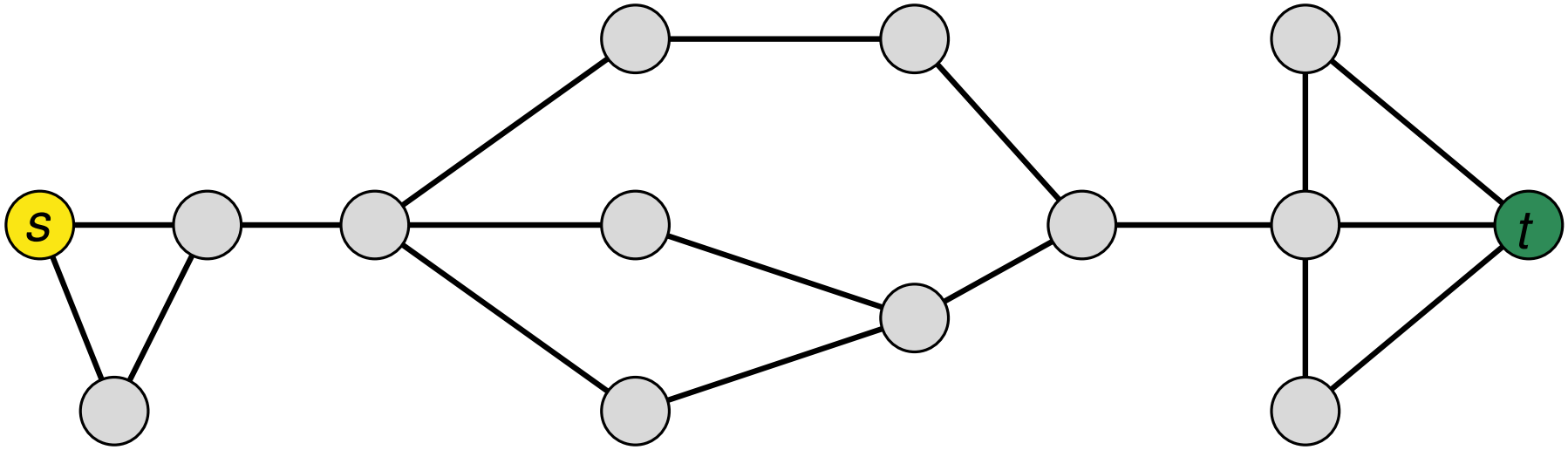
Dinic

- residual BFS to compute distance labels
- residual DFS to find edge-disjoint shortest augmenting paths
- repeat until no flow augmented

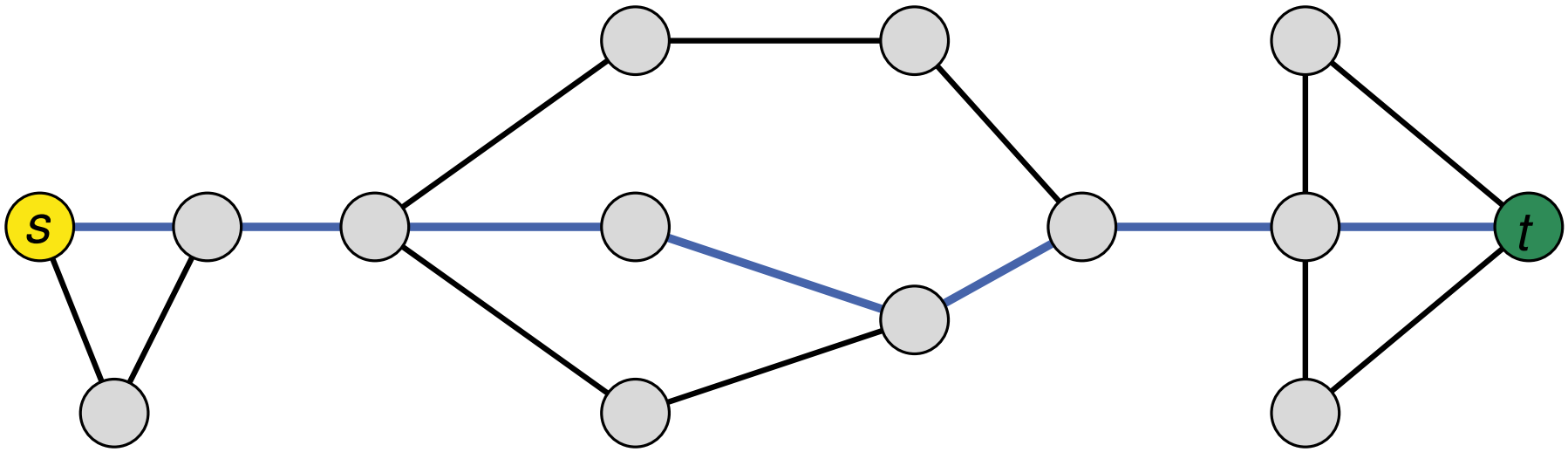
Optimizations

- capacity scaling
  - require $\geq \alpha$ residual capacity
  - if no flow augmented try with $\alpha \leftarrow \alpha/2$
- iterative DFS with stored iterators
- no allocations
- range of active values trick $\rightsquigarrow$ fast resets

FlowCutter

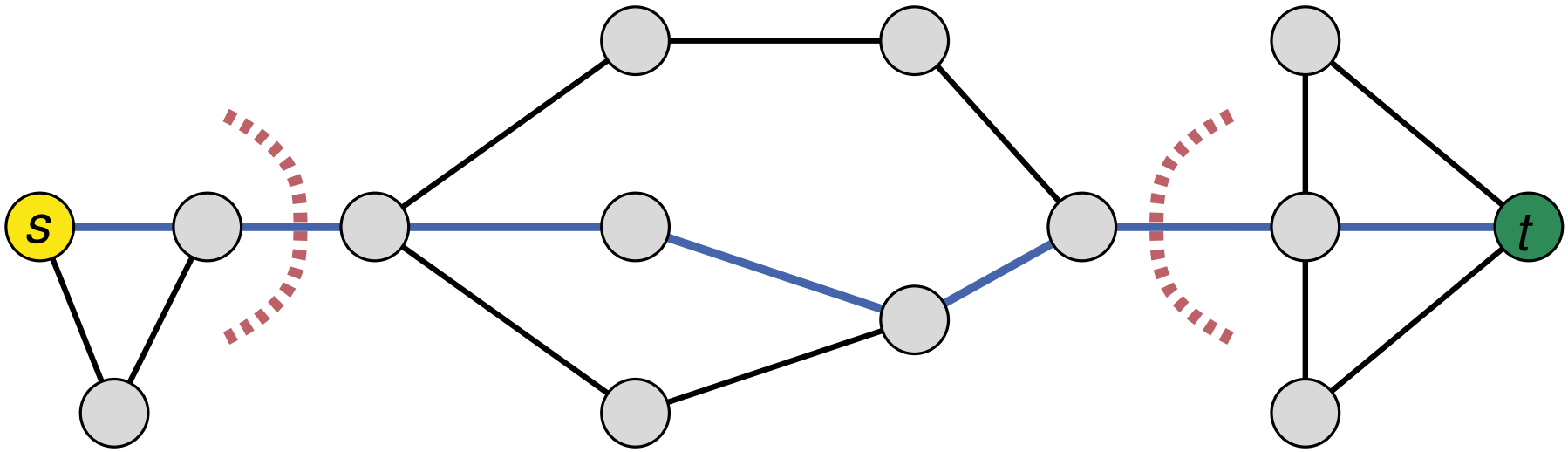


FlowCutter



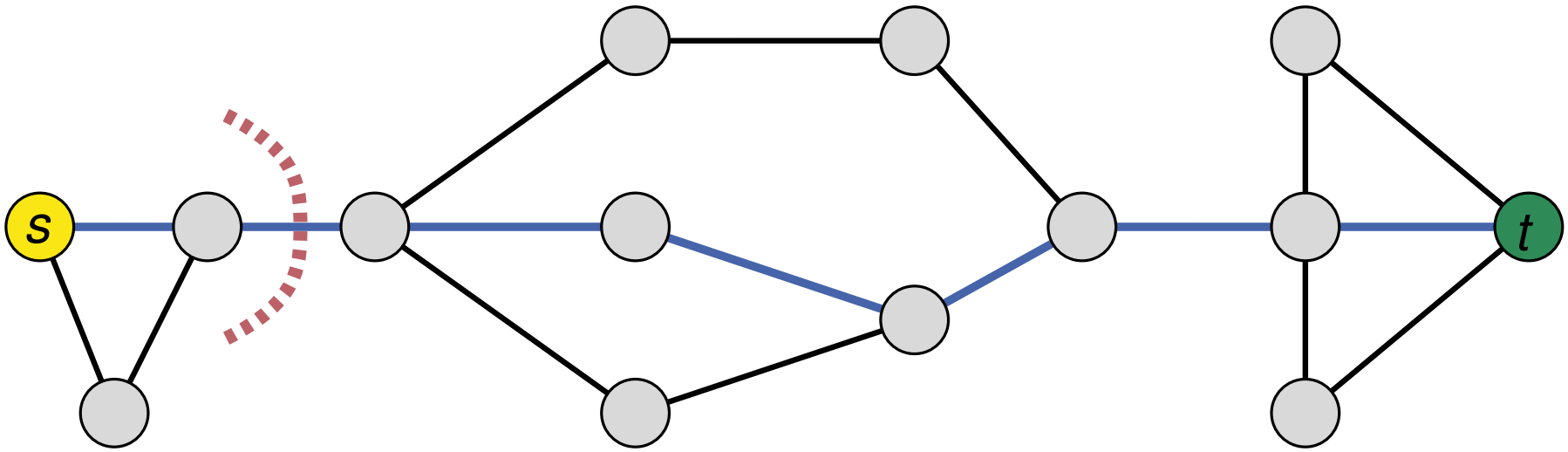
1. Augment flow

FlowCutter



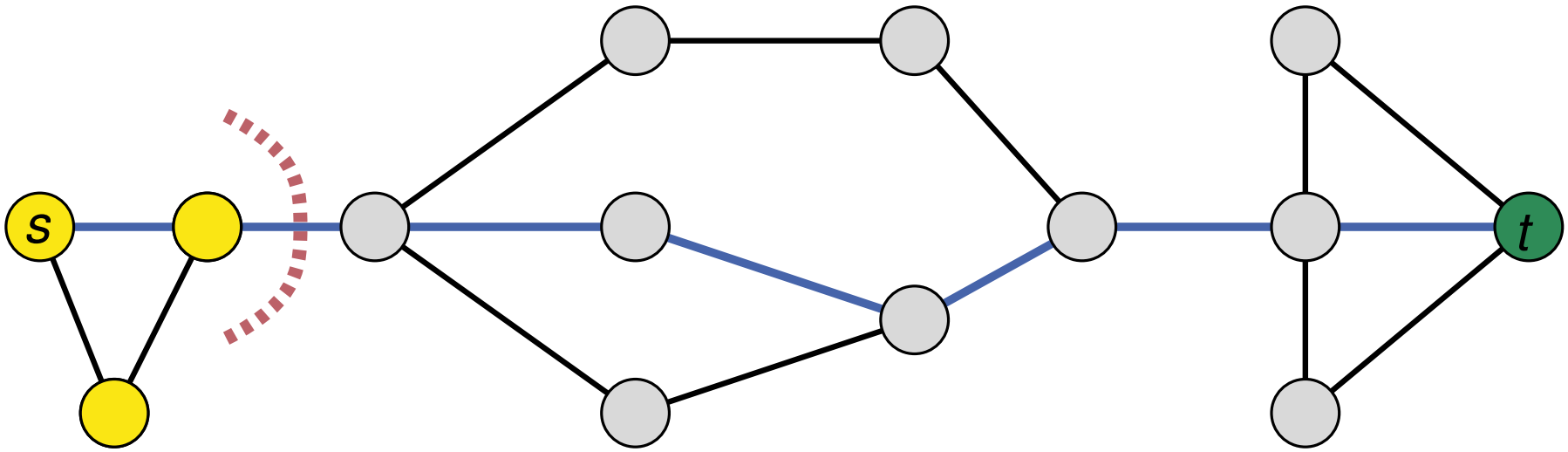
1. Augment flow
2. Find min *s*- and *t*-cut

FlowCutter



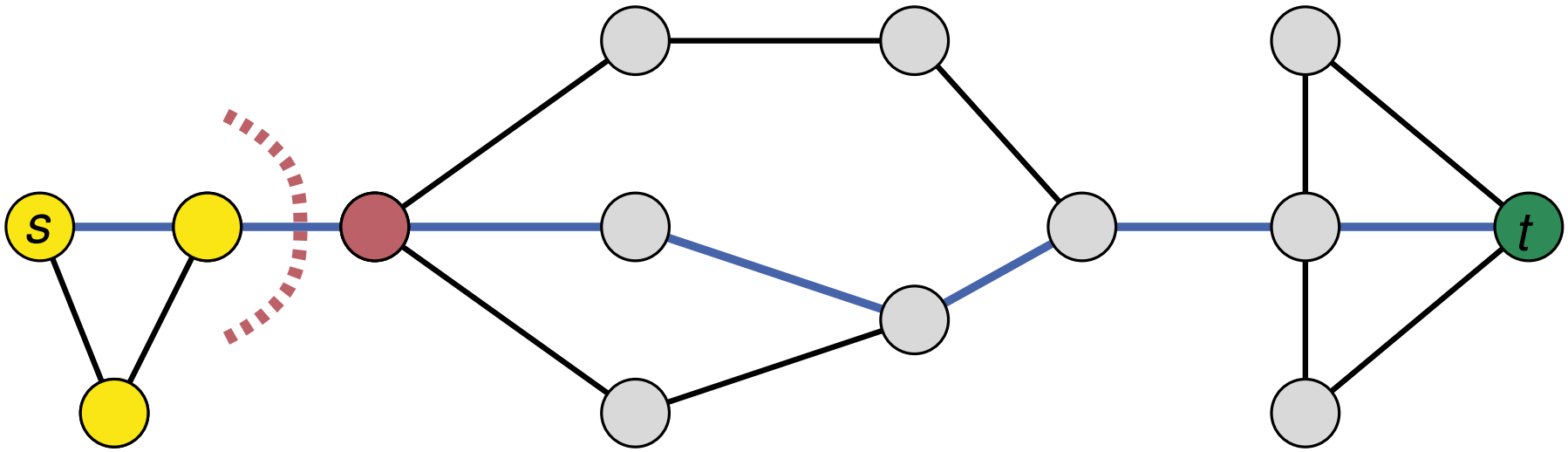
1. Augment flow
2. Find min *s*- and *t*-cut
3. Pick smaller side

FlowCutter



1. Augment flow
2. Find min *s*- and *t*-cut
3. Pick smaller side
4. Assimilate side

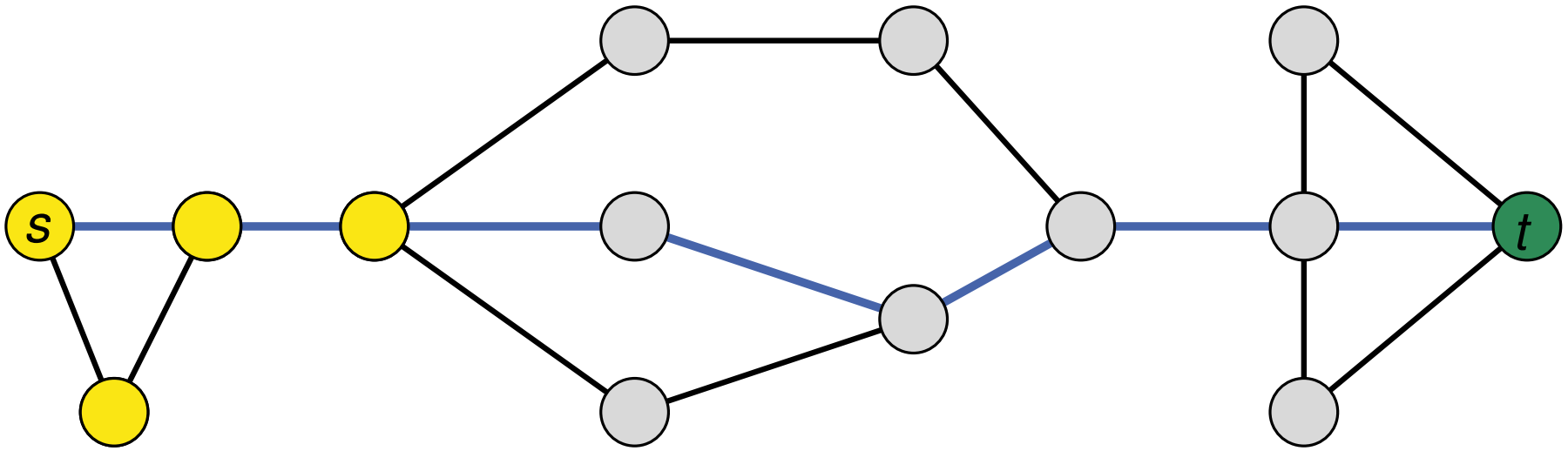
FlowCutter



1. Augment flow
2. Find min *s*- and *t*-cut
3. Pick smaller side
4. Assimilate side
5. Pierce cut

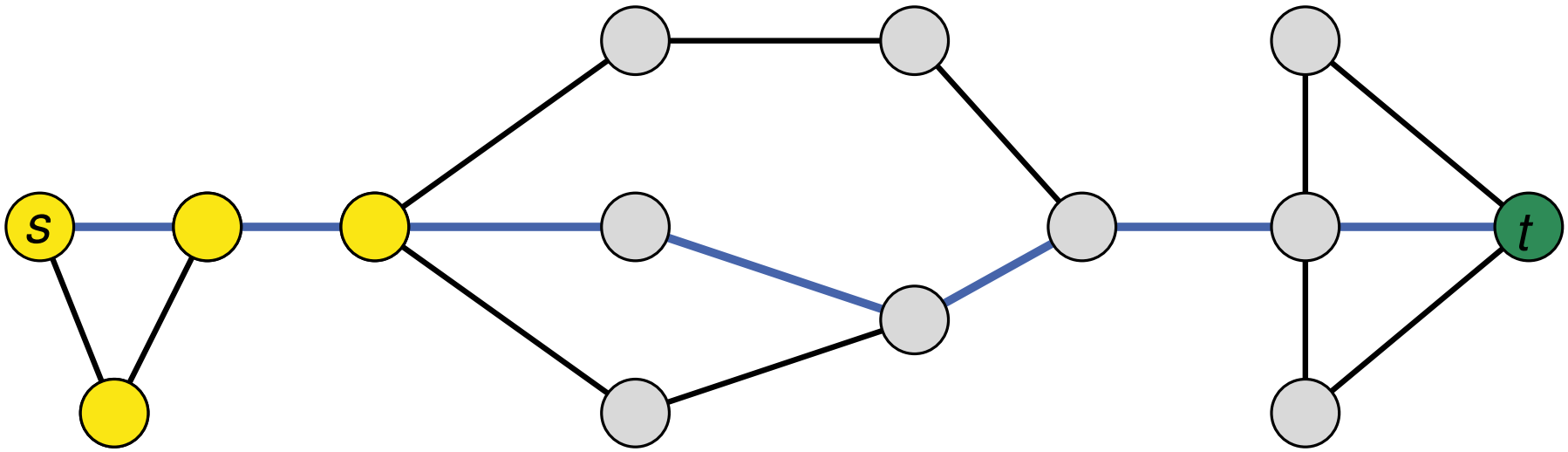
Institute of Theoretical Informatics
Algorithmics Group

FlowCutter



1. Augment flow
2. Find min *s*- and *t*-cut
3. Pick smaller side
4. Assimilate side
5. Pierce cut

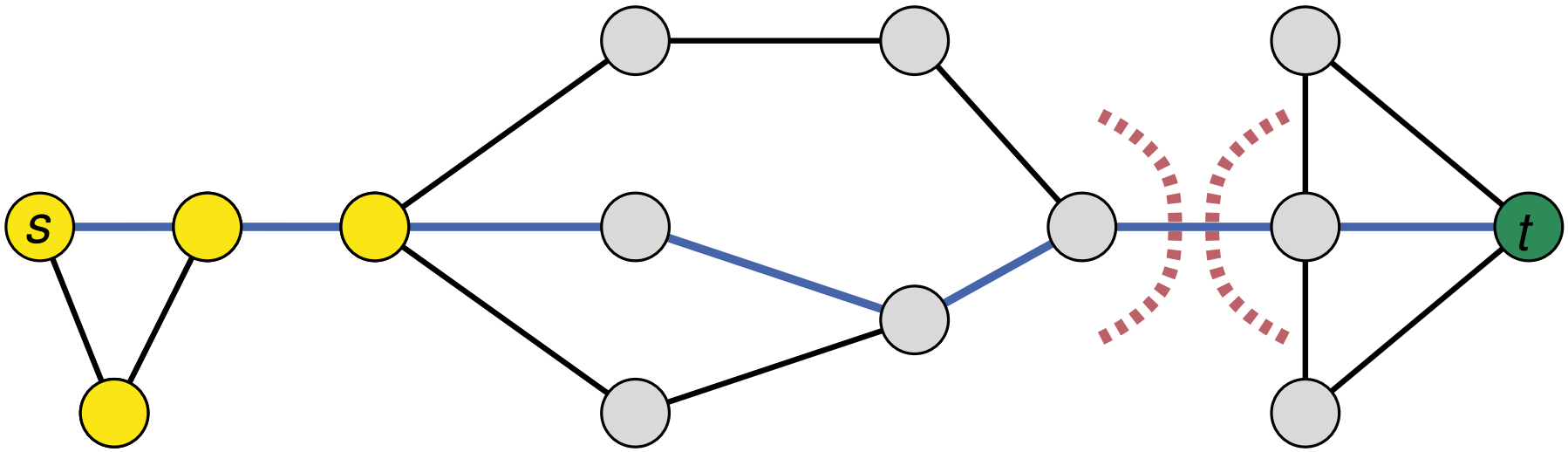
FlowCutter



1. Augment flow
2. Find min *s*- and *t*-cut
3. Pick smaller side
4. Assimilate side
5. Pierce cut

Institute of Theoretical Informatics
Algorithmics Group

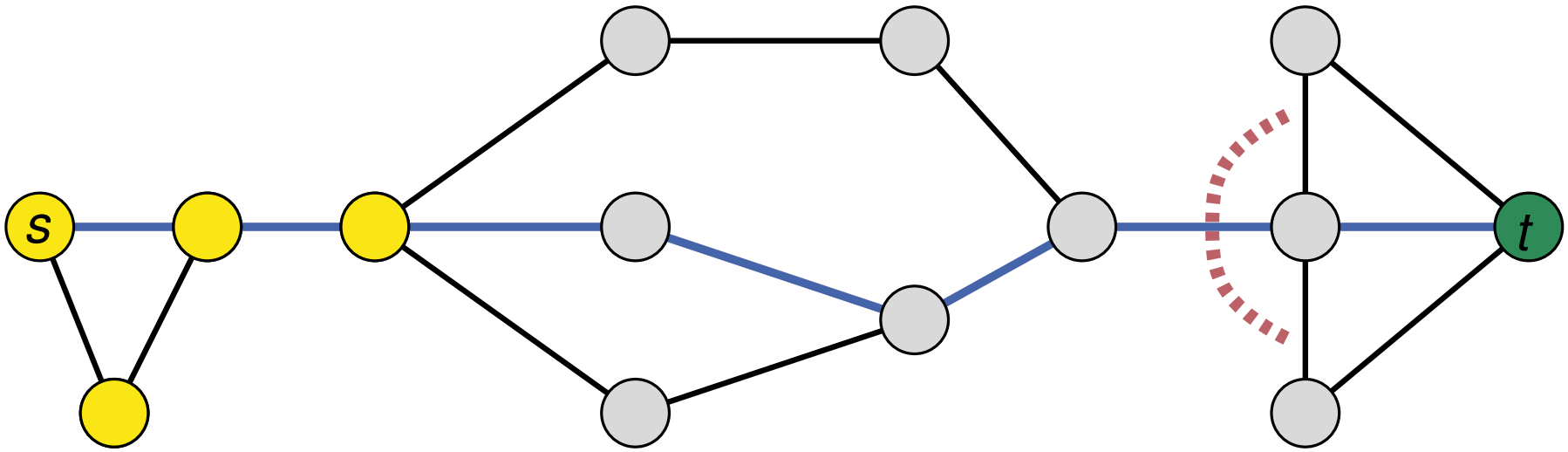
FlowCutter



1. Augment flow
2. Find min *s*- and *t*-cut
3. Pick smaller side
4. Assimilate side
5. Pierce cut

Institute of Theoretical Informatics
Algorithmics Group

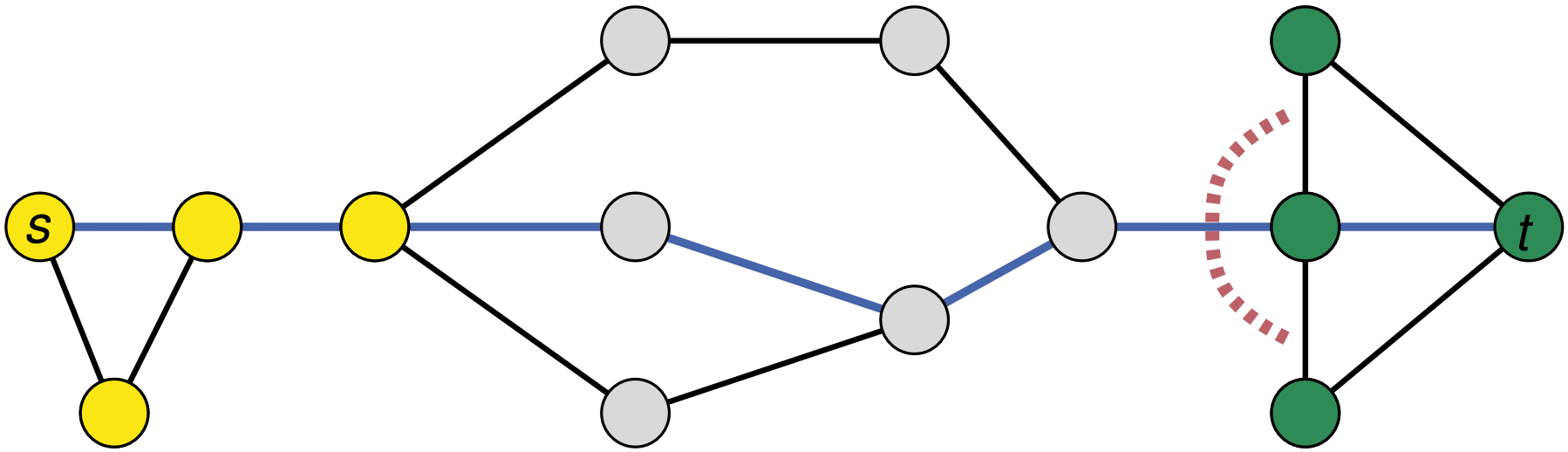
FlowCutter



1. Augment flow
2. Find min *s*- and *t*-cut
3. Pick smaller side
4. Assimilate side
5. Pierce cut

Institute of Theoretical Informatics
Algorithmics Group

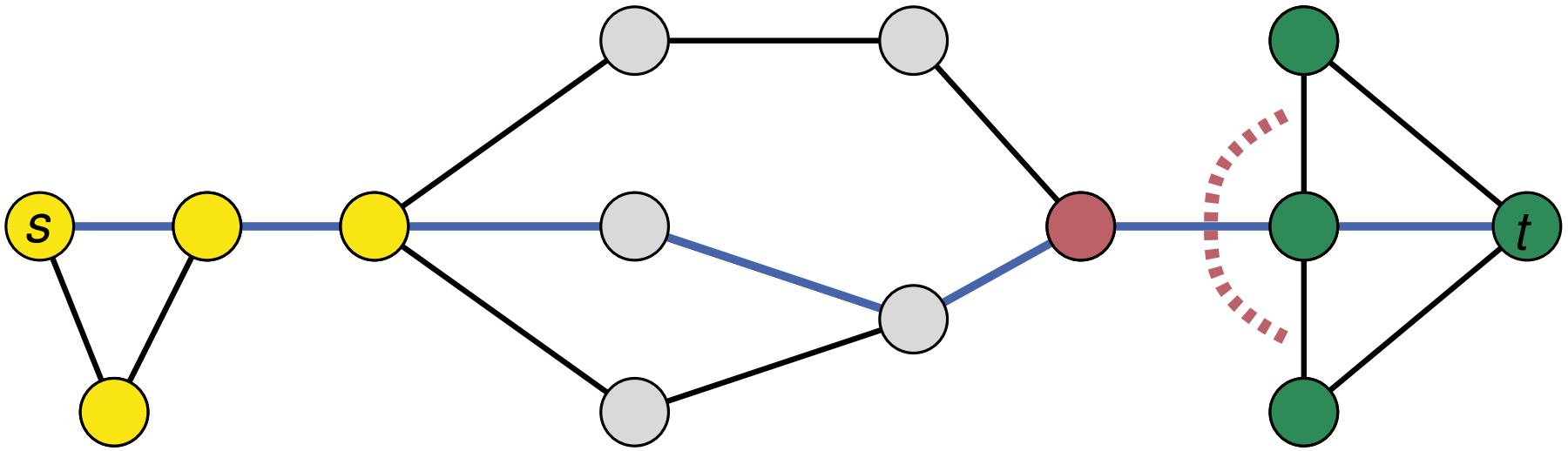
FlowCutter



1. Augment flow
2. Find min *s*- and *t*-cut
3. Pick smaller side
4. Assimilate side
5. Pierce cut

Institute of Theoretical Informatics
Algorithmics Group

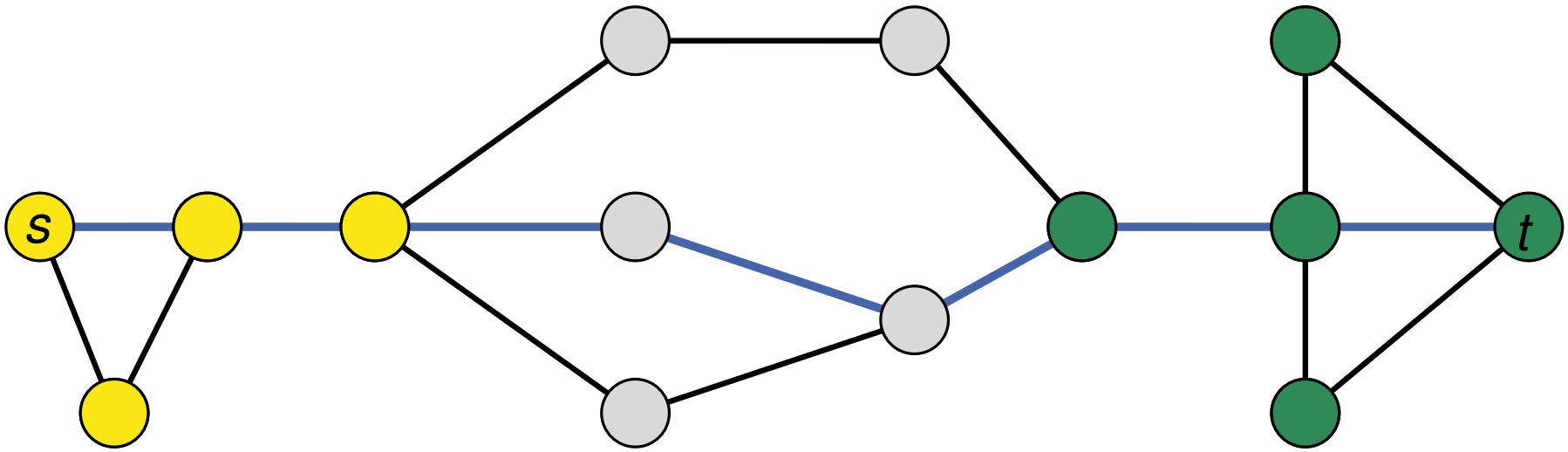
FlowCutter



1. Augment flow
2. Find min *s*- and *t*-cut
3. Pick smaller side
4. Assimilate side
5. Pierce cut

Institute of Theoretical Informatics
Algorithmics Group

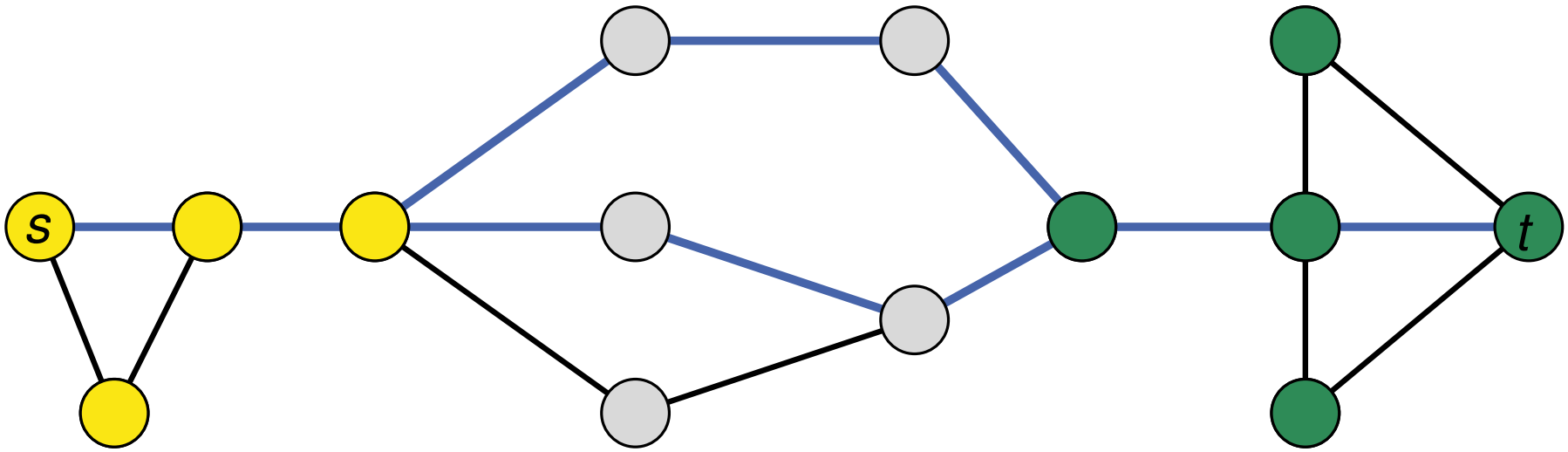
FlowCutter



1. Augment flow
2. Find min *s*- and *t*-cut
3. Pick smaller side
4. Assimilate side
5. Pierce cut

Institute of Theoretical Informatics
Algorithmics Group

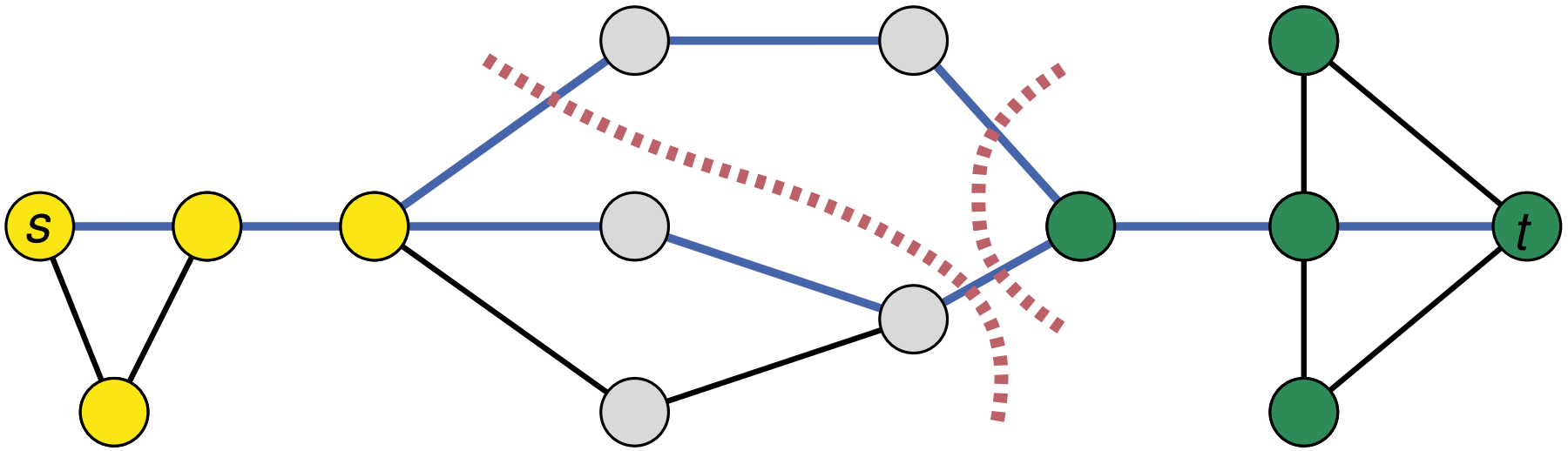
FlowCutter



1. Augment flow
2. Find min *s*- and *t*-cut
3. Pick smaller side
4. Assimilate side
5. Pierce cut

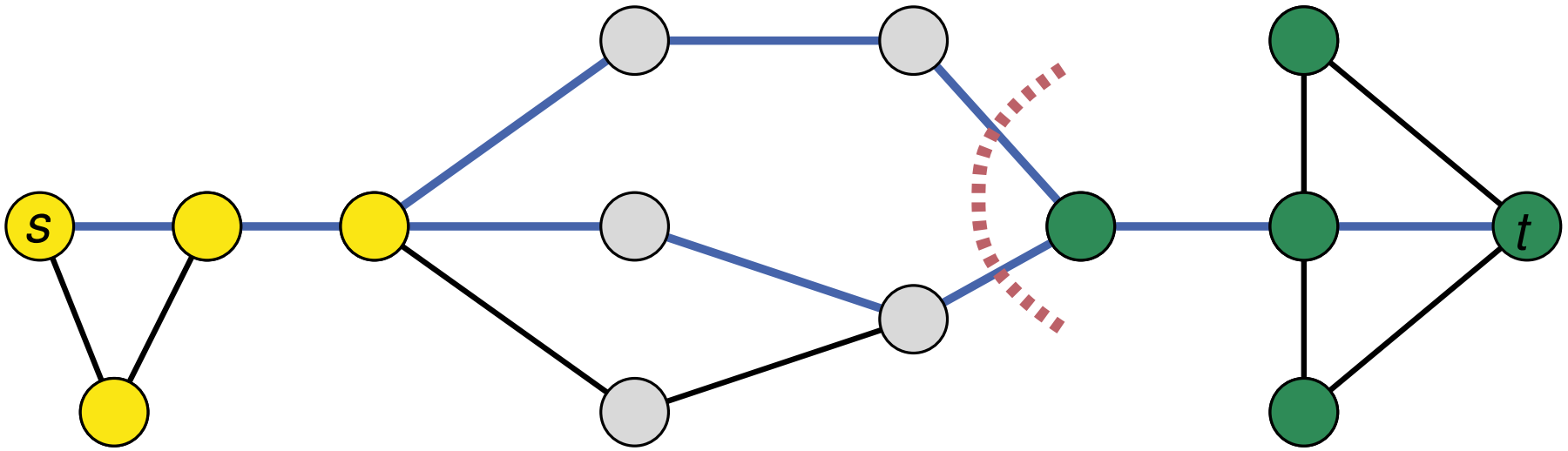
Institute of Theoretical Informatics
Algorithmics Group

FlowCutter



1. Augment flow
2. Find min *s*- and *t*-cut
3. Pick smaller side
4. Assimilate side
5. Pierce cut

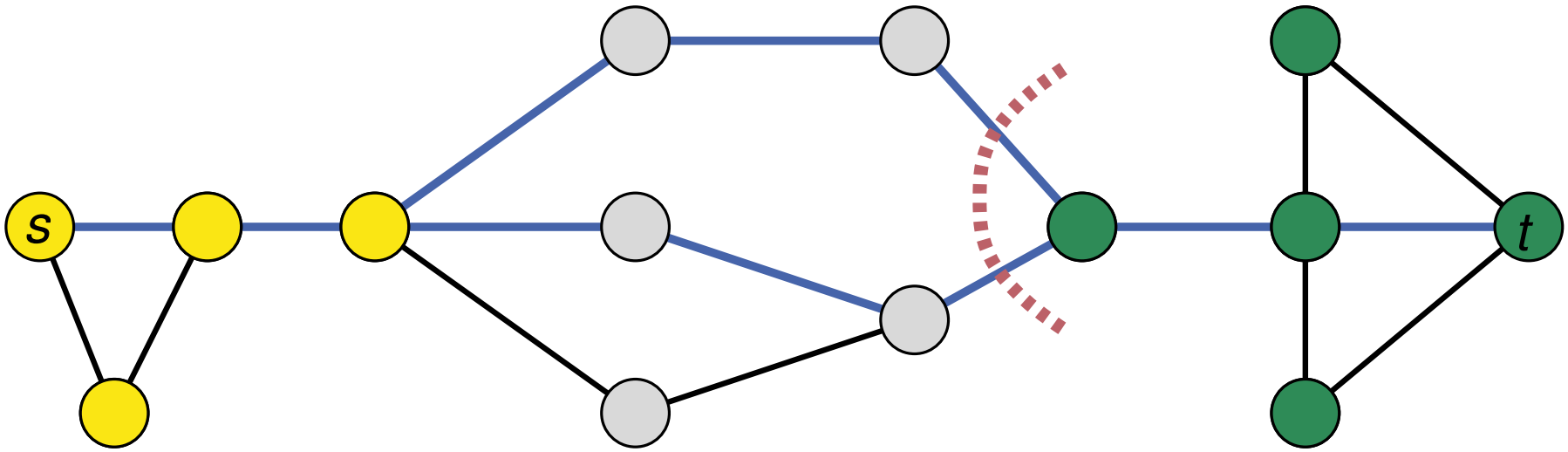
FlowCutter



1. Augment flow
2. Find min *s*- and *t*-cut
3. Pick smaller side
4. Assimilate side
5. Pierce cut

Institute of Theoretical Informatics
Algorithmics Group

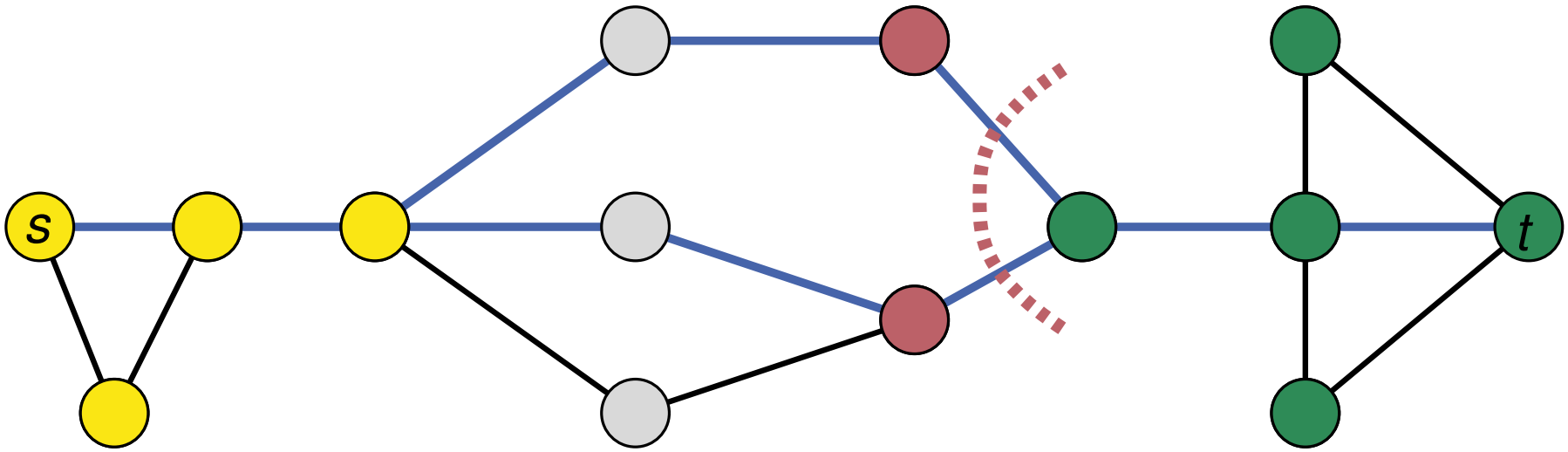
FlowCutter



1. Augment flow
2. Find min *s*- and *t*-cut
3. Pick smaller side
4. Assimilate side
5. Pierce cut

Institute of Theoretical Informatics
Algorithmics Group

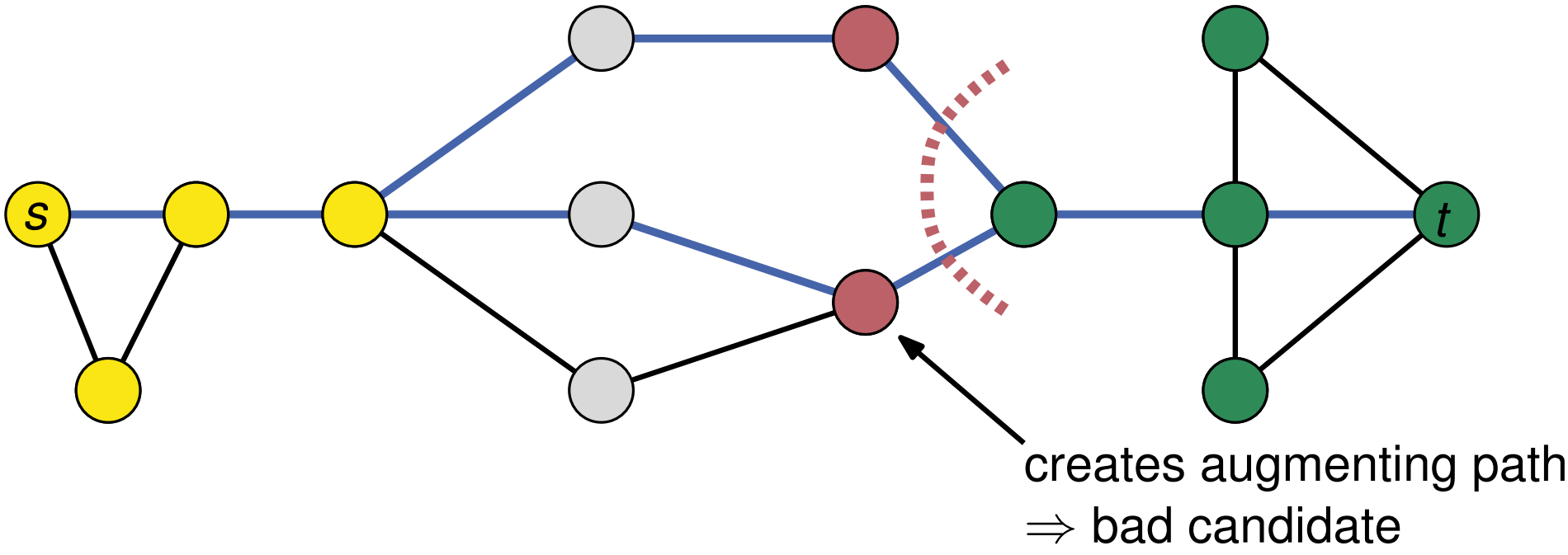
FlowCutter



1. Augment flow
2. Find min *s*- and *t*-cut
3. Pick smaller side
4. Assimilate side
5. Pierce cut

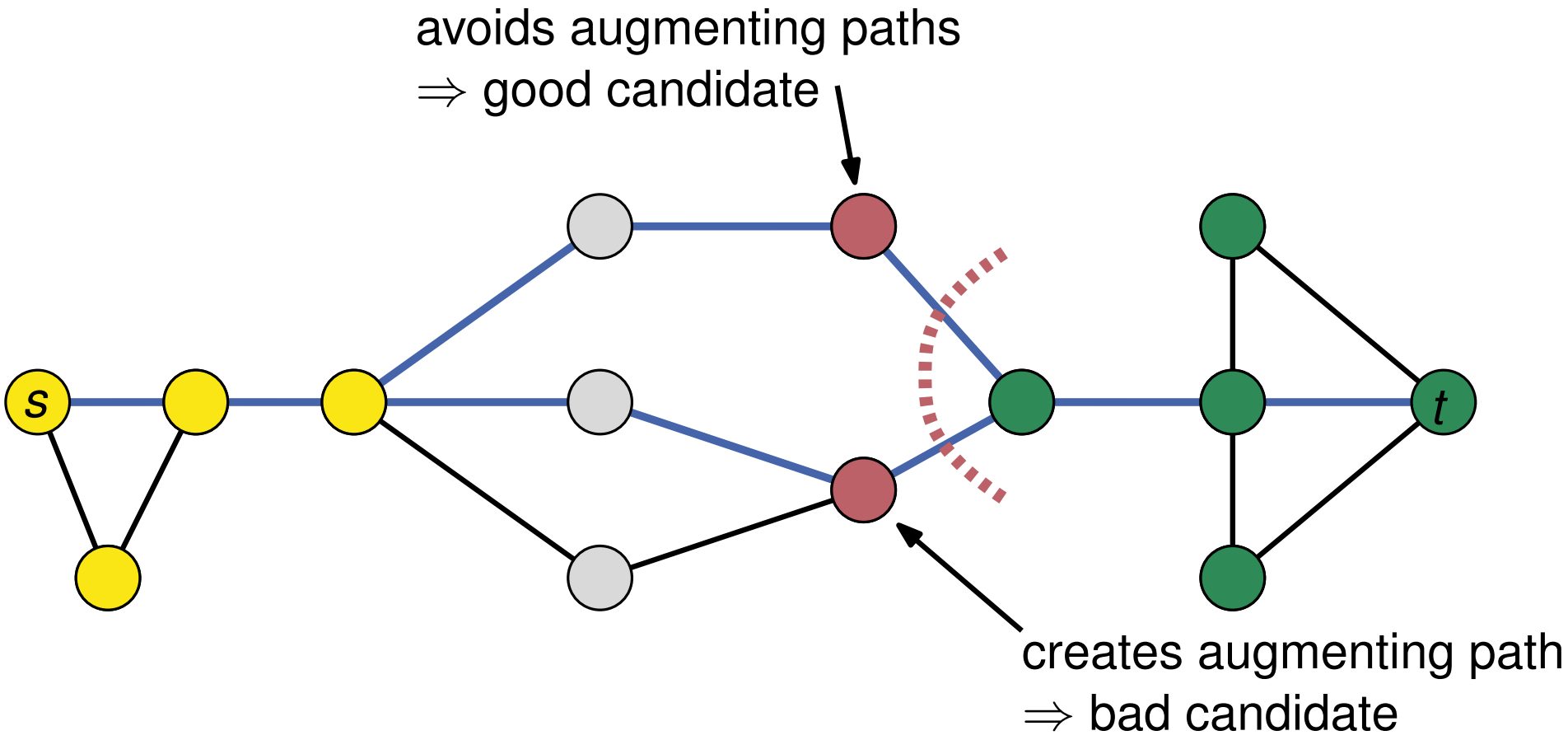
Institute of Theoretical Informatics
Algorithmics Group

FlowCutter



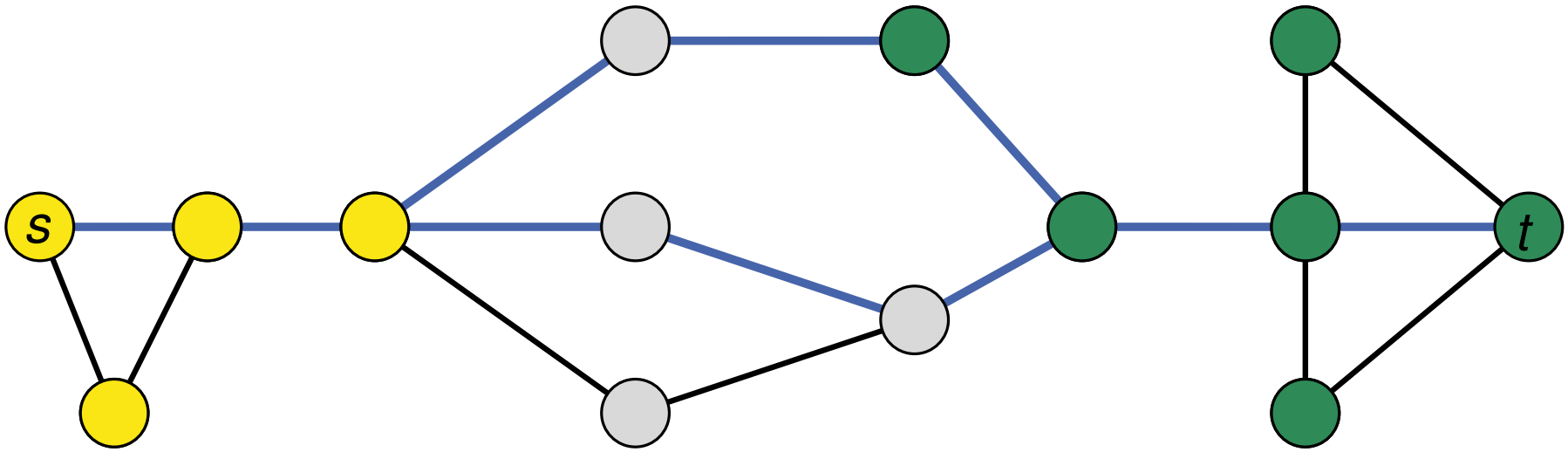
1. Augment flow
2. Find min *s*- and *t*-cut
3. Pick smaller side
4. Assimilate side
5. Pierce cut

FlowCutter



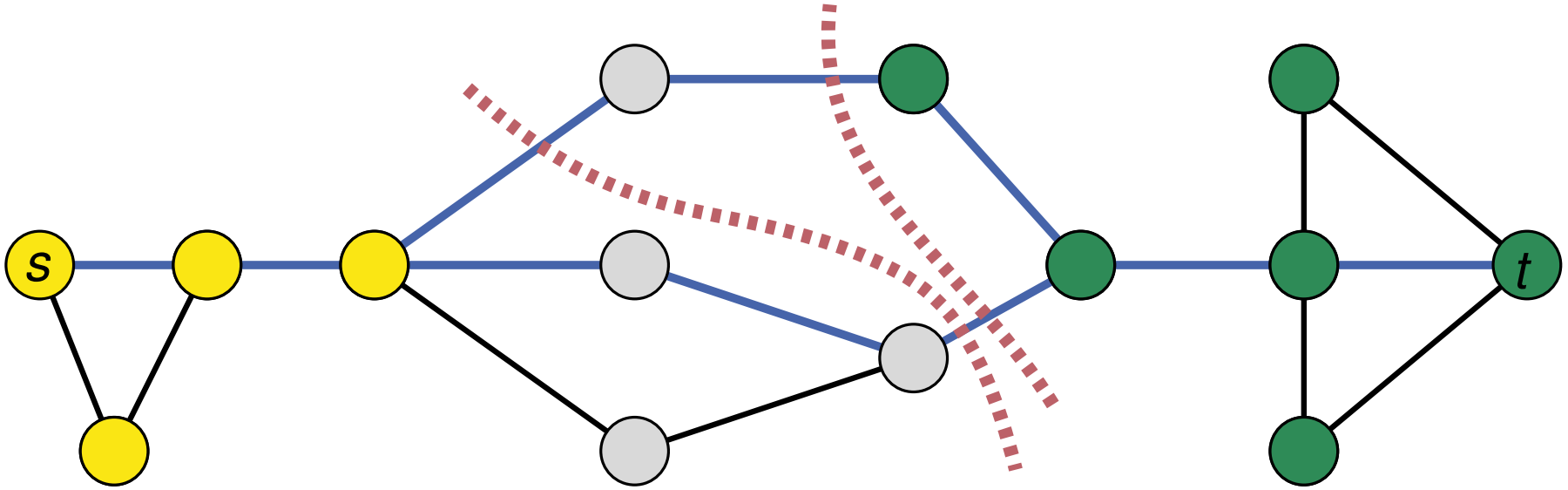
1. Augment flow
2. Find min *s*- and *t*-cut
3. Pick smaller side
4. Assimilate side
5. Pierce cut

FlowCutter



1. Augment flow
2. Find min *s*- and *t*-cut
3. Pick smaller side
4. Assimilate side
5. Pierce cut

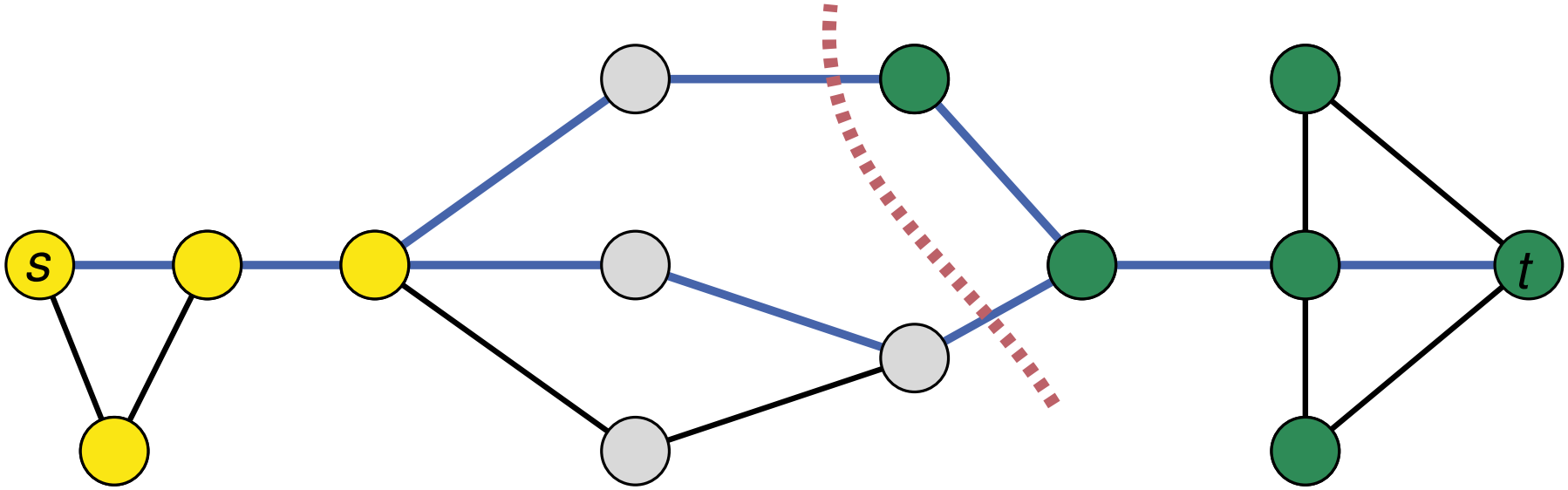
FlowCutter



1. Augment flow
2. Find min *s*- and *t*-cut
3. Pick smaller side
4. Assimilate side
5. Pierce cut

Institute of Theoretical Informatics
Algorithmics Group

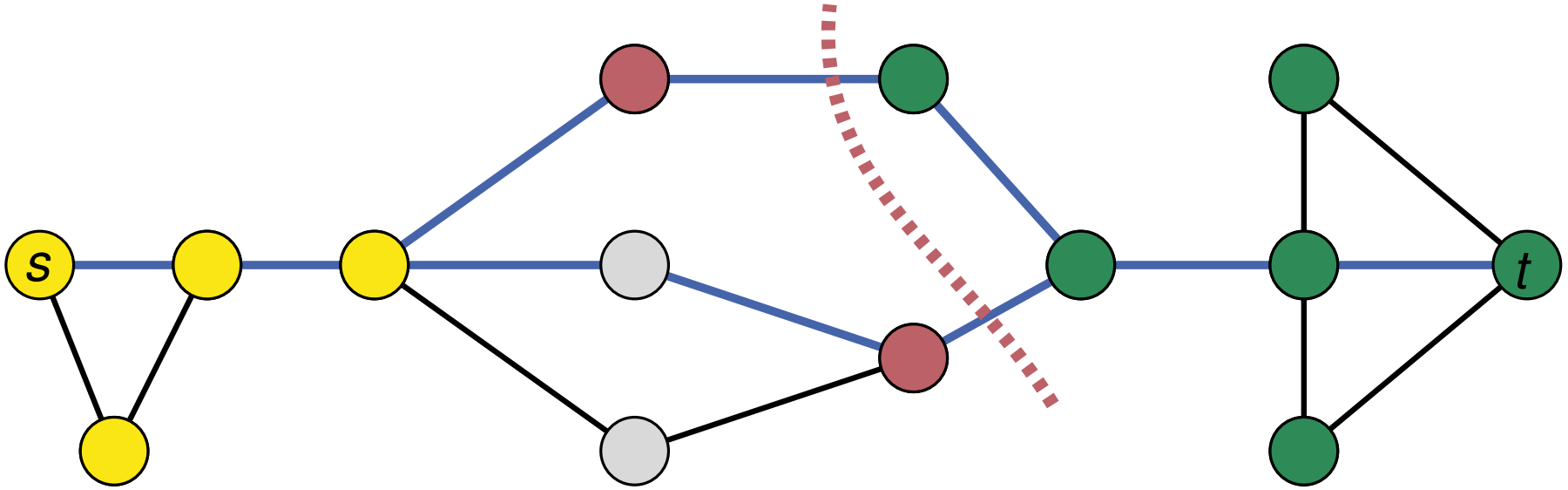
FlowCutter



1. Augment flow
2. Find min *s*- and *t*-cut
3. Pick smaller side
4. Assimilate side
5. Pierce cut

Institute of Theoretical Informatics
Algorithmics Group

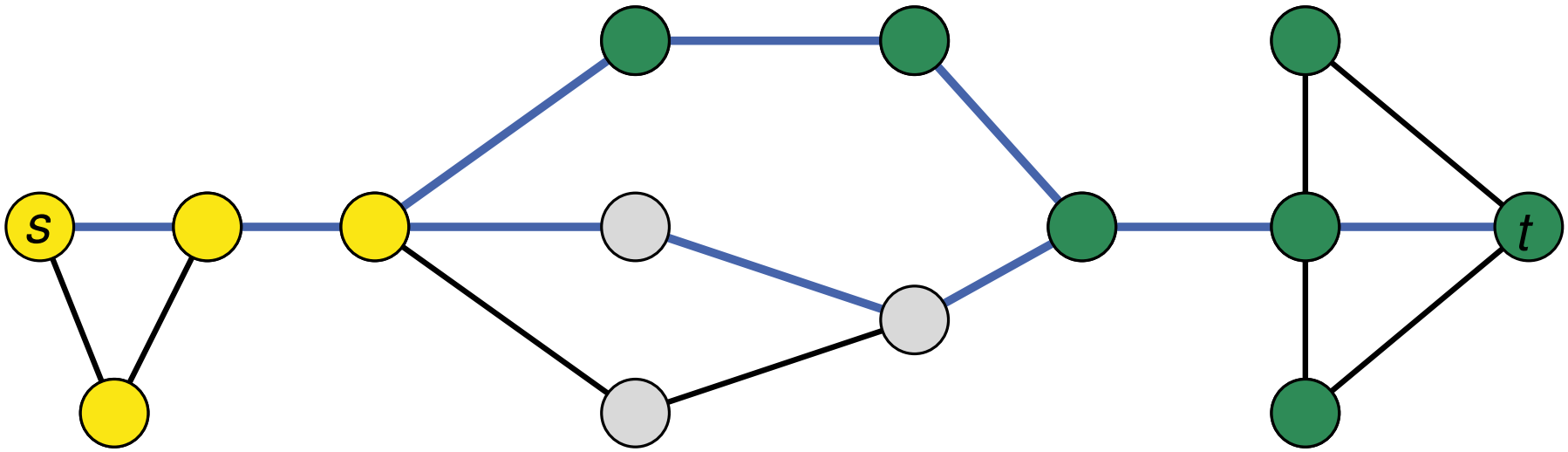
FlowCutter



1. Augment flow
2. Find min *s*- and *t*-cut
3. Pick smaller side
4. Assimilate side
5. Pierce cut

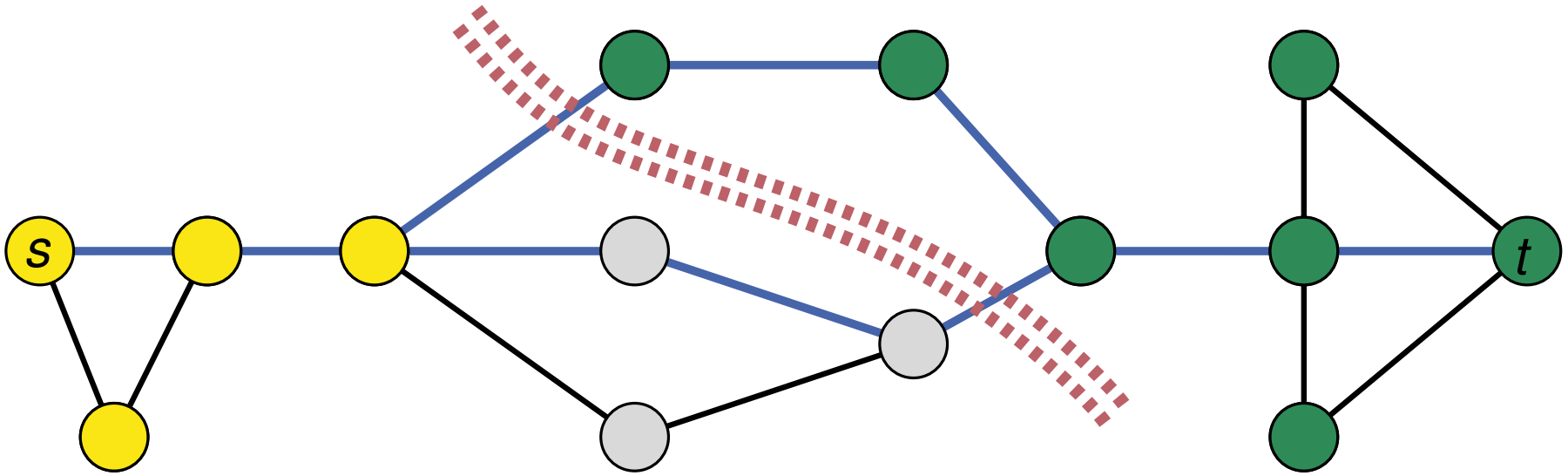
Institute of Theoretical Informatics
Algorithmics Group

FlowCutter



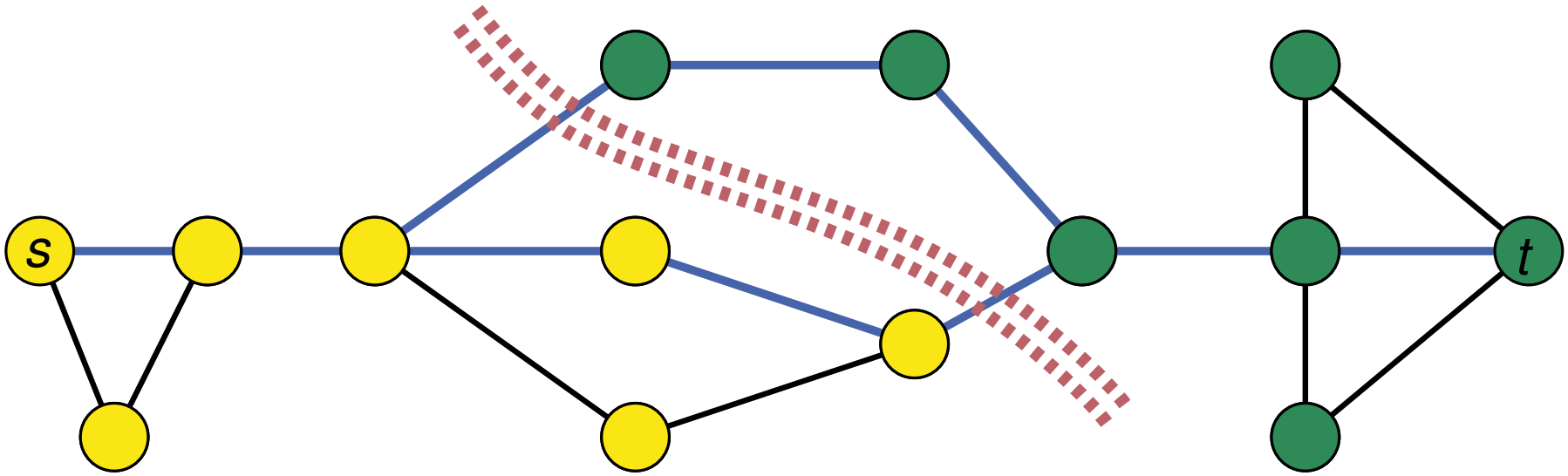
1. Augment flow
2. Find min *s*- and *t*-cut
3. Pick smaller side
4. Assimilate side
5. Pierce cut

FlowCutter



1. Augment flow
2. Find min *s*- and *t*-cut
3. Pick smaller side
4. Assimilate side
5. Pierce cut

FlowCutter



1. Augment flow
2. Find min *s*- and *t*-cut
3. Pick smaller side
4. Assimilate side
5. Pierce cut

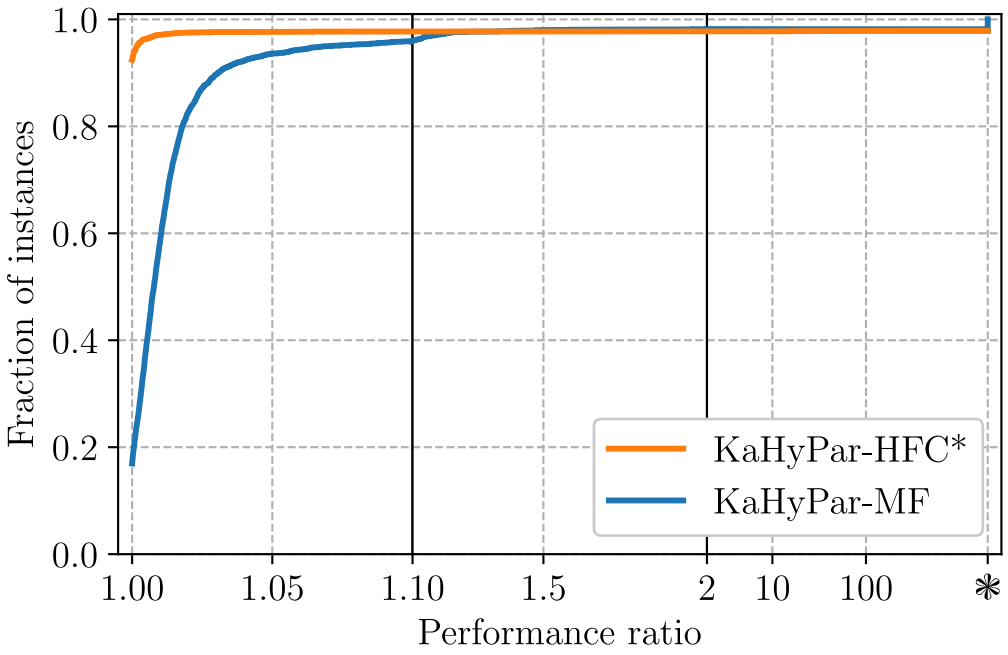
Experimental Setup

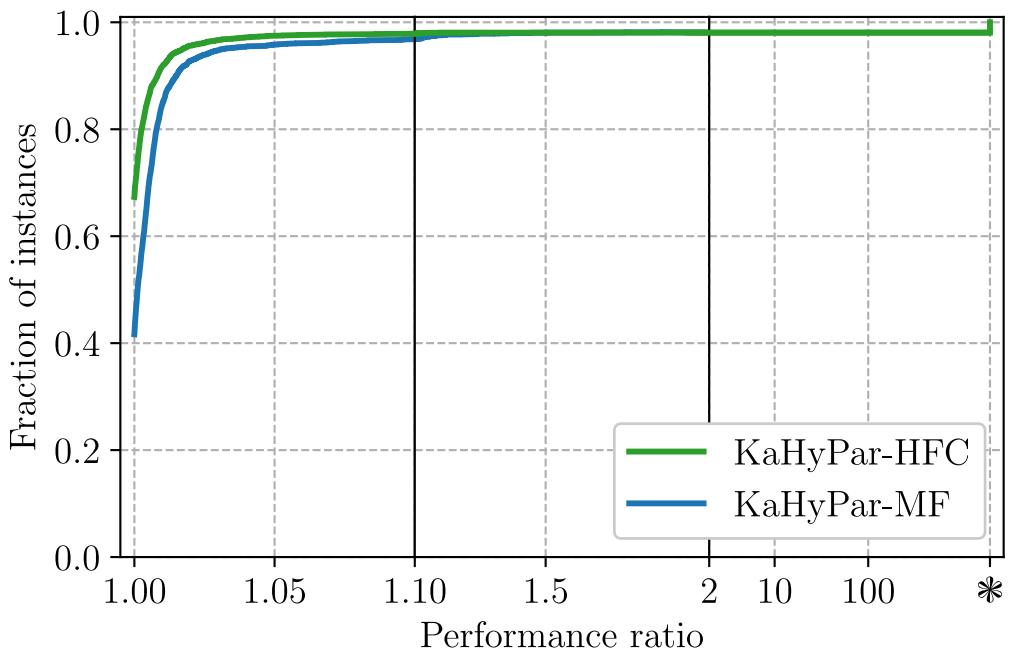
- Intel Xeon E5-2670 @ 2.6 Ghz, 64 GB RAM, 20 MB L3, 256KB L2
- # Hypergraphs: [publicly available]
  - SuiteSparse Matrix Collection 184
  - SAT Competition 2014 (3 representations) 92·3
  - ISPD98 & DAC2012 VLSI Circuits 28
- $k \in \{2, 4, 8, 16, 32, 64, 128\}$ with imbalance: $\varepsilon = 3\%$
- 10 random seeds

Experimental Setup

- Intel Xeon E5-2670 @ 2.6 Ghz, 64 GB RAM, 20 MB L3, 256KB L2
- # Hypergraphs: [publicly available]
  - SuiteSparse Matrix Collection 184
  - SAT Competition 2014 (3 representations) 92·3
  - ISPD98 & DAC2012 VLSI Circuits 28
- $k \in \{2, 4, 8, 16, 32, 64, 128\}$ with imbalance: $\varepsilon = 3\%$
- 10 random seeds
- Comparing **KaHyPar-HFC*** and **KaHyPar-HFC** with:
  - KaHyPar-MF
  - hMetis-R(ecursive Bisection) & hMetis-K (direct -kway)
  - PaToH-D(efault) & PaToH-Q(uality)
  - Mondriaan
  - Zoltan-AlgD
  - HYPE

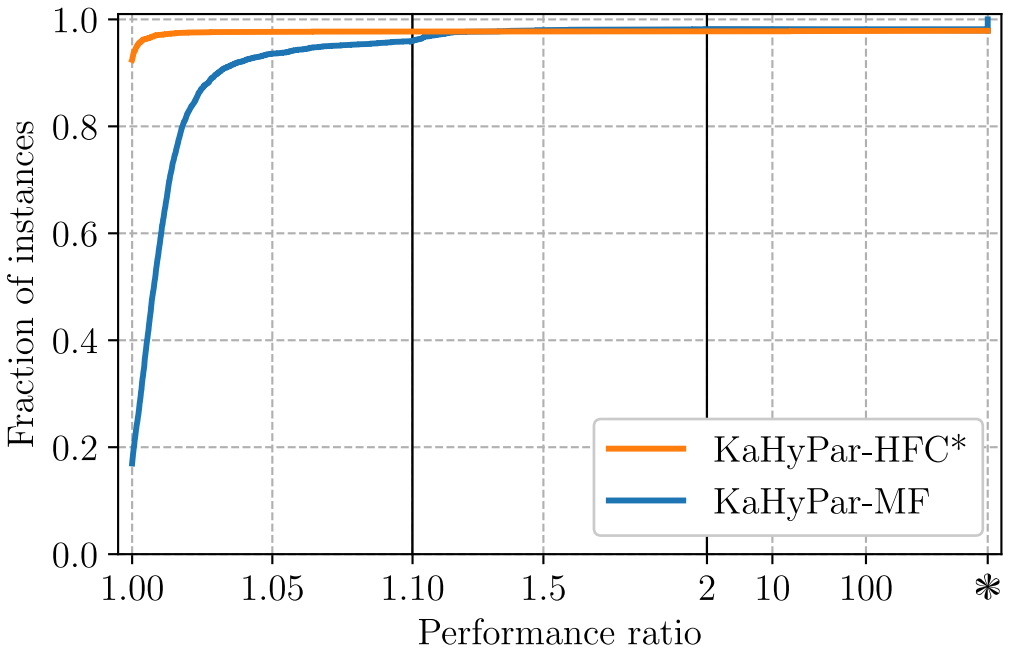
Comparison with previous KaHyPar

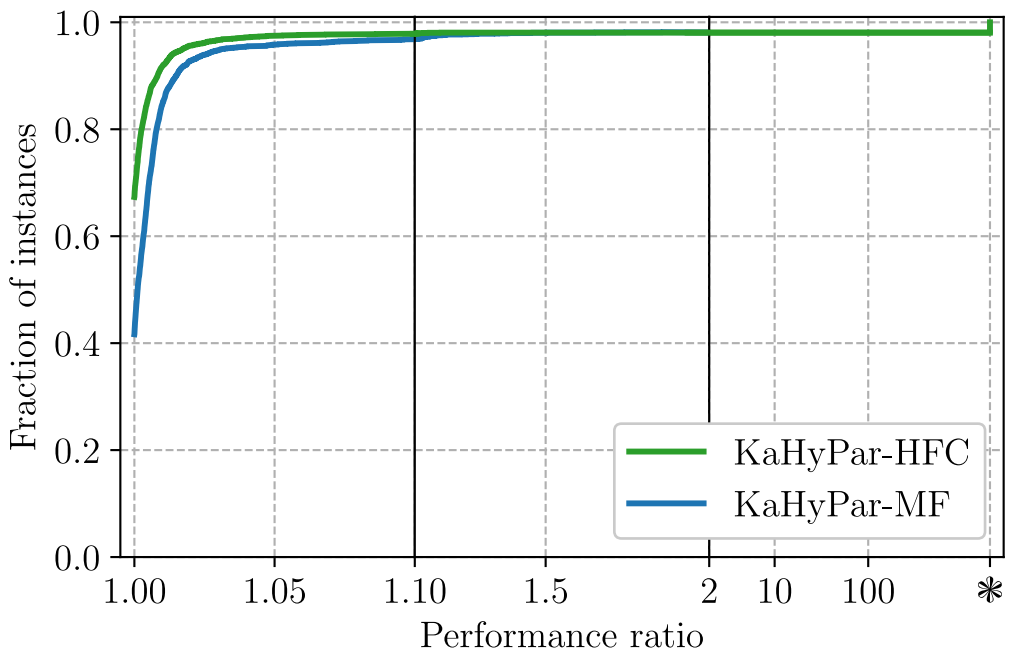




Institute of Theoretical Informatics
Algorithmics Group

Comparison with previous KaHyPar

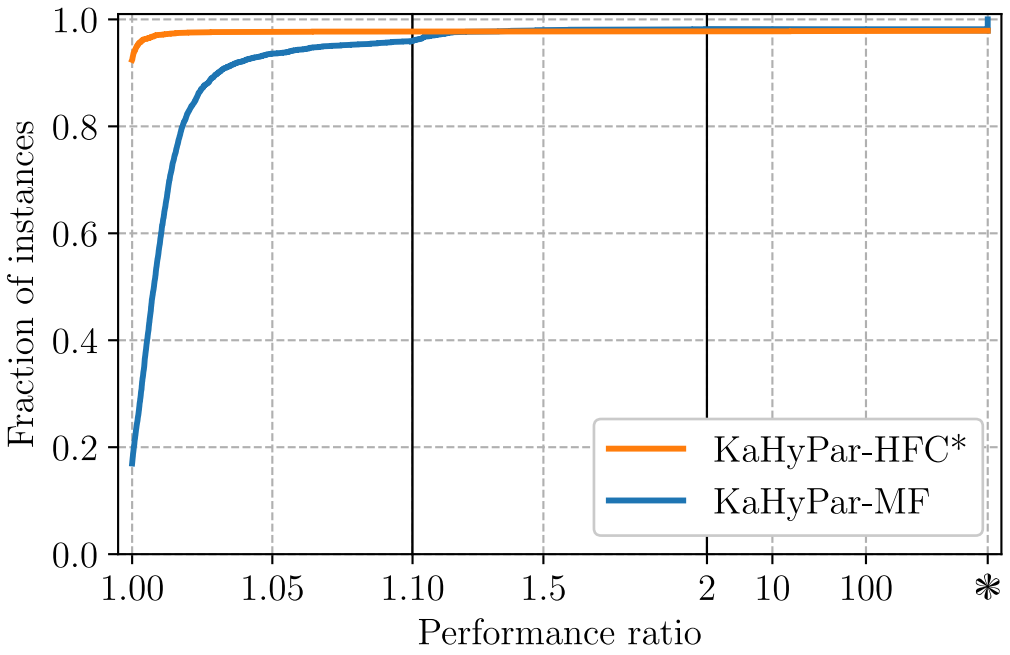


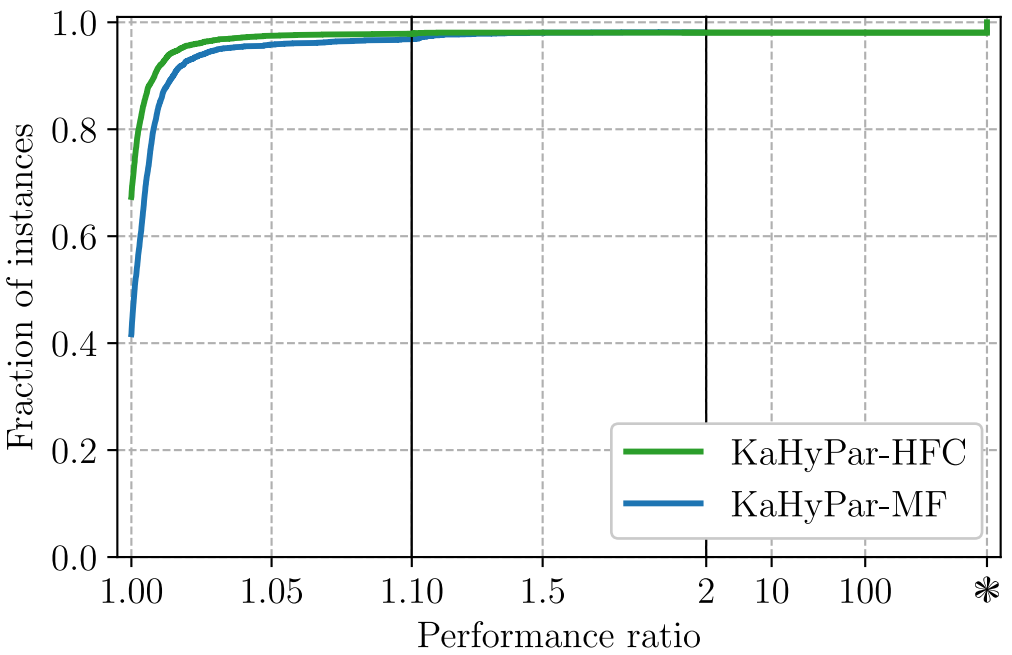


- performance ratio $r(a, i) = \frac{con(a,i)}{\min\{con(a',i) \mid a' \in \mathcal{A}\}}$ of algorithm a on instance i

Institute of Theoretical Informatics
Algorithmics Group

Comparison with previous KaHyPar

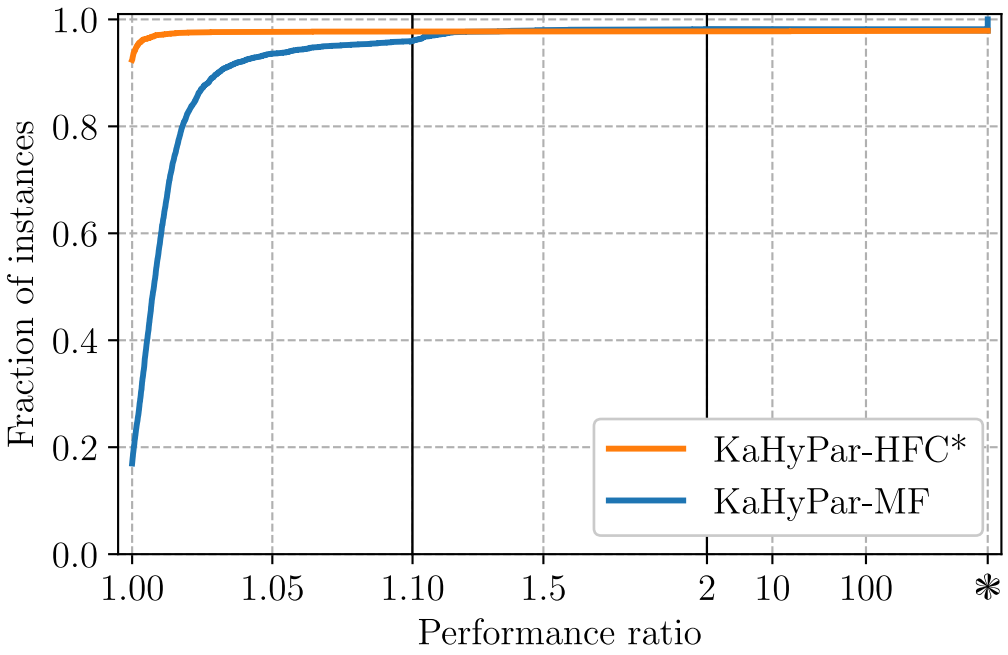


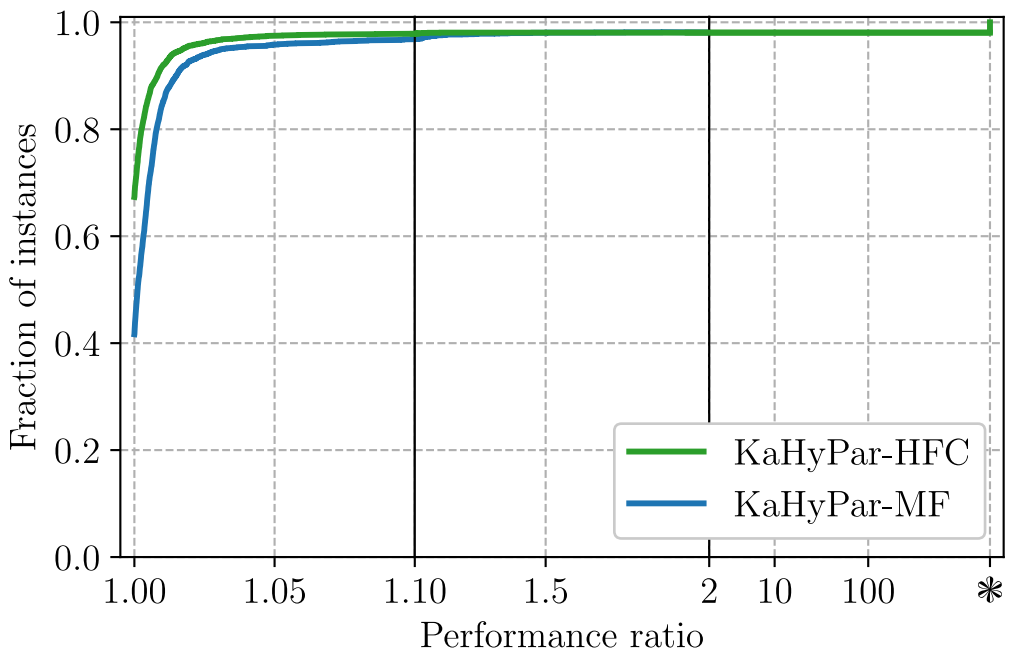


- performance ratio $r(a, i) = \frac{con(a,i)}{\min\{con(a',i) \mid a' \in \mathcal{A}\}}$ of algorithm a on instance i
- y-axis = fraction of instances with smaller performance ratio than value on x-axis

Institute of Theoretical Informatics
Algorithmics Group

Comparison with previous KaHyPar

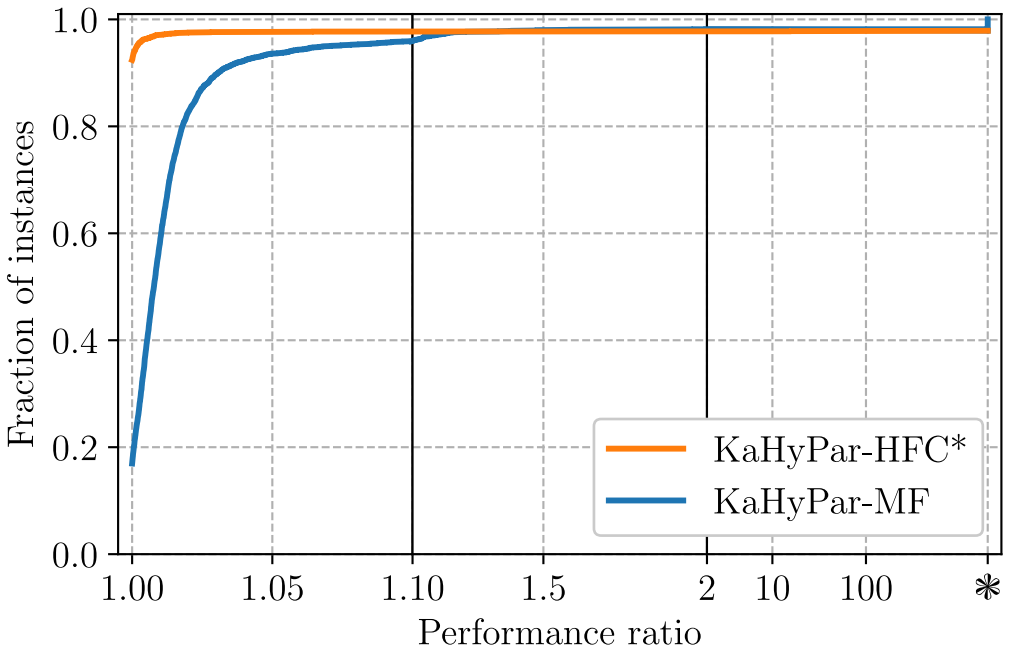


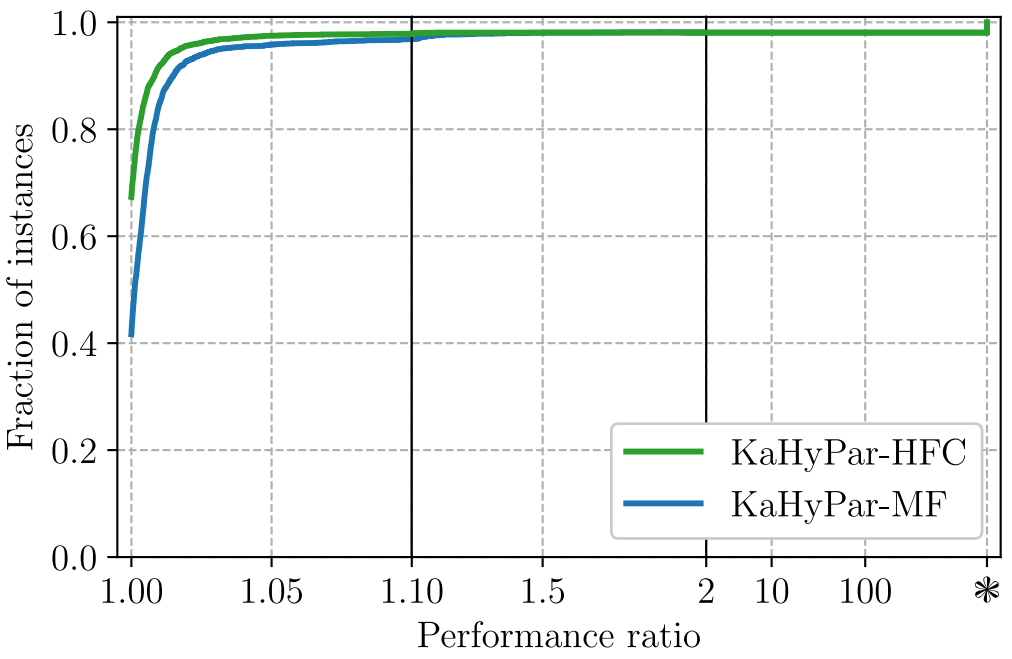


- performance ratio $r(a, i) = \frac{con(a,i)}{\min\{con(a',i) \mid a' \in \mathcal{A}\}}$ of algorithm a on instance i
- y-axis = fraction of instances with smaller performance ratio than value on x-axis
- x = 1 $\Rightarrow$ fraction of instances on which algorithm is the best
- higher is better

Institute of Theoretical Informatics
Algorithmics Group

Comparison with previous KaHyPar

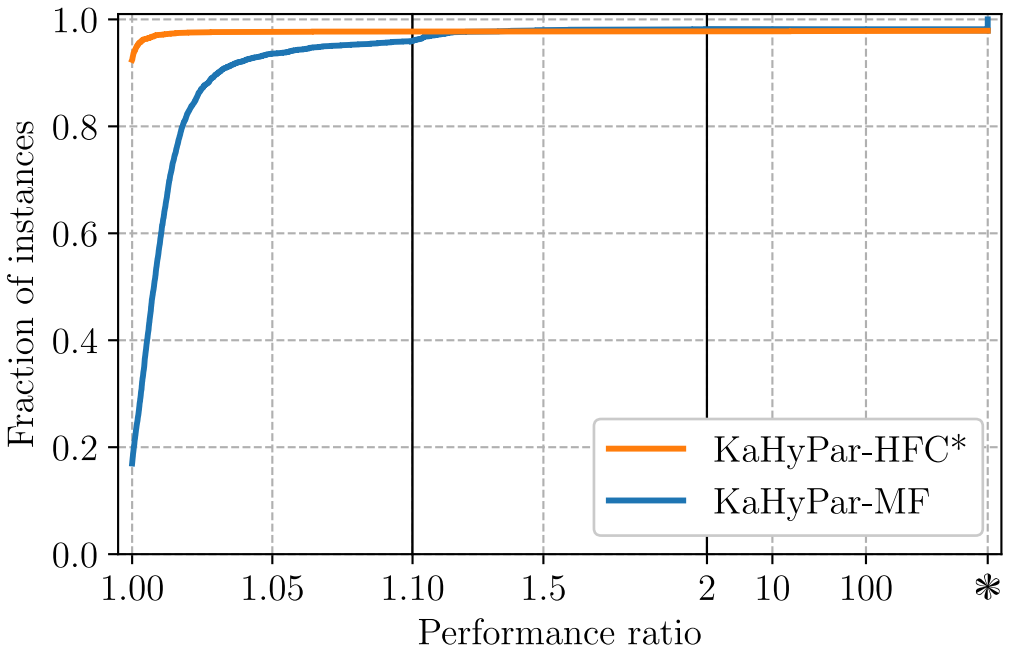


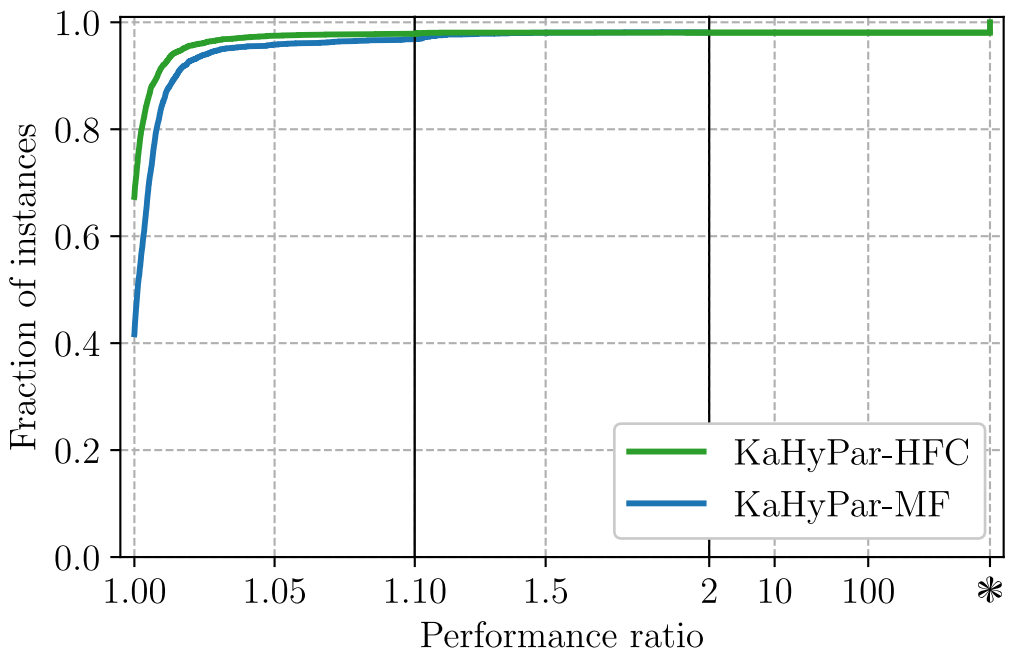


Algorithm	$t[s]$
KaHyPar-MF	67.07
KaHyPar-HFC*	62.49
KaHyPar-HFC	44.84

Institute of Theoretical Informatics
Algorithmics Group

Comparison with previous KaHyPar





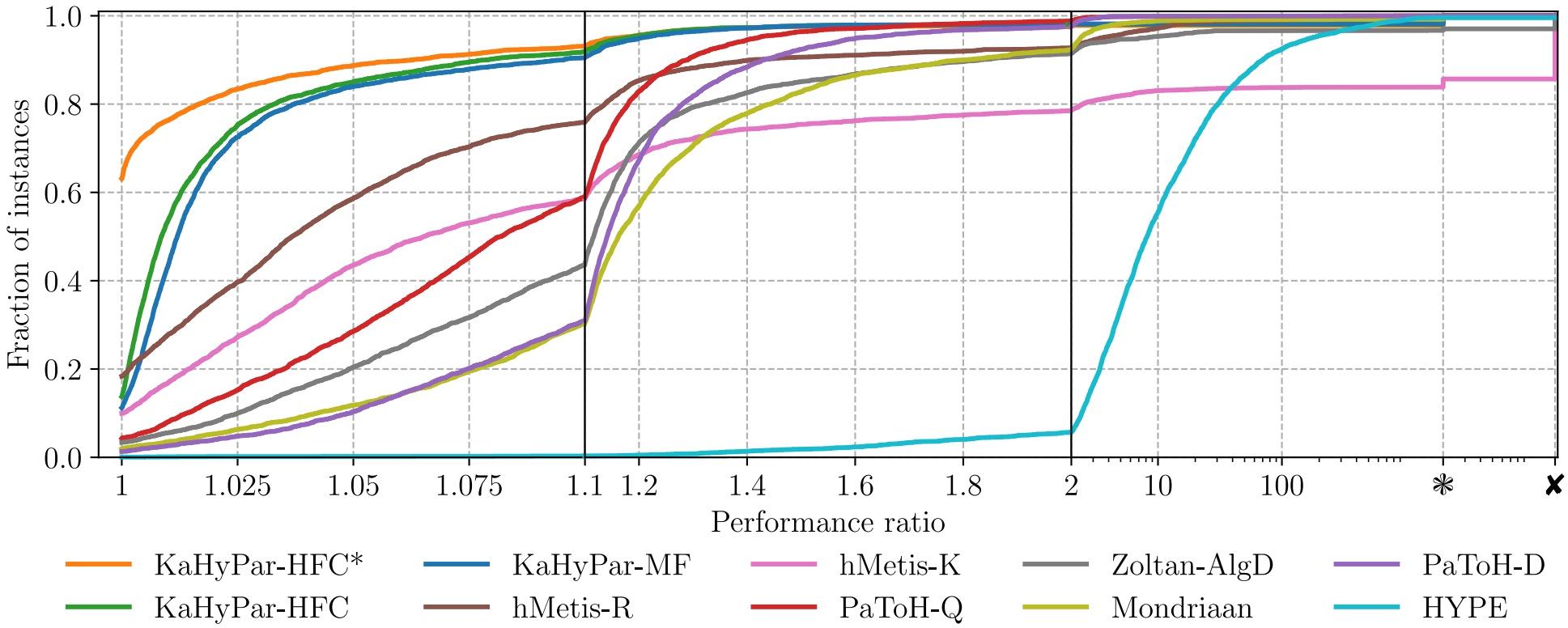
Algorithm	$t[s]$
KaHyPar-MF	67.07
KaHyPar-HFC*	62.49
KaHyPar-HFC	44.84

Algorithm	$t[s]$
KaHyPar-CA (no flows)	$\sim$ 40

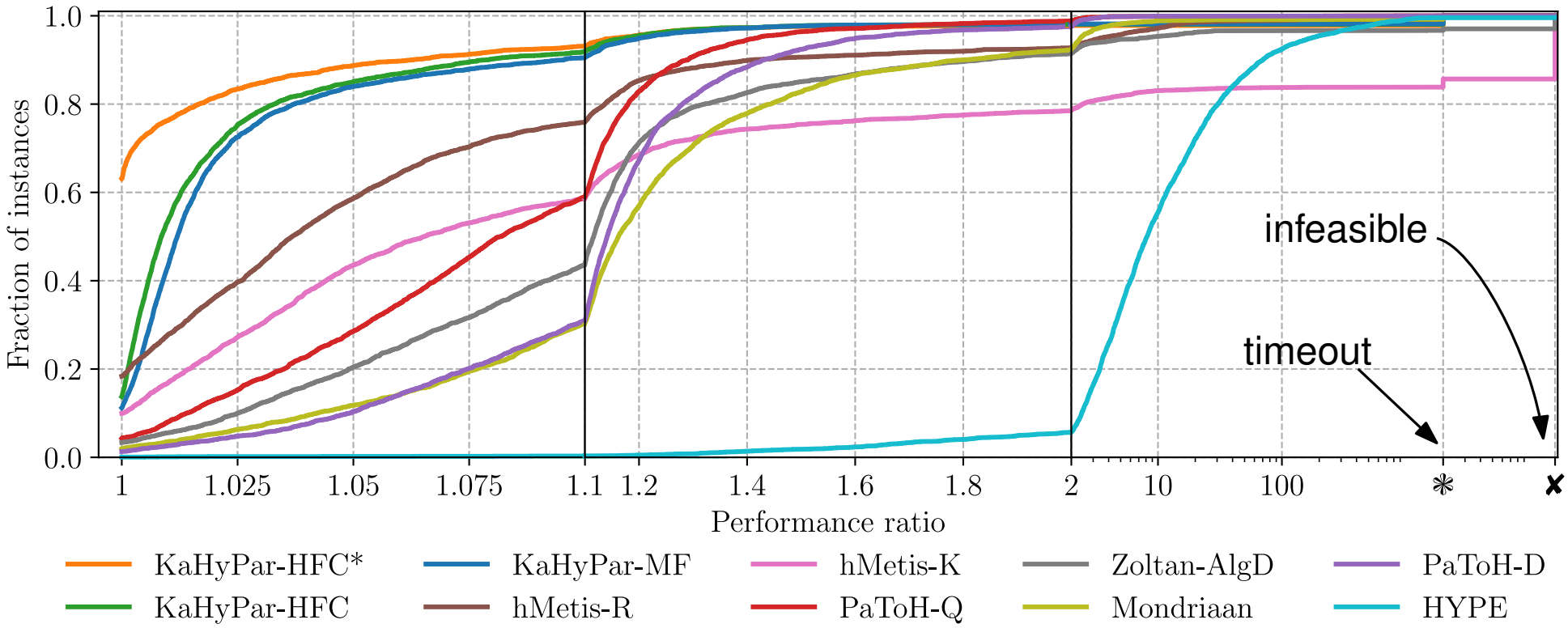
take with a grain of salt!

Institute of Theoretical Informatics
Algorithmics Group

Comparison with other Partitioners

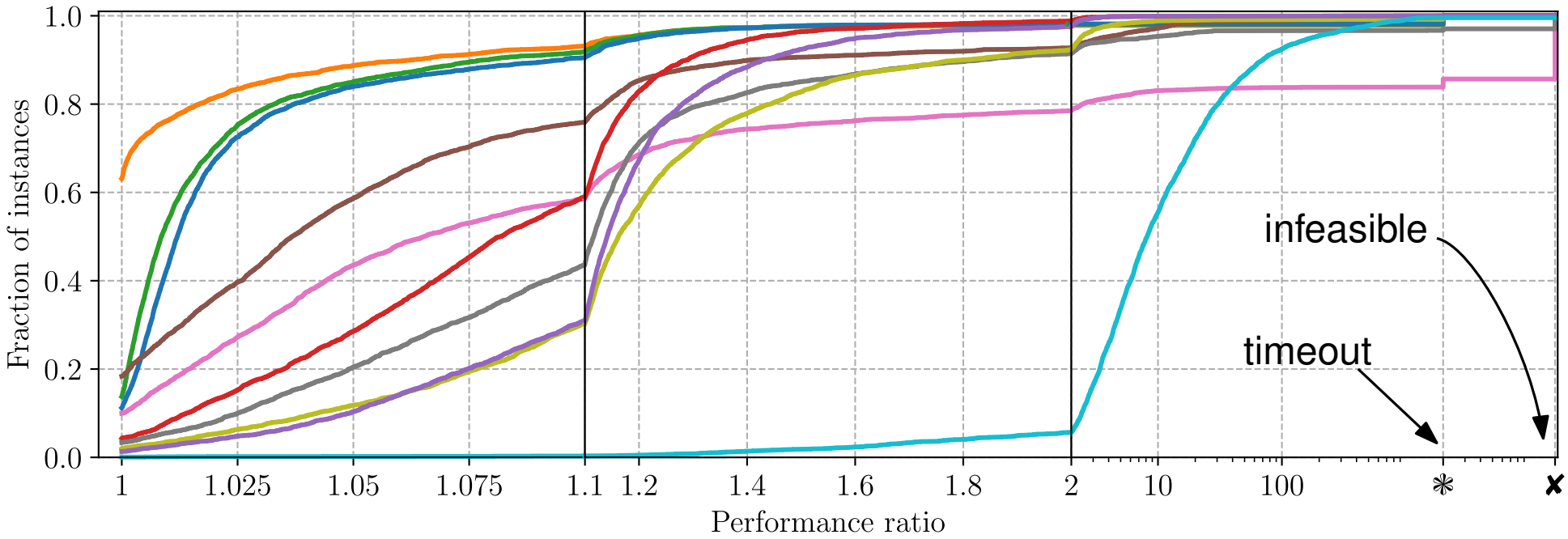


Comparison with other Partitioners



Institute of Theoretical Informatics
Algorithmics Group

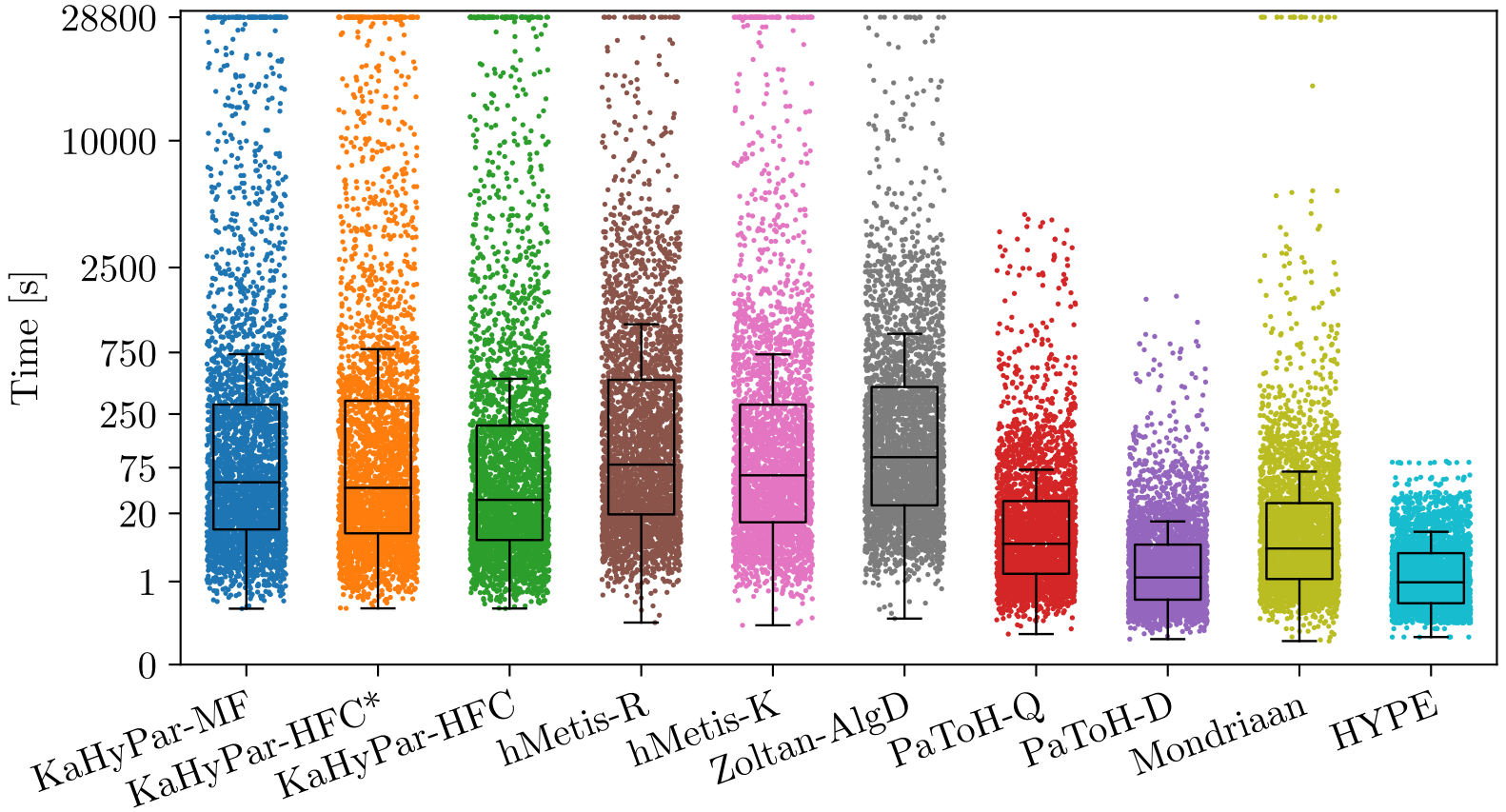
Comparison with other Partitioners



Algorithm	$t[s]$
KaHyPar-MF	67.07
KaHyPar-HFC*	62.49
KaHyPar-HFC	44.84
hMetis-R	96.55
hMetis-K	73.65

Algorithm	$t[s]$
Zoltan-AlgD	107.60
PaToH-Q	7.48
PaToH-D	1.47
Mondriaan	6.44
HYPE	1.03

Detailed Running Time



Institute of Theoretical Informatics
Algorithmics Group

Conclusion

KaHyPar-HFC – better and faster

Institute of Theoretical Informatics
Algorithmics Group

Conclusion

KaHyPar-HFC – better and faster

In the TR

- configuration study / assess impact of components
- different k / instance classes $\Rightarrow$ improves a lot on dual SAT and large k
- earlier balance with subset sum for special vertices
- distance-based piercing
- flow routing

Institute of Theoretical Informatics
Algorithmics Group

Conclusion

KaHyPar-HFC – better and faster

In the TR

- configuration study / assess impact of components
- different k / instance classes $\Rightarrow$ improves a lot on dual SAT and large k
- earlier balance with subset sum for special vertices
- distance-based piercing
- flow routing

`https://kahypar.org`

`https://github.com/kahypar/kahypar`

`https://github.com/larsgottesbueren/WHFC`

Institute of Theoretical Informatics
Algorithmics Group

Conclusion

KaHyPar-HFC – better and faster

In the TR

- configuration study / assess impact of components
- different k / instance classes $\Rightarrow$ improves a lot on dual SAT and large k
- earlier balance with subset sum for special vertices
- distance-based piercing
- flow routing

- parallel versions coming soon^{TM}

`https://kahypar.org`

`https://github.com/kahypar/kahypar`

`https://github.com/larsgottesbueren/WHFC`

Institute of Theoretical Informatics
Algorithmics Group