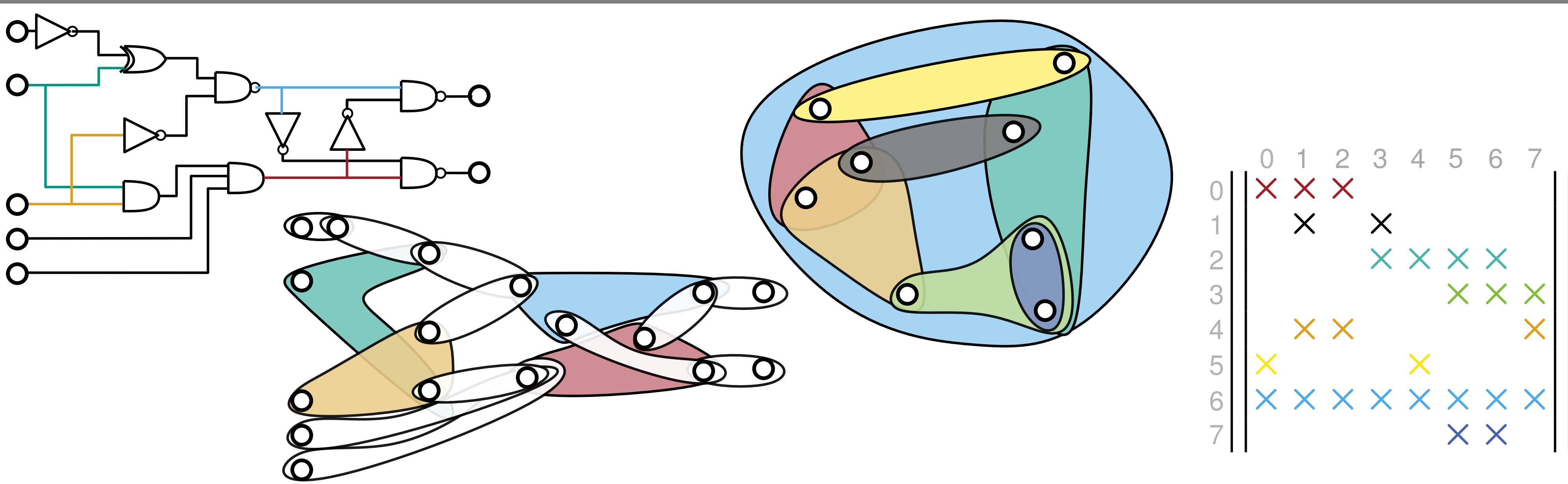
Engineering a direct k-way Hypergraph Partitioning Algorithm

ALENEX'17 · January 17, 2017

Yaroslav Akhremtsev, Tobias Heuer, Peter Sanders, Sebastian Schlag

INSTITUTE OF THEORETICAL INFORMATICS · ALGORITHMICS GROUP

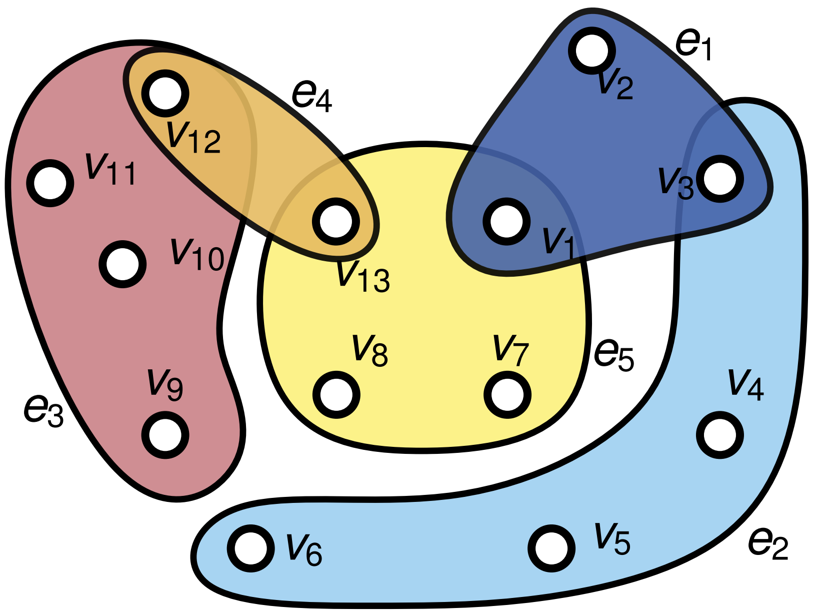


Hypergraphs

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

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

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



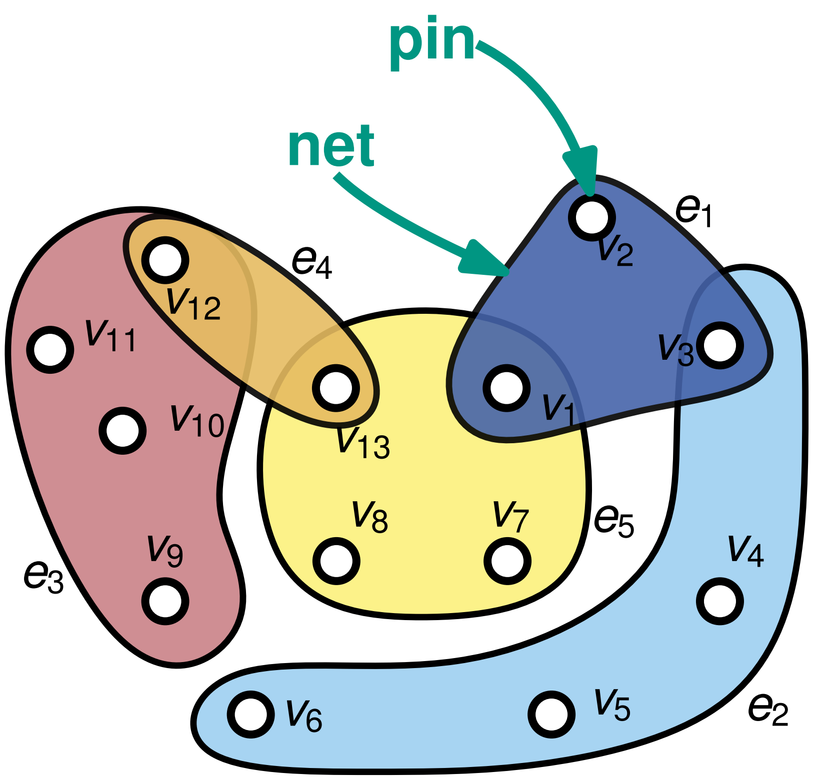
Hypergraphs

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

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



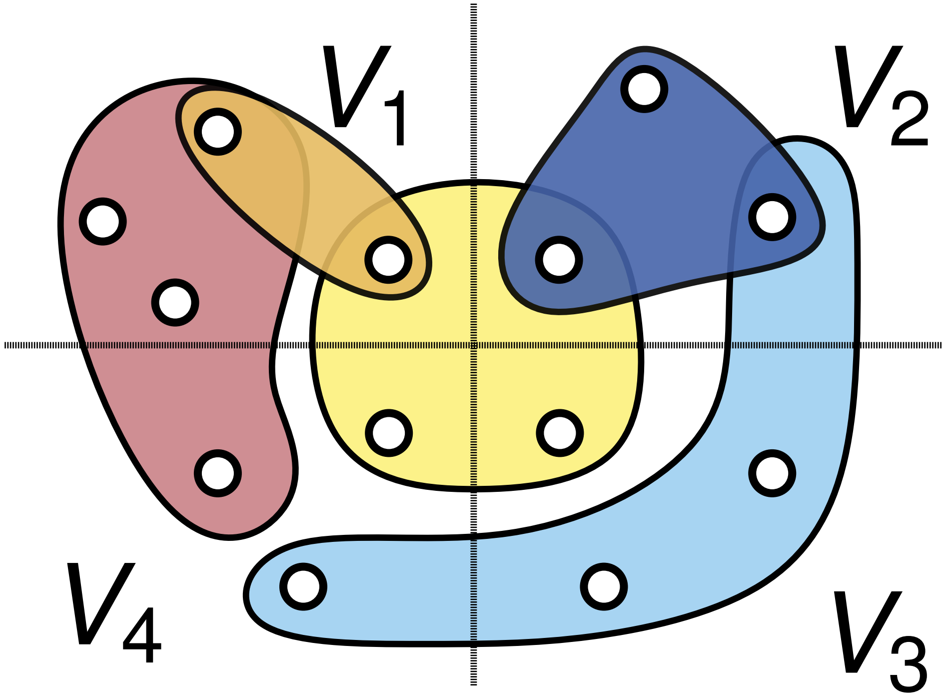
Institute of Theoretical Informatics
Algorithmics Group

Hypergraph Partitioning Problem

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

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

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



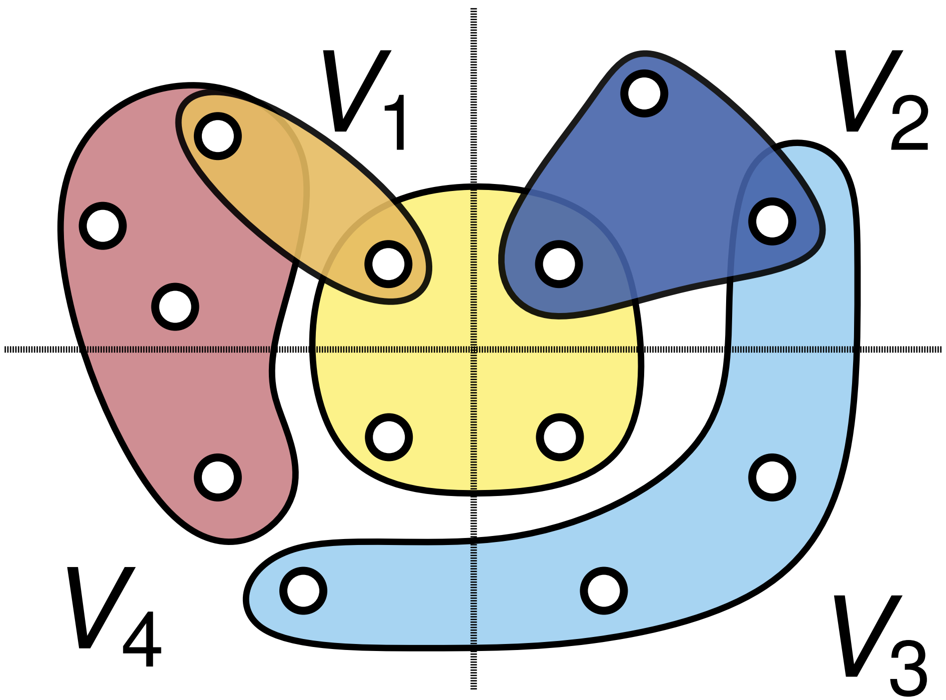
Hypergraph Partitioning Problem

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

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



Hypergraph Partitioning Problem

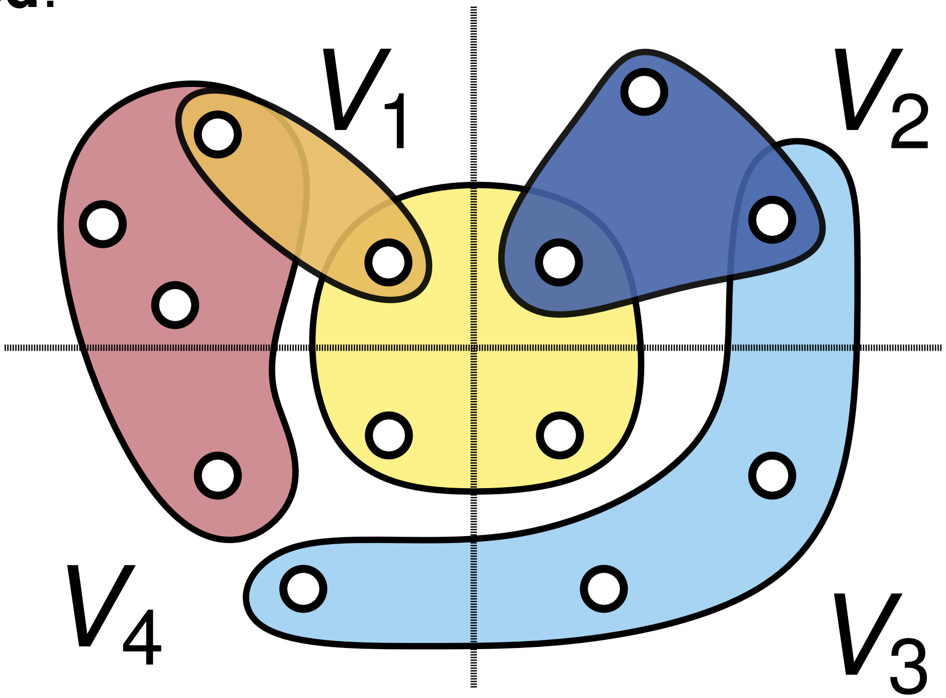
Partition hypergraph $H = (V, E, c, \omega)$ into $\mathbf{k}$ disjoint blocks $\Pi = \{V_1, \ldots, V_k\}$ such that:

- 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**:



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**:

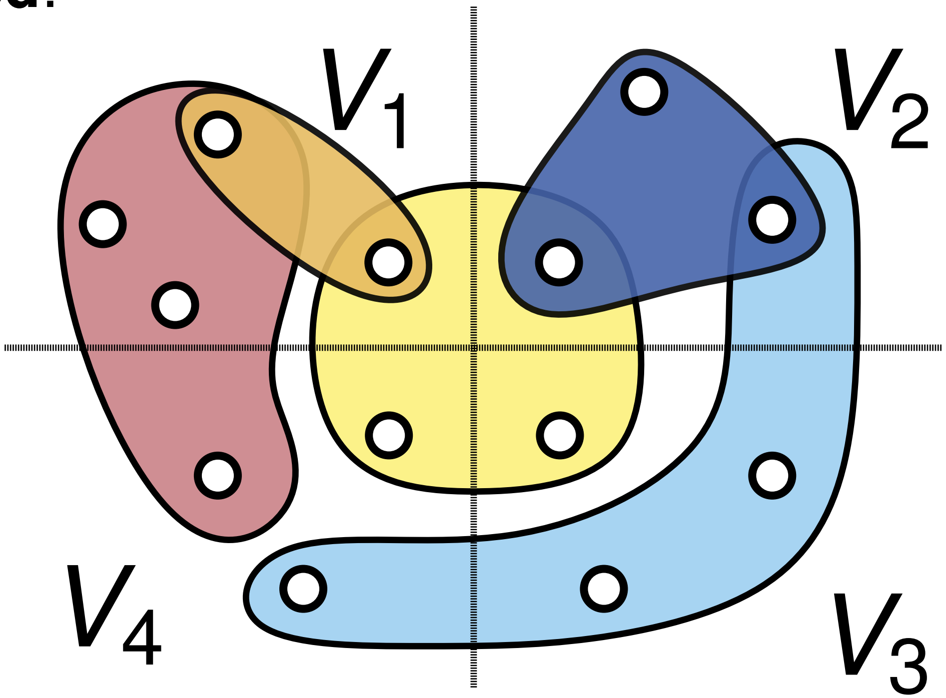
$$c(V_i) \leq (1 + \varepsilon) \left\lceil \frac{c(V)}{k} \right\rceil$$

imbalance parameter

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

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

connectivity:
 # blocks connected by net e



Hypergraph Partitioning Problem

Karlsruhe Institute of Technology

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**:

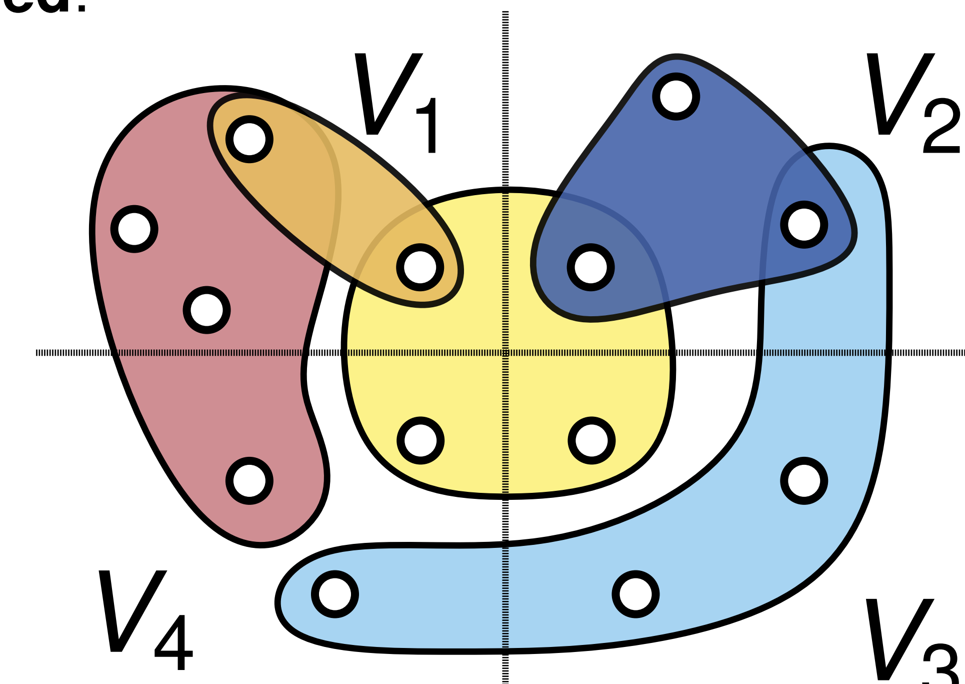
$$c(V_i) \leq (1 + \varepsilon) \left\lceil \frac{c(V)}{k} \right\rceil$$

imbalance parameter

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

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

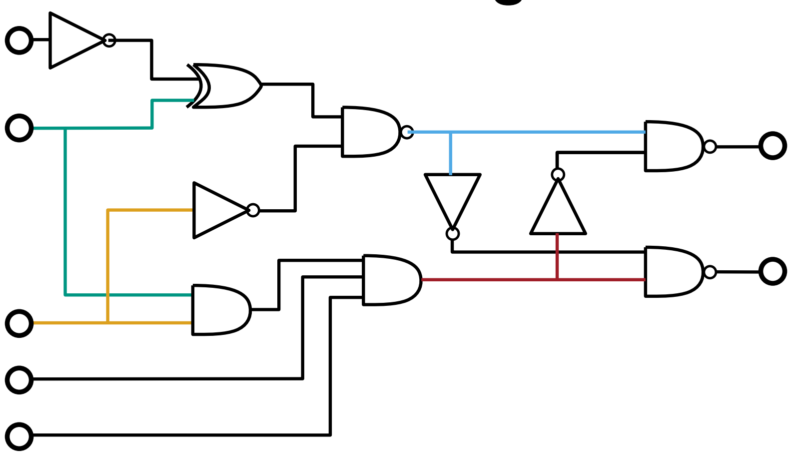
connectivity:
 # blocks connected by net e



Institute of Theoretical Informatics
Algorithmics Group

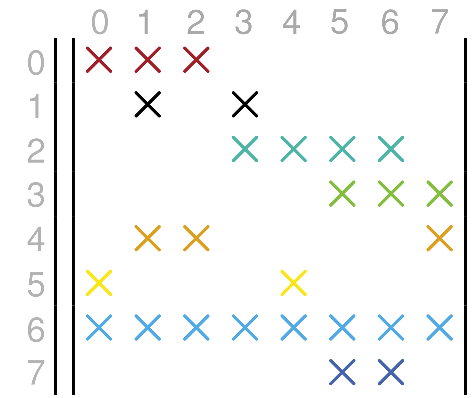
Applications

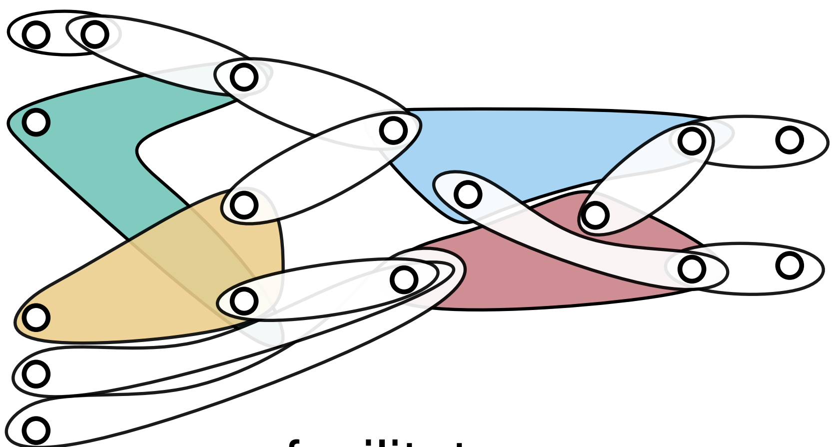
VLSI Design



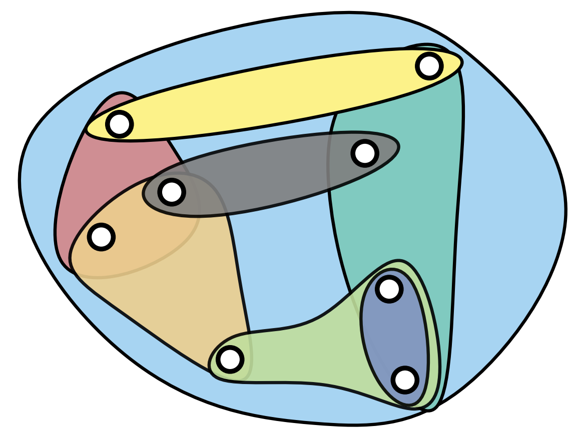
Application Domain

Scientific Computing





Hypergraph Model

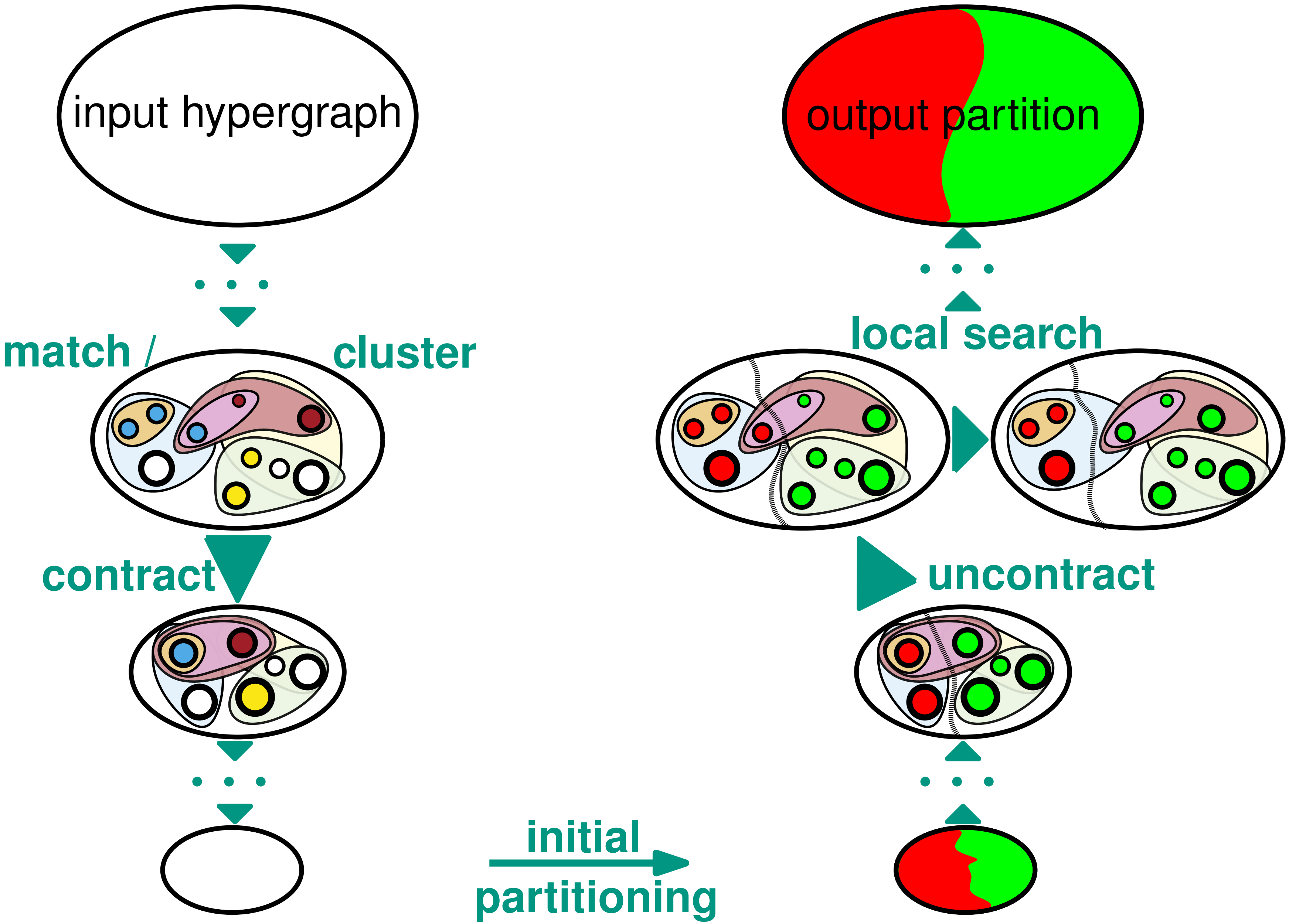


facilitate floorplanning & placement

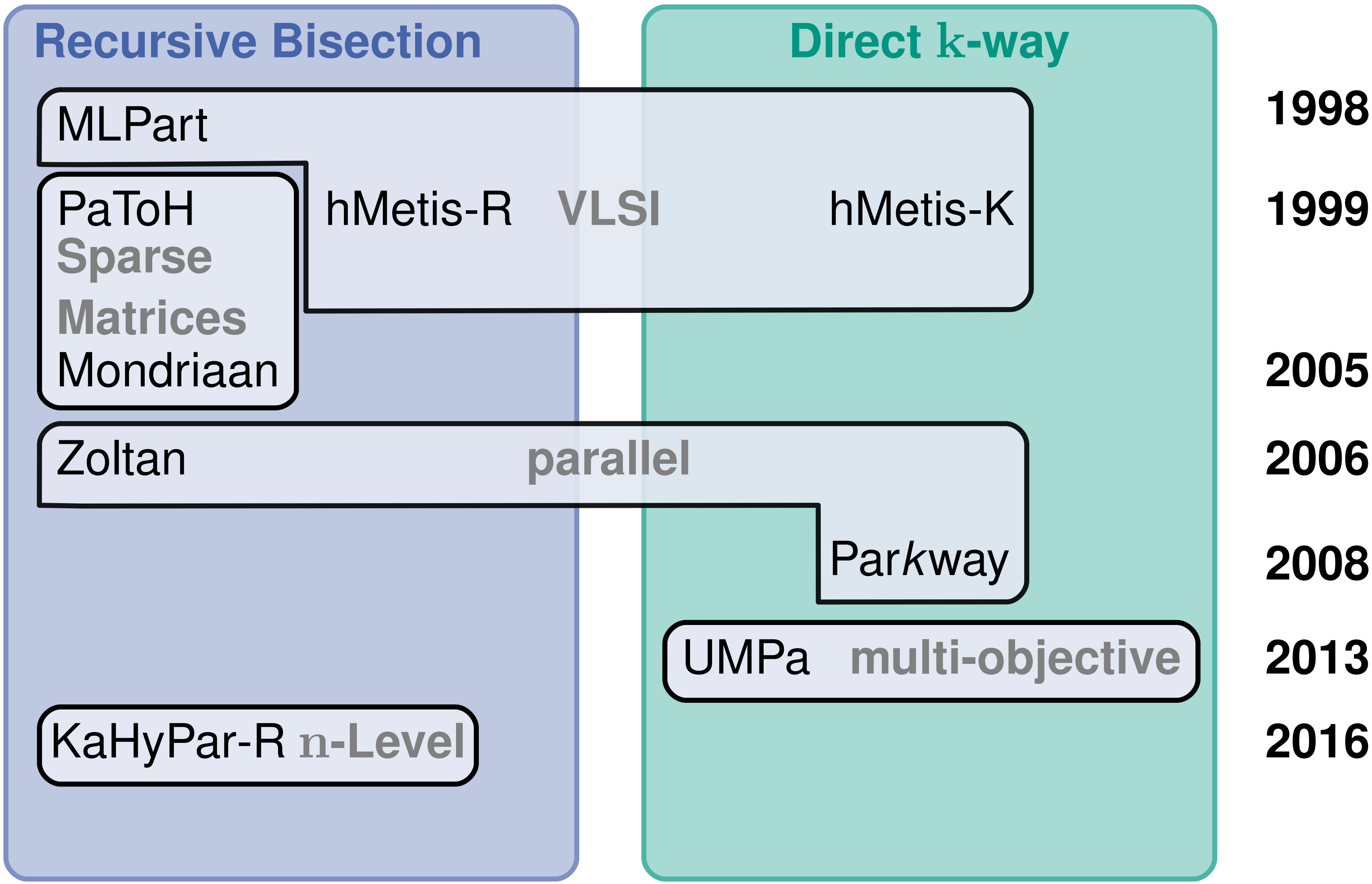
Goal

minimize communication

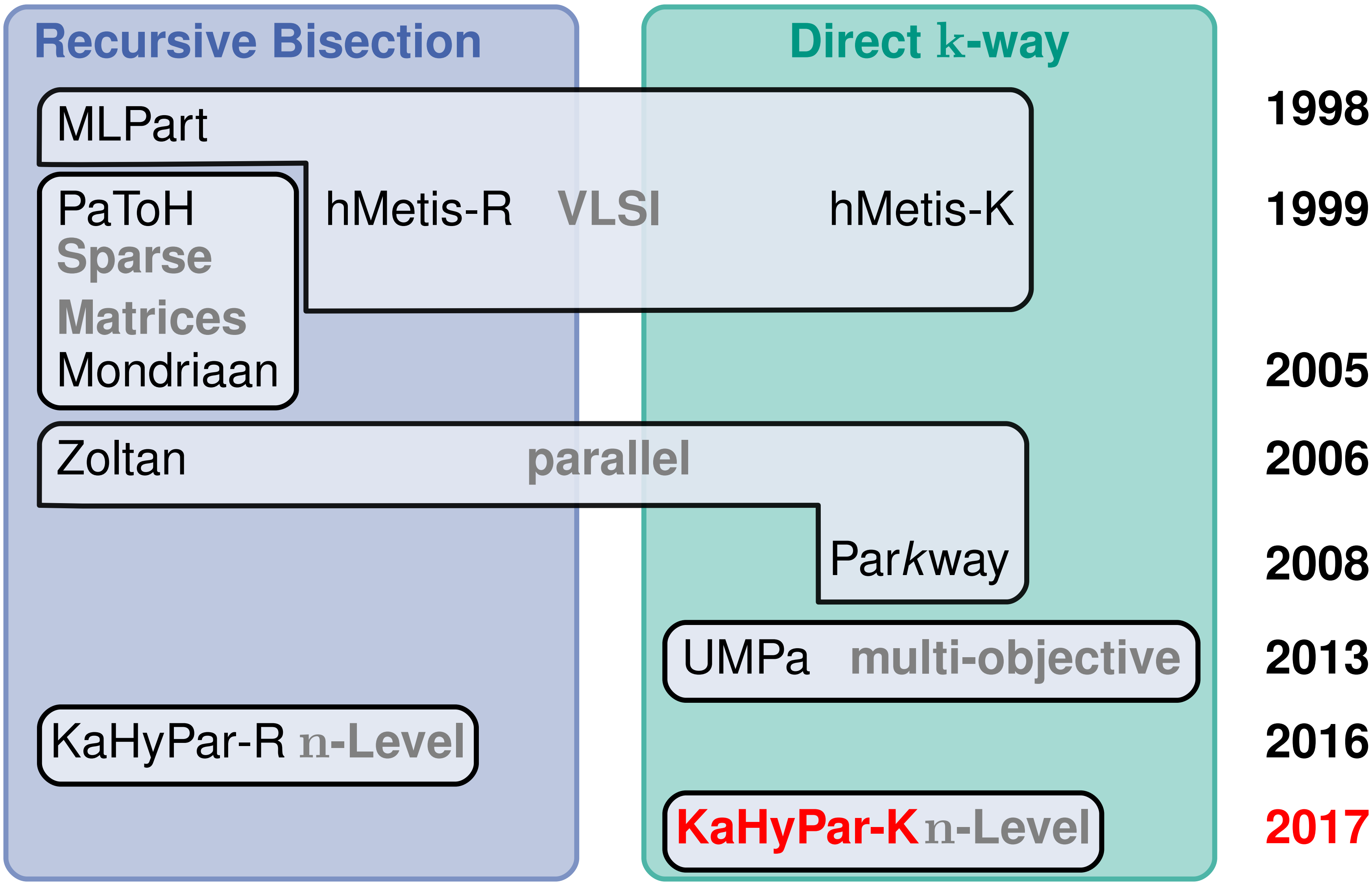
The Multilevel Framework



Taxonomy of Hypergraph Partitioning Tools



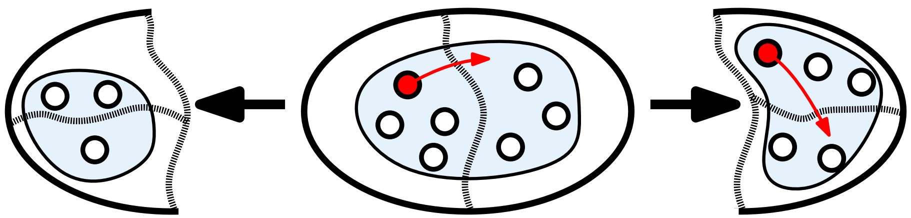
Taxonomy of Hypergraph Partitioning Tools



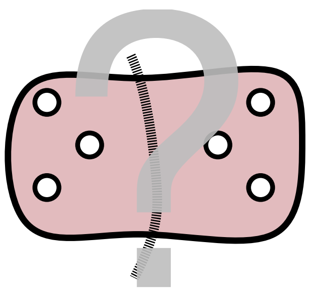
Institute of Theoretical Informatics
Algorithmics Group

Why look at direct *k*-way partitioning?

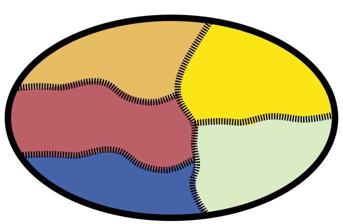
Recursive Bisection



restricted solution space

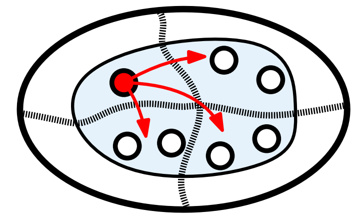


local search in **large** nets ✗

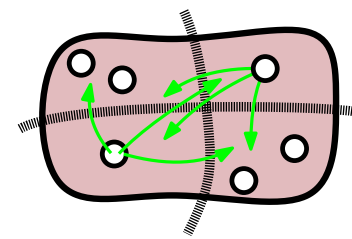


adaptive imbalance adjustment

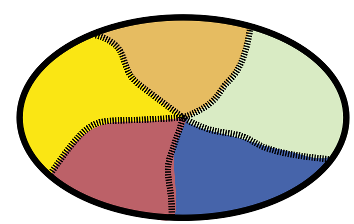
Direct k-way



global view of **all** *k* blocks



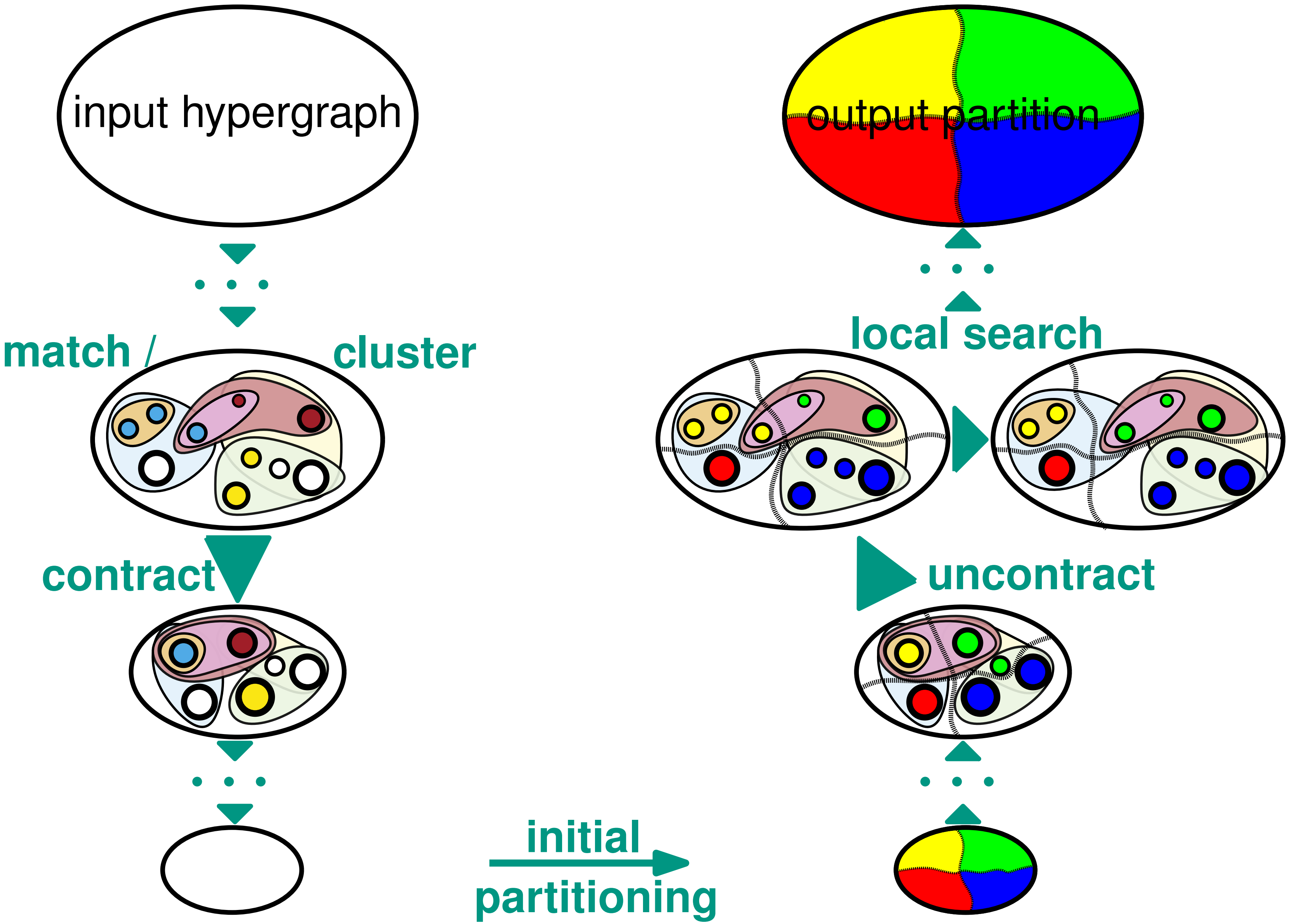
local search in **large** nets ✓



direct imbalance enforcement

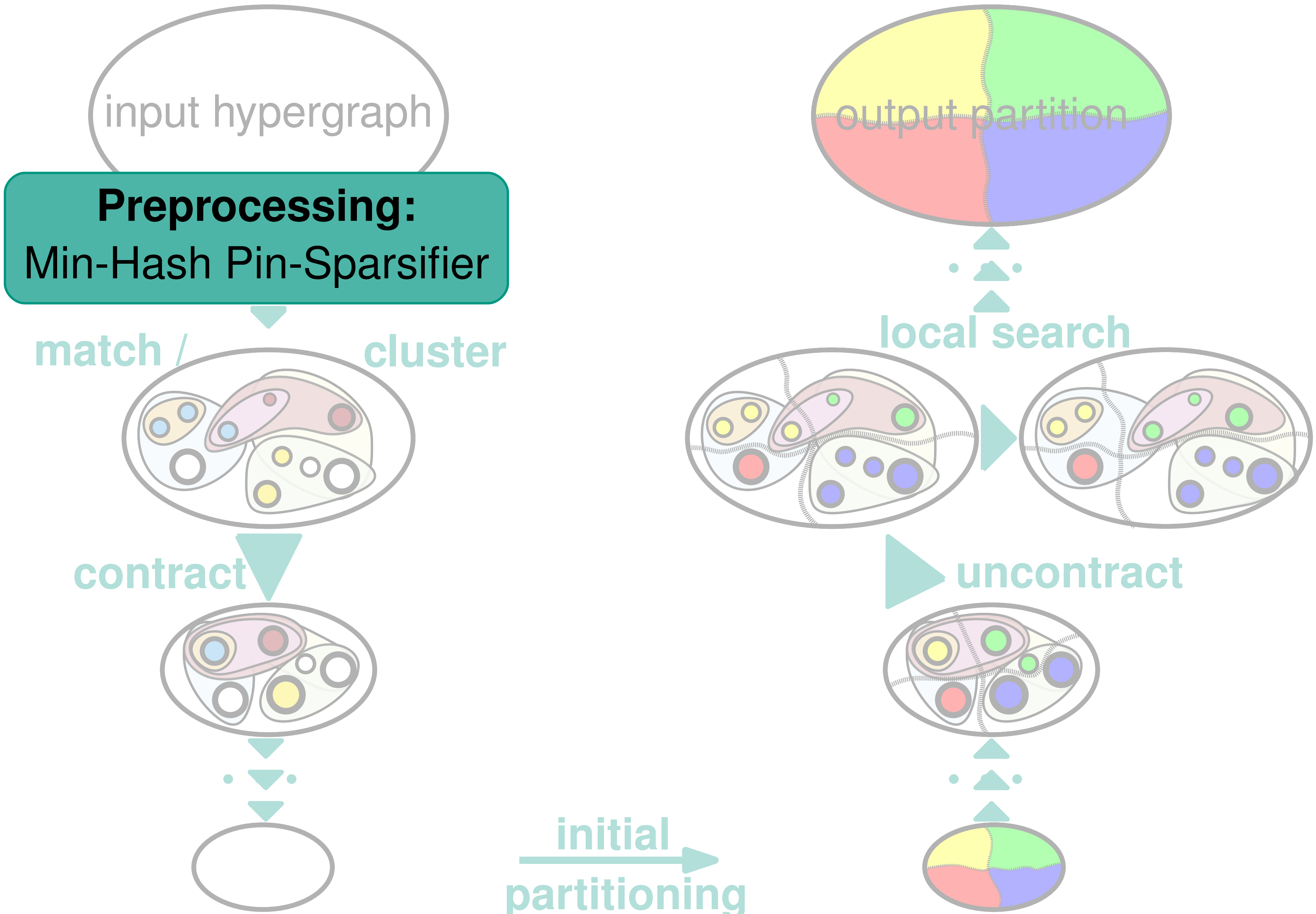
Institute of Theoretical Informatics
Algorithmics Group

Our Contributions

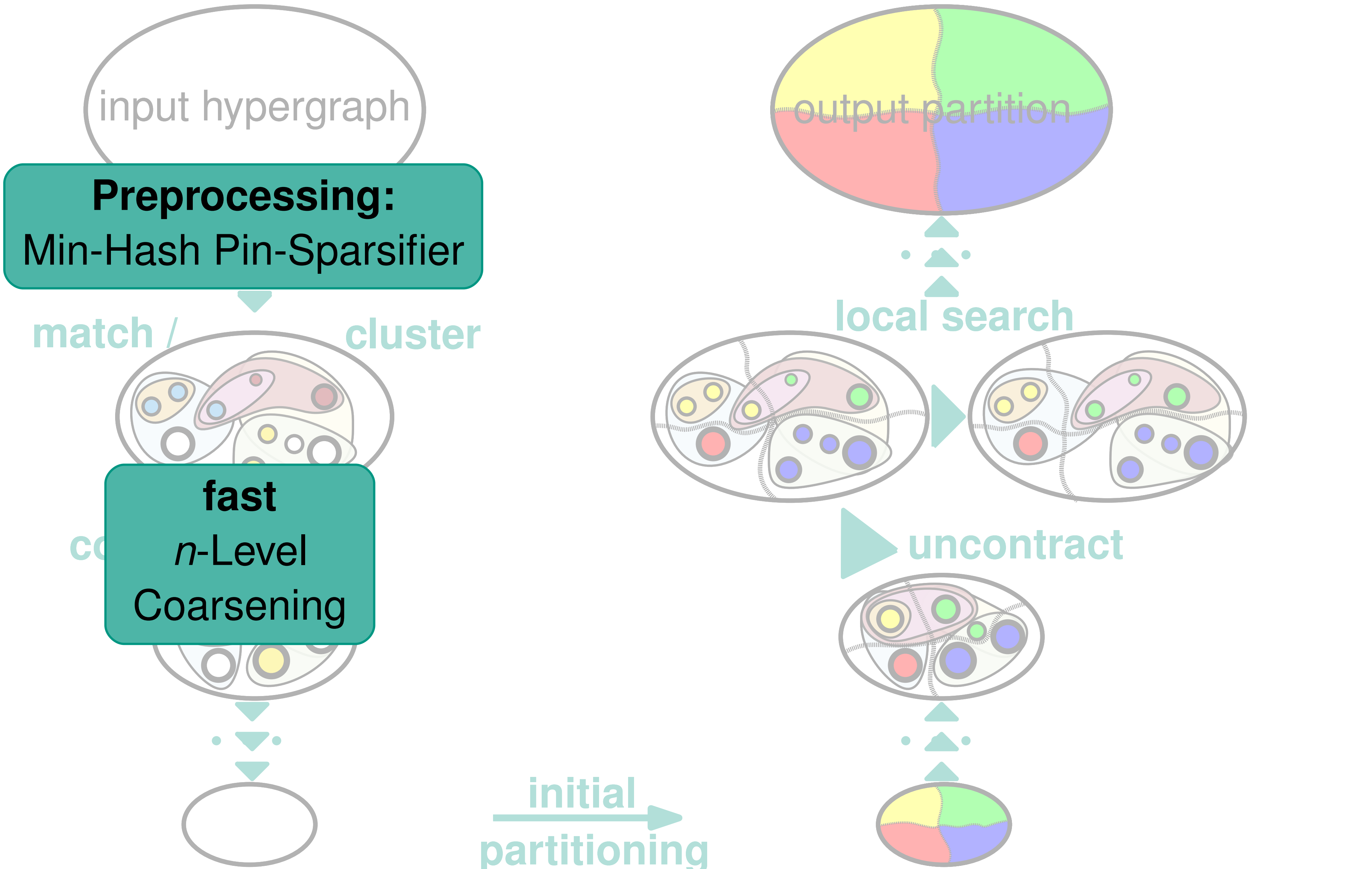


Institute of Theoretical Informatics
Algorithmics Group

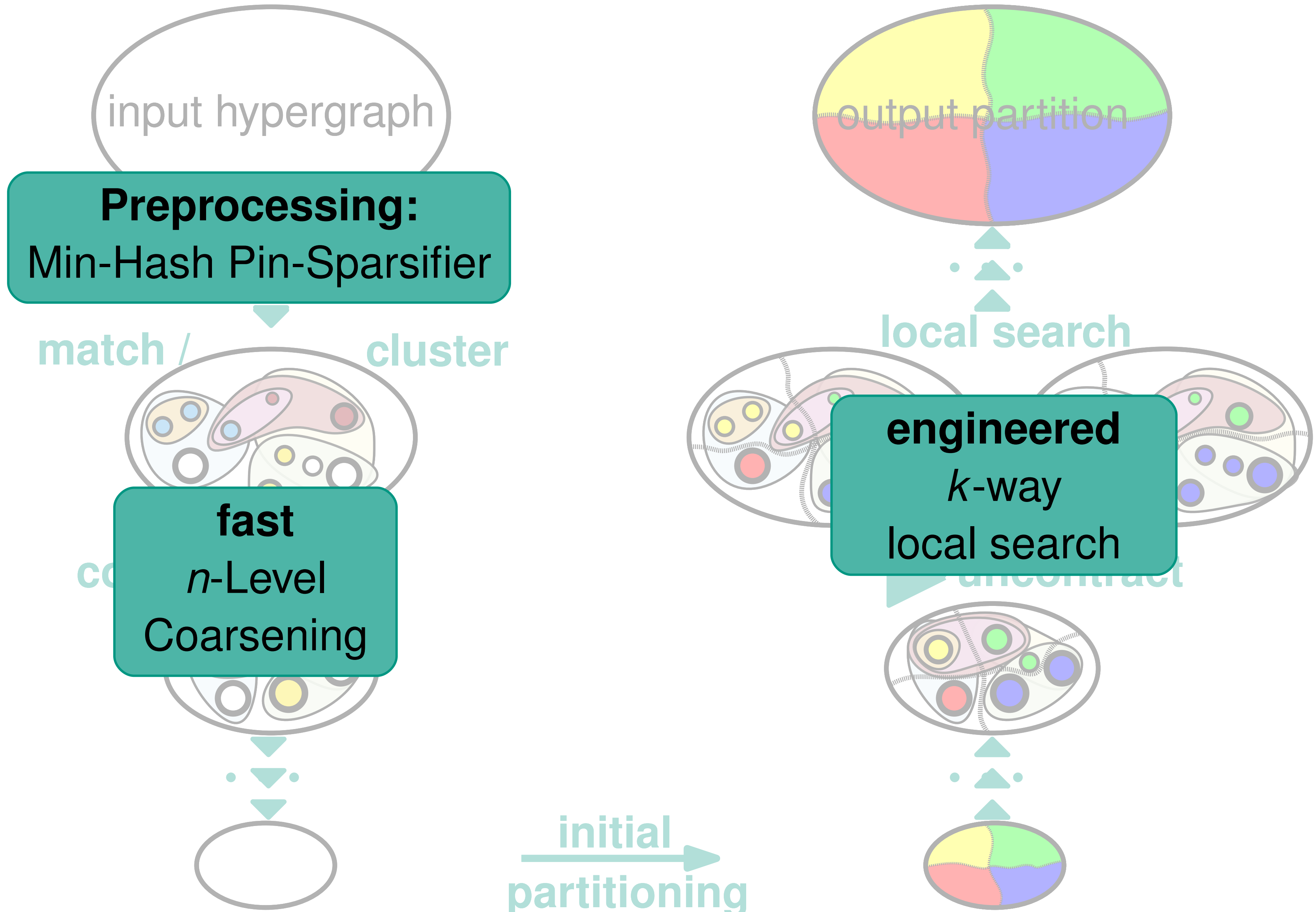
Our Contributions



Our Contributions



Our Contributions



Institute of Theoretical Informatics
Algorithmics Group

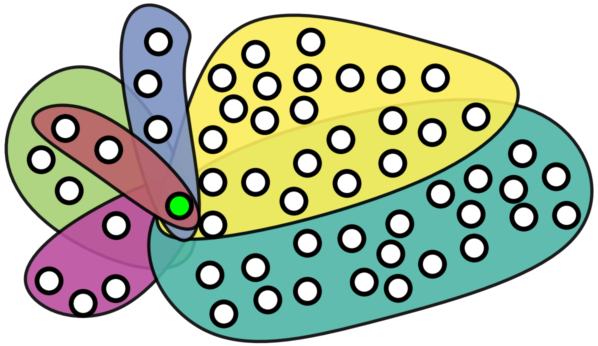
Preprocessing

Institute of Theoretical Informatics
Algorithmics Group

Min-Hash Based Pin Sparsifier

Motivation: HGP Algorithms contain code like this

foreach *net e incident to v* **do**
  foreach *pin p* $\in$ *e* **do**
    do something

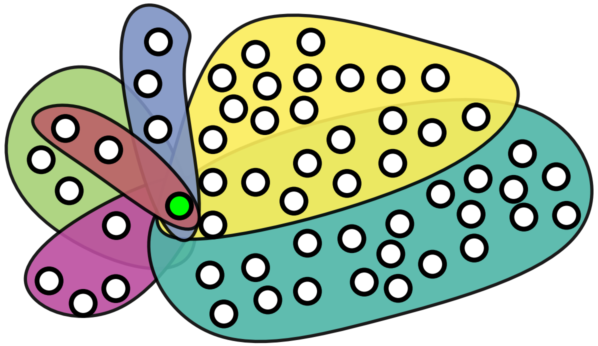


$\Longrightarrow$ large nets $\rightsquigarrow$ large # pins /neighbors $\Longrightarrow$ **slow!**

Min-Hash Based Pin Sparsifier

Motivation: HGP Algorithms contain code like this

foreach *net e incident to v* **do**
  foreach *pin p* $\in$ *e* **do**
    do something



$\Longrightarrow$ large nets $\rightsquigarrow$ large # pins /neighbors $\Longrightarrow$ **slow!**

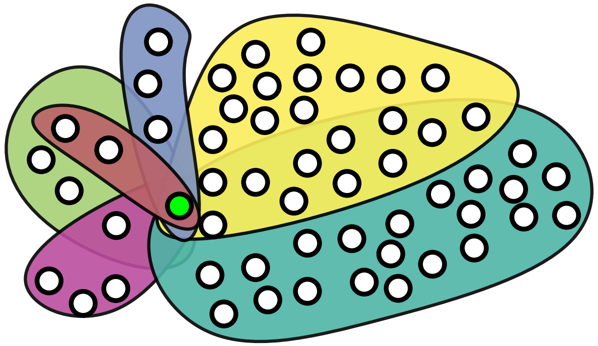
Central Idea: Merge ”close” vertices

Institute of Theoretical Informatics
Algorithmics Group

Min-Hash Based Pin Sparsifier

Motivation: HGP Algorithms contain code like this

foreach *net e incident to v* **do**
  foreach *pin* $p \in e$ **do**
    do something



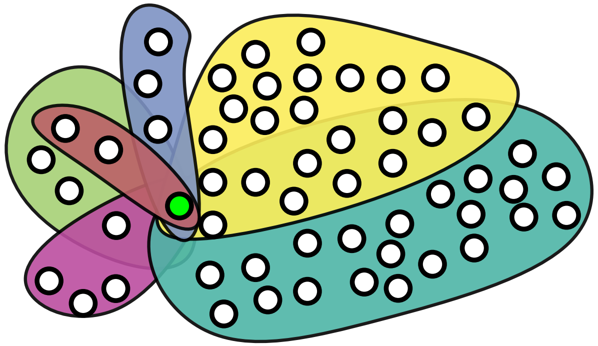
⇒ large nets $\rightsquigarrow$ large # pins /neighbors ⇒ **slow!**

Central Idea: Merge "close" vertices ← share many nets

Min-Hash Based Pin Sparsifier

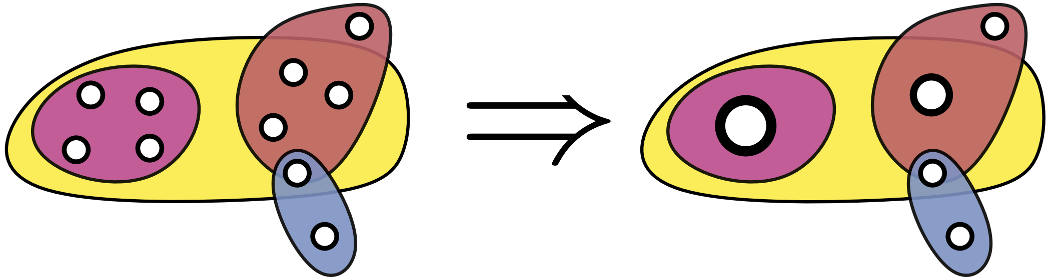
Motivation: HGP Algorithms contain code like this

```
foreach net e incident to v do
    foreach pin p ∈ e do
        do something
```



⟹ large nets ⇝ large # pins /neighbors ⟹ **slow!**

Central Idea: Merge "close" vertices ← share many nets

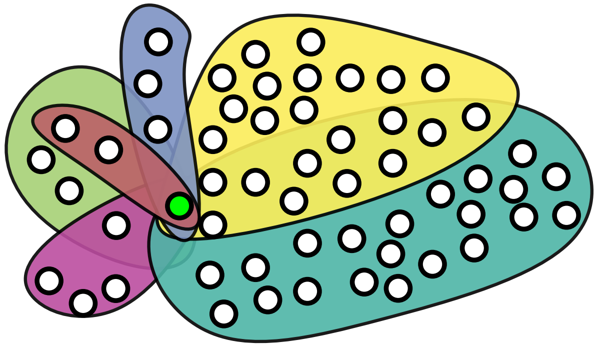


Institute of Theoretical Informatics
Algorithmics Group

Min-Hash Based Pin Sparsifier

Motivation: HGP Algorithms contain code like this

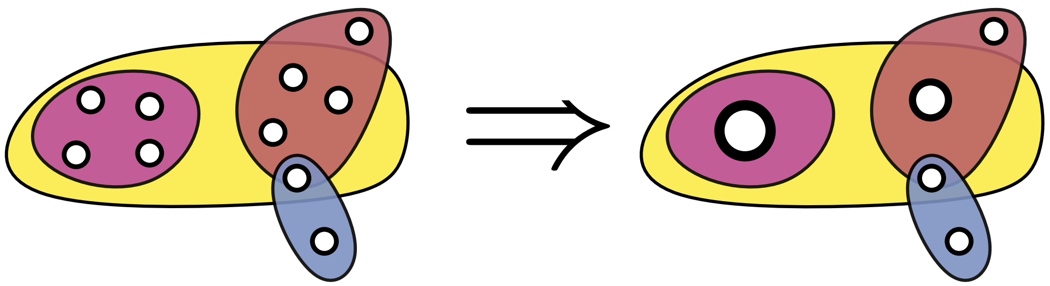
foreach *net e incident to v* **do**
  foreach *pin* $p \in e$ **do**
    do something



$\Longrightarrow$ large nets $\rightsquigarrow$ large # pins /neighbors $\Longrightarrow$ **slow!**

Central Idea: Merge "close" vertices ← share many nets

Distance $D(u, v) := 1 - \frac{|I(u) \cap I(v)|}{|I(u) \cup I(v)|}$



Min-Hash Based Pin Sparsifier

Problem: set operations are expensive!

Solution:

- **approximate** $\frac{|I(u) \cap I(v)|}{|I(u) \cup I(v)|}$ via **min-hash** fingerprints [Broder'97]
- merge vertices with equal fingerprint

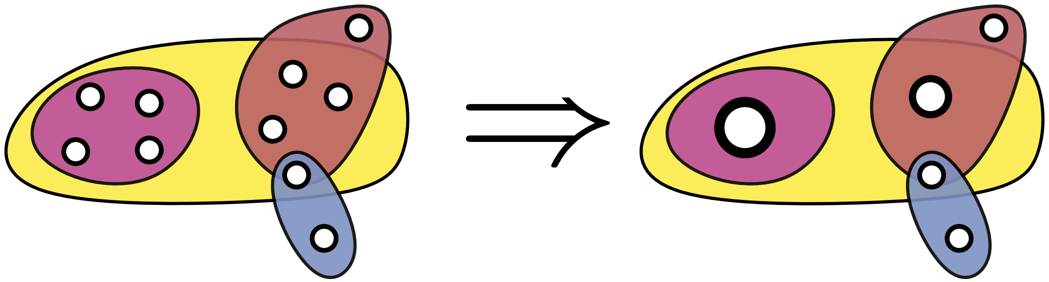
Institute of Theoretical Informatics
Algorithmics Group

Min-Hash Based Pin Sparsifier

Problem: set operations are expensive!

Solution:

- **approximate** $\frac{|I(u) \cap I(v)|}{|I(u) \cup I(v)|}$ via **min-hash** fingerprints [Broder'97]
- merge vertices with equal fingerprint



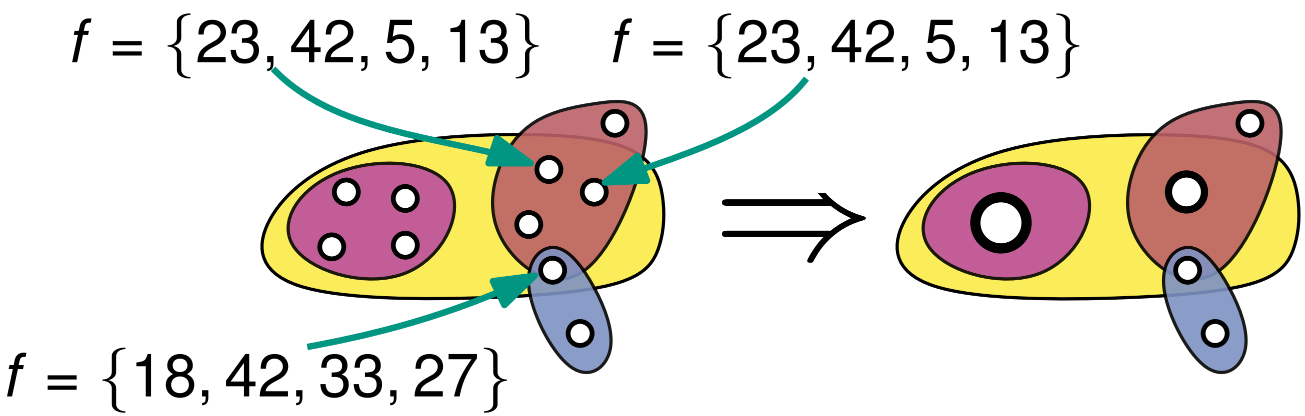
Institute of Theoretical Informatics
Algorithmics Group

Min-Hash Based Pin Sparsifier

Problem: set operations are expensive!

Solution:

- **approximate** $\frac{|I(u) \cap I(v)|}{|I(u) \cup I(v)|}$ via **min-hash** fingerprints [Broder'97]
- merge vertices with equal fingerprint



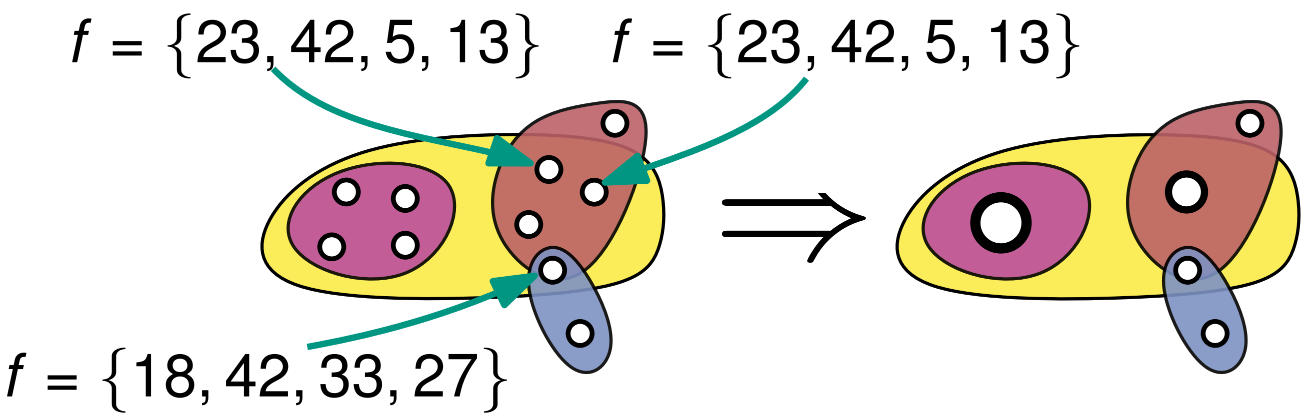
Institute of Theoretical Informatics
Algorithmics Group

Min-Hash Based Pin Sparsifier

Problem: set operations are expensive!

Solution:

- **approximate** $\frac{|I(u) \cap I(v)|}{|I(u) \cup I(v)|}$ via **min-hash** fingerprints [Broder'97]
- merge vertices with equal fingerprint



$$\textbf{fingerprint}(v) = \{h_1(v), h_2(v), h_3(v), \ldots\}$$

$$h_x(v) := \min\{\sigma(e) \mid e \in I(v)\}$$

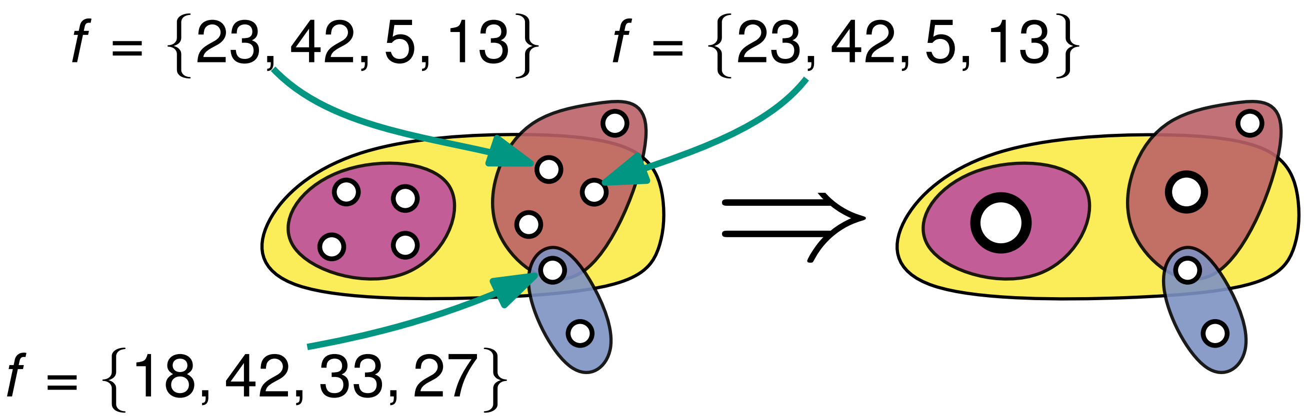
Institute of Theoretical Informatics
Algorithmics Group

Min-Hash Based Pin Sparsifier

Problem: set operations are expensive!

Solution:

- **approximate** $\frac{|I(u) \cap I(v)|}{|I(u) \cup I(v)|}$ via **min-hash** fingerprints [Broder'97]
- merge vertices with equal fingerprint



$$\textbf{fingerprint}(v) = \{h_1(v), h_2(v), h_3(v), \ldots\}$$

$$h_x(v) := \min\{\sigma(e) | e \in I(v)\}$$

- Running time: $\mathcal{O}(|P|)$

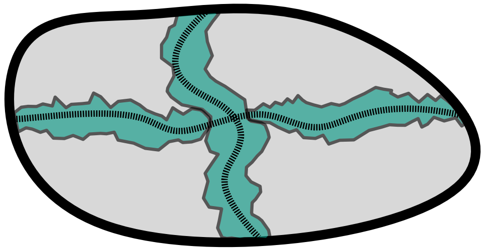
Institute of Theoretical Informatics
Algorithmics Group

Local Search

Localized adaptive *k*-way Local Search

Current direct *k*-way multilevel HGP tools:

- uncontract one **level**
- ⇝ **simple greedy** local search around border



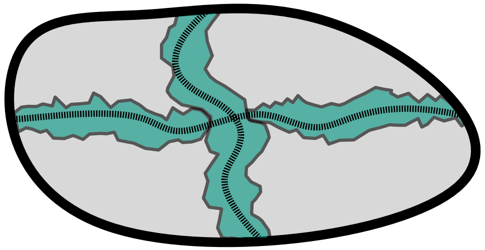
cannot escape from local optima!

Institute of Theoretical Informatics
Algorithmics Group

Localized adaptive *k*-way Local Search

Current direct *k*-way multilevel HGP tools:

- uncontract one **level**
- ⇝ **simple greedy** local search around border



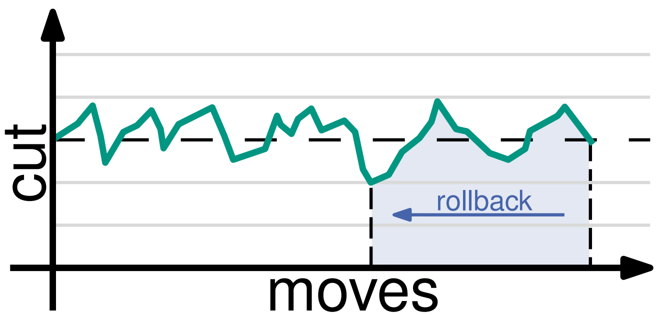
cannot escape from local optima!

Advanced alternatives exist:

Algorithm 1: FM Local Search

---

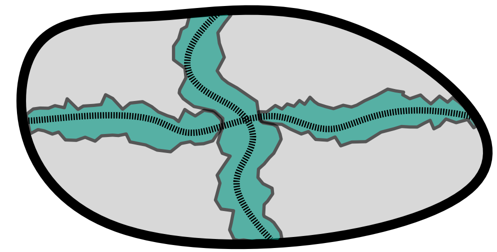
while ¬ *done* **do**
  `find` best move
  `perform` best move
`rollback` to best solution



can worsen solution

Institute of Theoretical Informatics
Algorithmics Group

Localized adaptive *k*-way Local Search



Current direct *k*-way multilevel HGP tools:

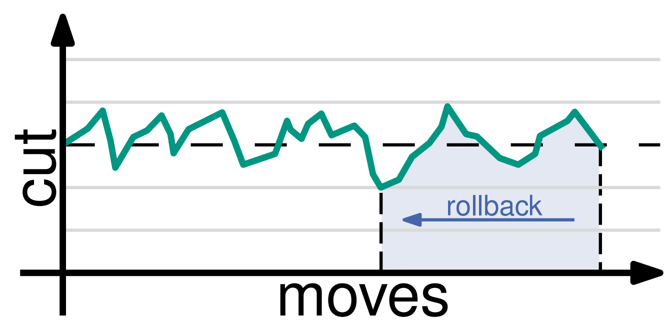
- uncontract one **level**
- ⇝ **simple greedy** local search around border

cannot escape from local optima!

Advanced alternatives exist:

Algorithm 1: FM Local Search

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



can worsen solution

Reason for sticking with greedy:

⇒ existing *k*-way FM algorithm [Sanchis] is **slow**!

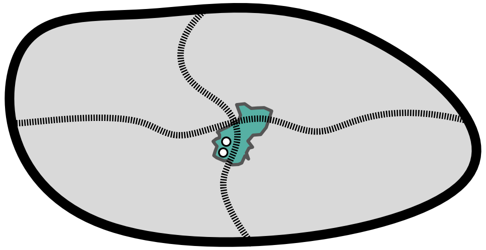
⇒ **not** evaluated in multilevel context!

Localized adaptive k-way Local Search

Our algorithm:

- uncontract **a single vertex pair** $\rightsquigarrow$ local search around **2** nodes
- **simplified**
- **fast**
- **n-level**

} version of Sanchis' algorithm



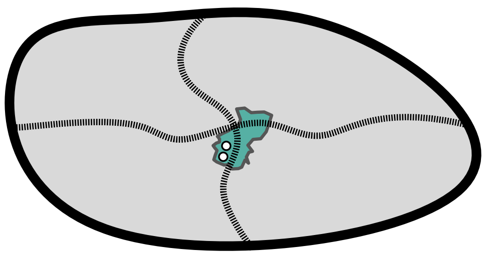
Institute of Theoretical Informatics
Algorithmics Group

Localized adaptive *k*-way Local Search

Our algorithm:

- uncontract **a single vertex pair** $\rightsquigarrow$ local search around **2** nodes
- **simplified**
- **fast**
- **n-level**

} version of Sanchis' algorithm



Simplifications/Improvements:

- **reduce** # PQs: $k(k-1) \rightsquigarrow k$
- **reduce** # moves: only consider **adjacent** blocks
- **cache** gain values
- **stop** unpromising search early
- **exclude** nets from gain updates

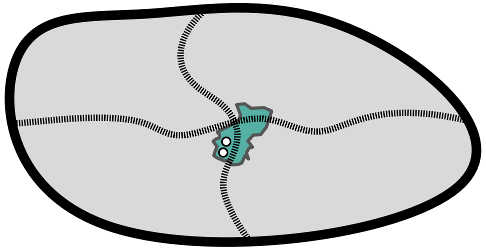
Institute of Theoretical Informatics
Algorithmics Group

Localized adaptive *k*-way Local Search

Our algorithm:

- uncontract **a single vertex pair** $\rightsquigarrow$ local search around **2** nodes
- **simplified**
- **fast**
- **n-level**

} version of Sanchis' algorithm

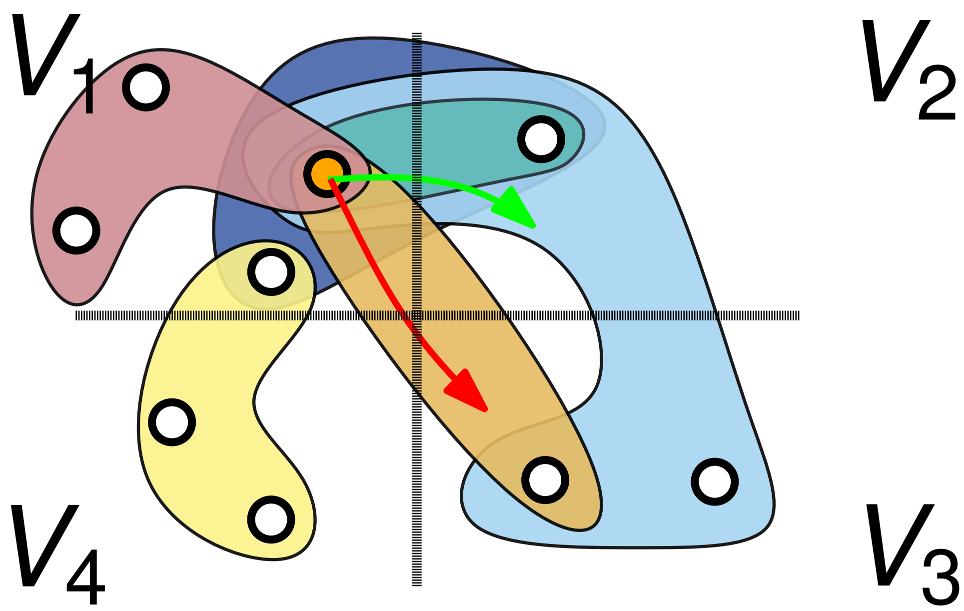


Simplifications/Improvements:

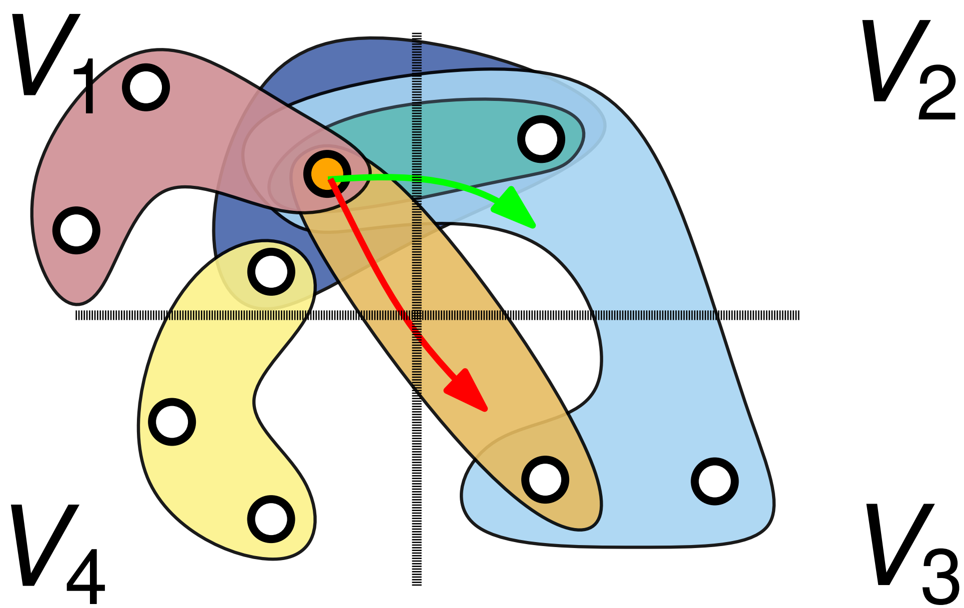
- **reduce** # PQs: $k(k-1) \rightsquigarrow k$
- **reduce** # moves: only consider **adjacent** blocks
- **cache** gain values
- **stop** unpromising search early
- **exclude** nets from gain updates

Institute of Theoretical Informatics
Algorithmics Group

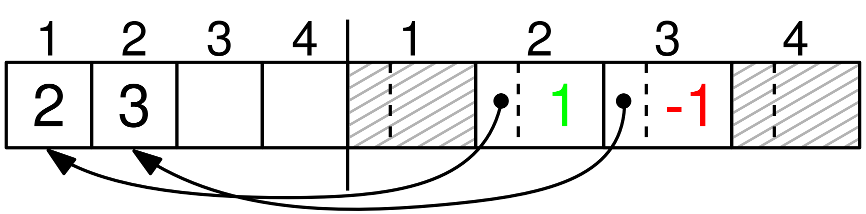
k-way Gain Cache - Key Concepts



k-way Gain Cache - Key Concepts



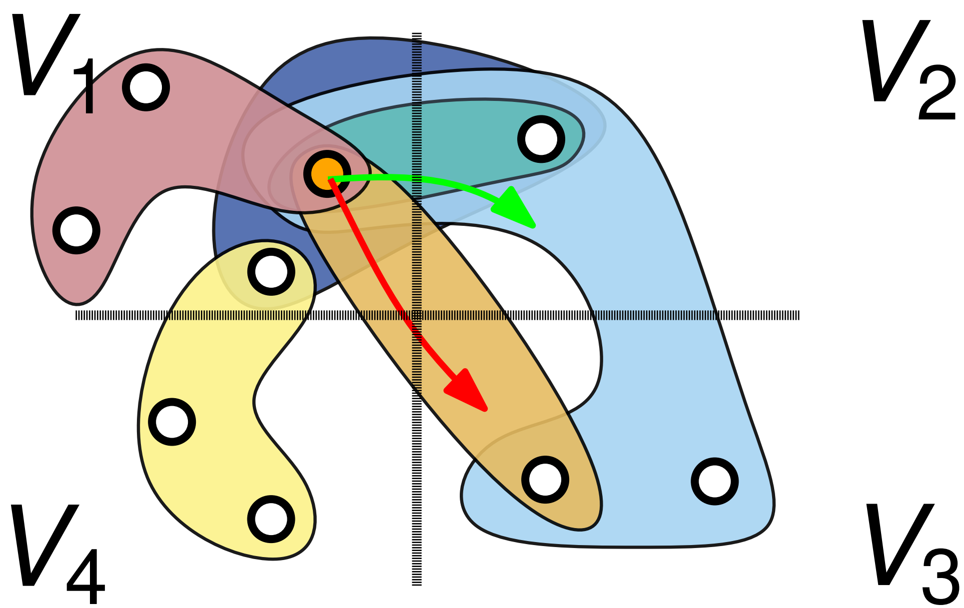
Gain-Cache of ● :



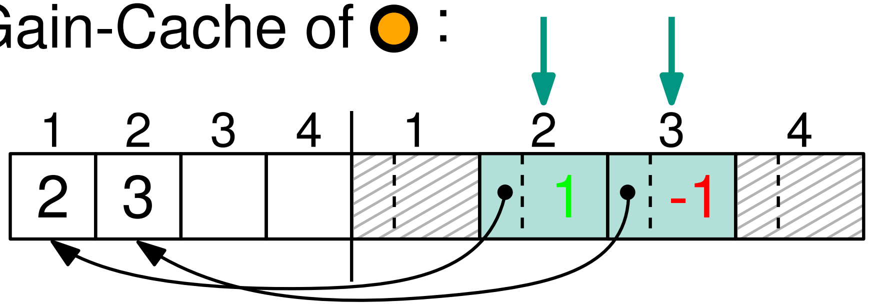
Sparse Set [Briggs and Torczon]

- $\mathcal{O}(1)$ insert/remove/update
- linear time iteration

k-way Gain Cache - Key Concepts



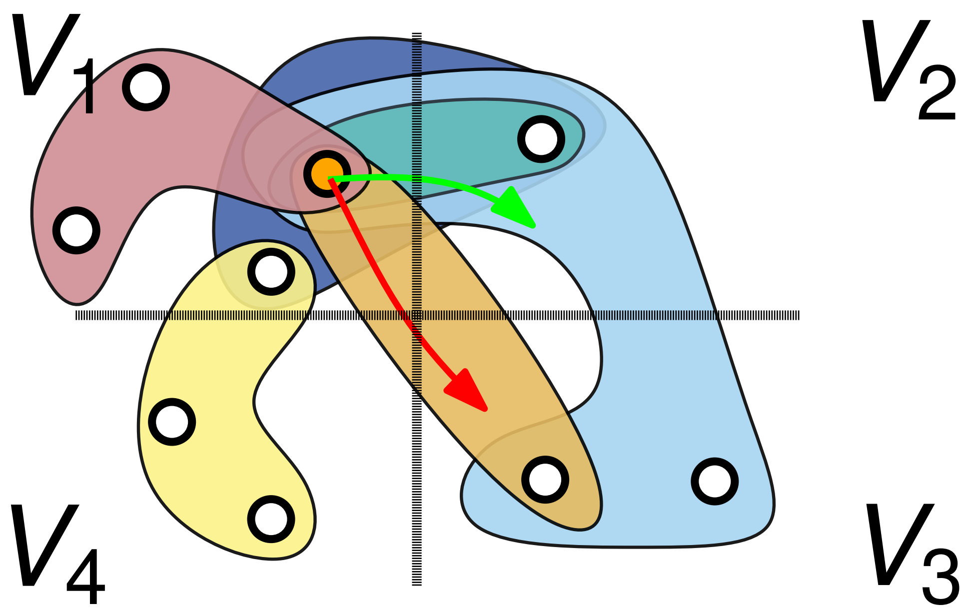
Gain-Cache of ● :



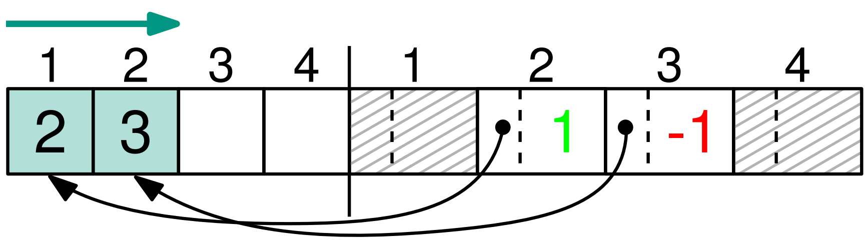
Sparse Set [Briggs and Torczon]

- $\mathcal{O}(1)$ insert/remove/update
- linear time iteration

k-way Gain Cache - Key Concepts



Gain-Cache of ● :

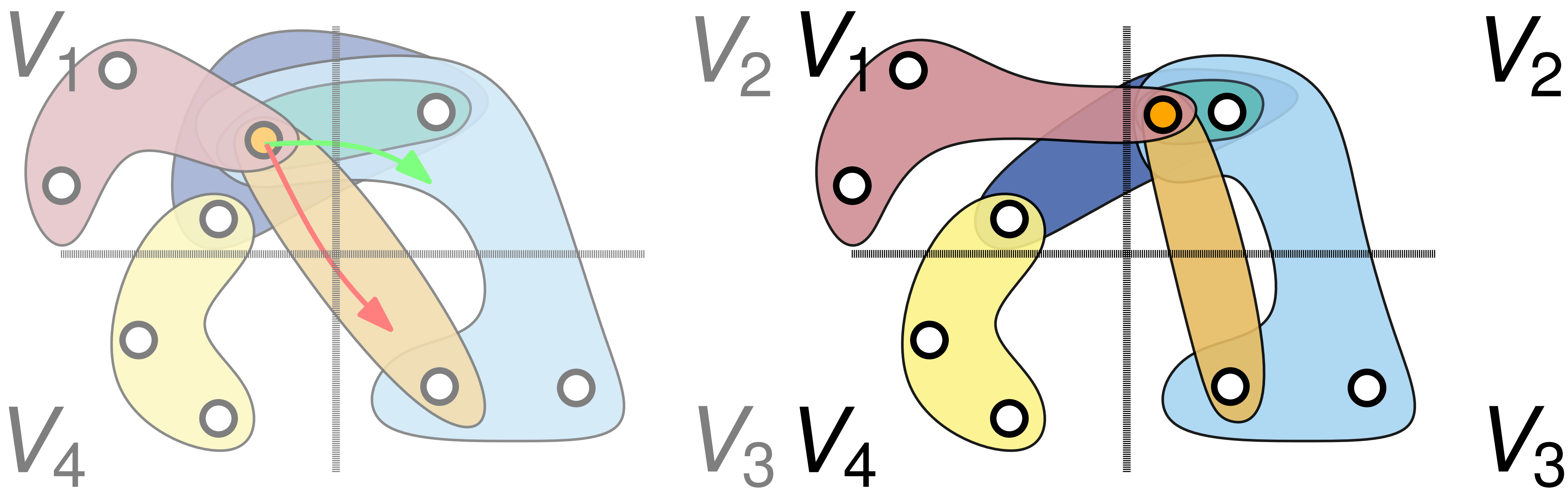


Sparse Set [Briggs and Torczon]

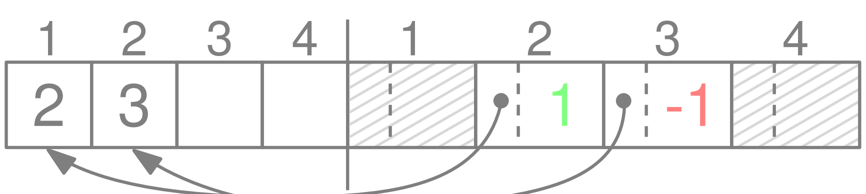
- $\mathcal{O}(1)$ insert/remove/update
- linear time iteration

Institute of Theoretical Informatics
Algorithmics Group

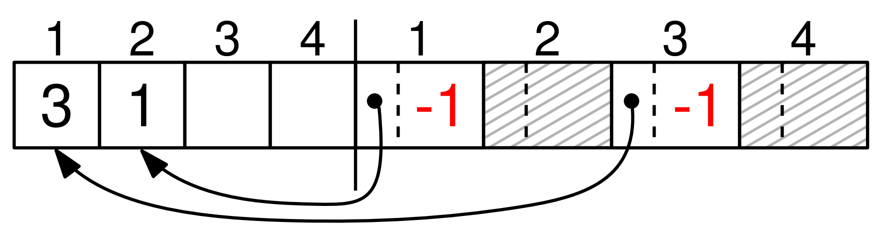
k-way Gain Cache - Key Concepts



Gain-Cache of :



Gain-Cache of :



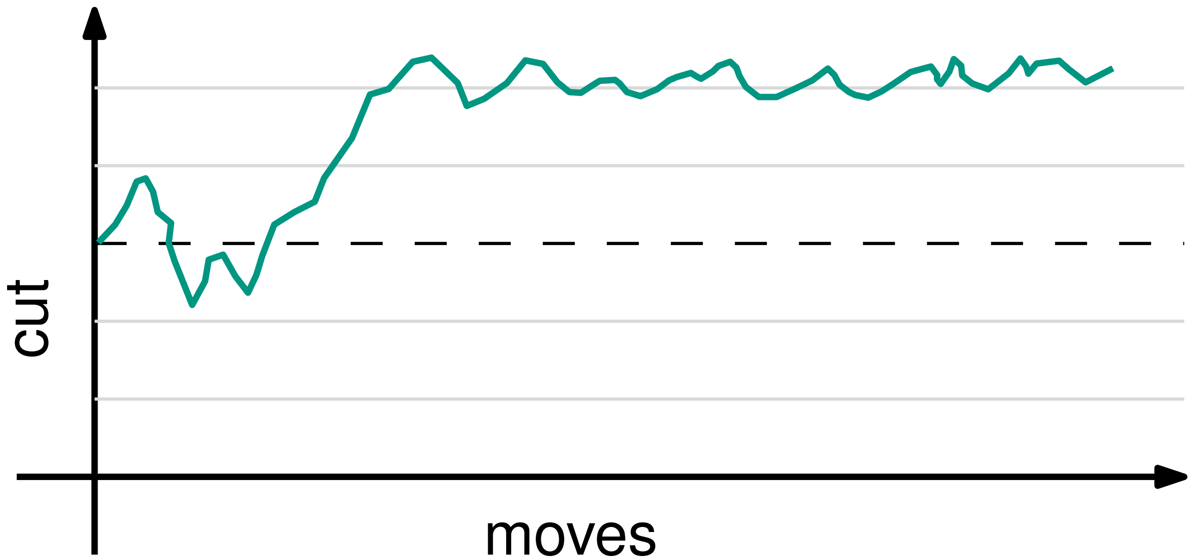
Sparse Set [Briggs and Torczon]

- $\mathcal{O}(1)$ insert/remove/update
- linear time iteration

Institute of Theoretical Informatics
Algorithmics Group

Adaptive Stopping Rule

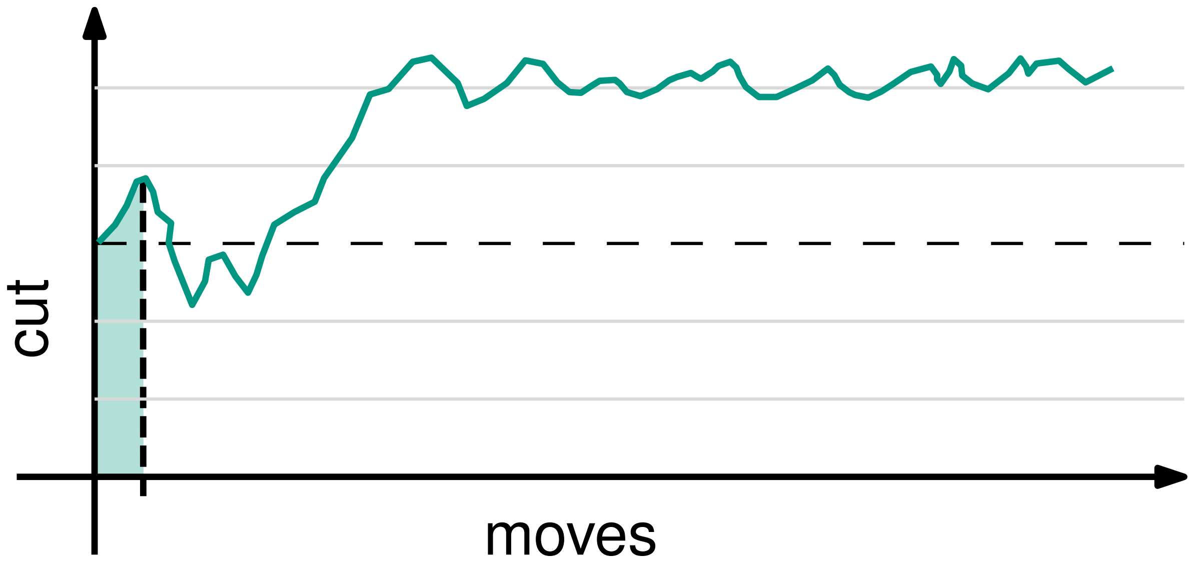
Idea: stop local search if improvement becomes **unlikely** [KaSPar]



Institute of Theoretical Informatics
Algorithmics Group

Adaptive Stopping Rule

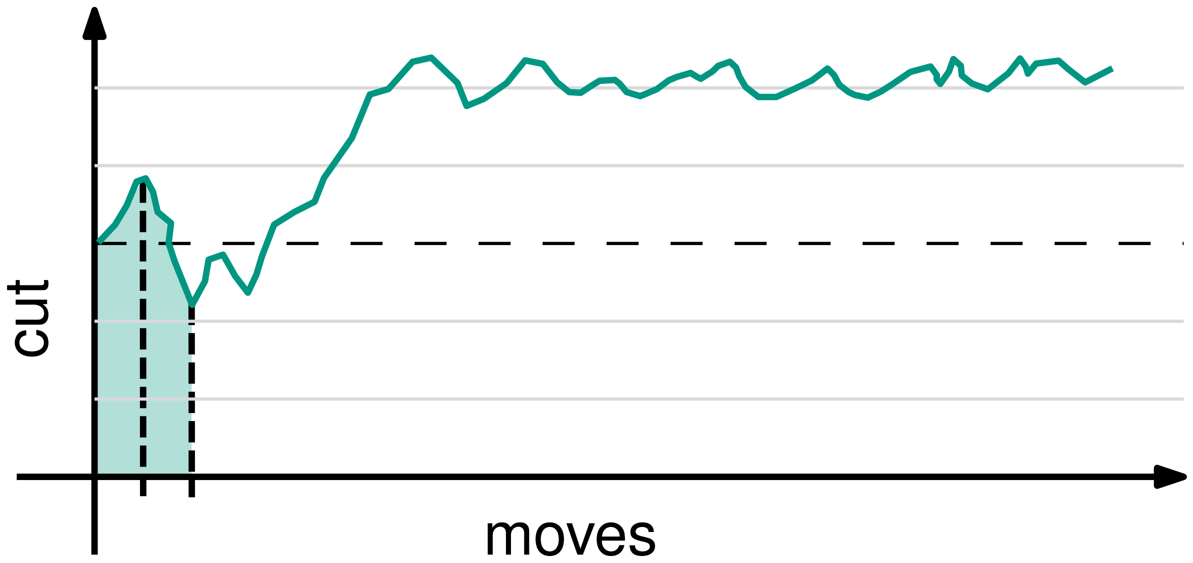
Idea: stop local search if improvement becomes **unlikely** [KaSPar]



Institute of Theoretical Informatics
Algorithmics Group

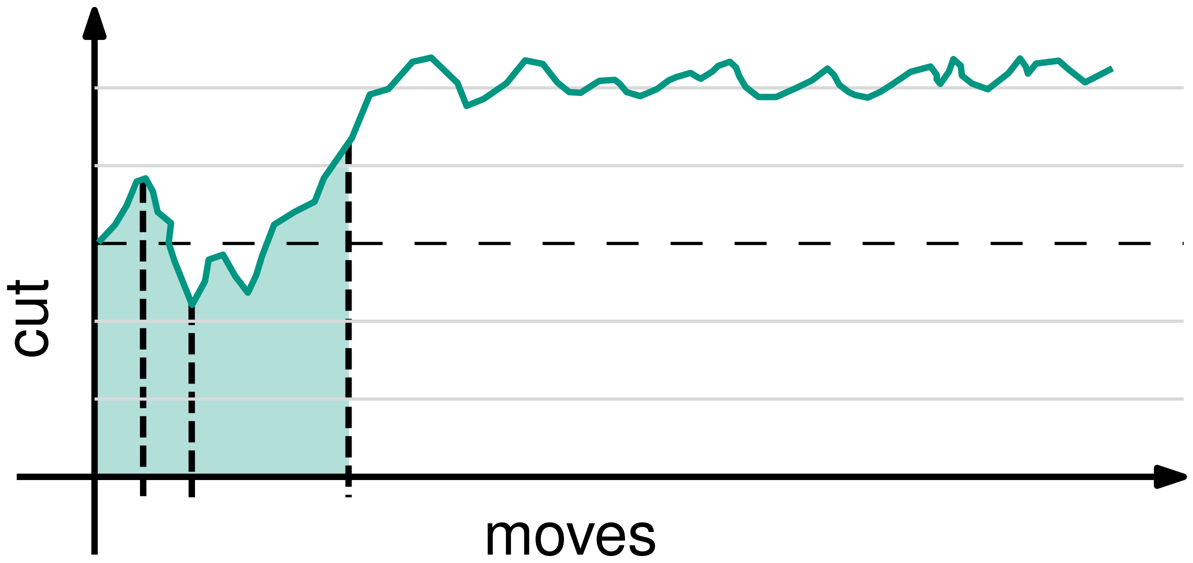
Adaptive Stopping Rule

Idea: stop local search if improvement becomes **unlikely** [KaSPar]



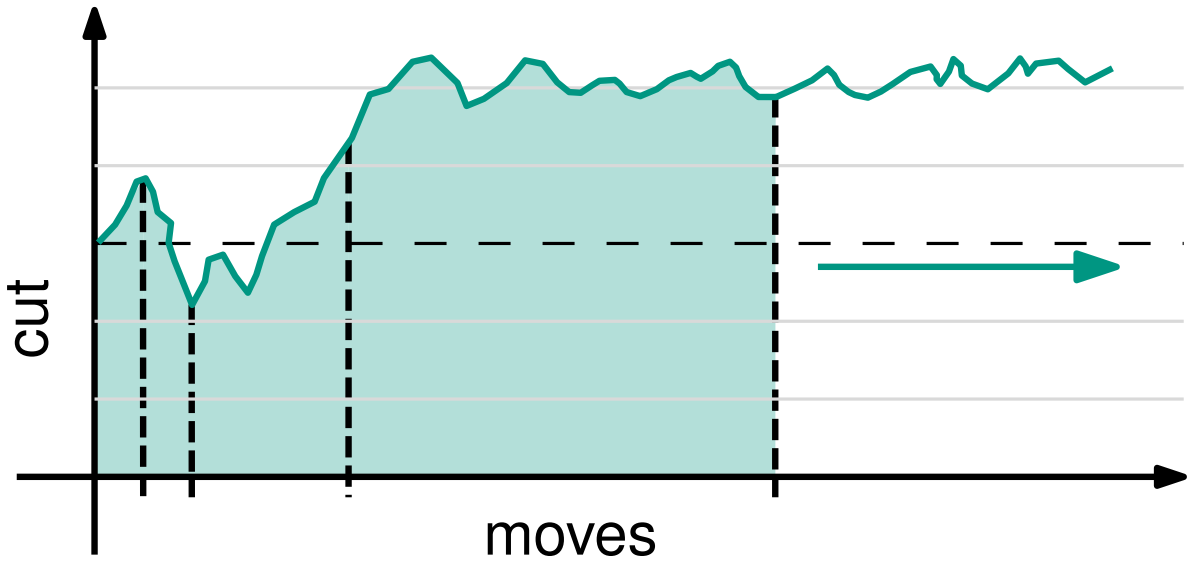
Adaptive Stopping Rule

Idea: stop local search if improvement becomes **unlikely** [KaSPar]



Adaptive Stopping Rule

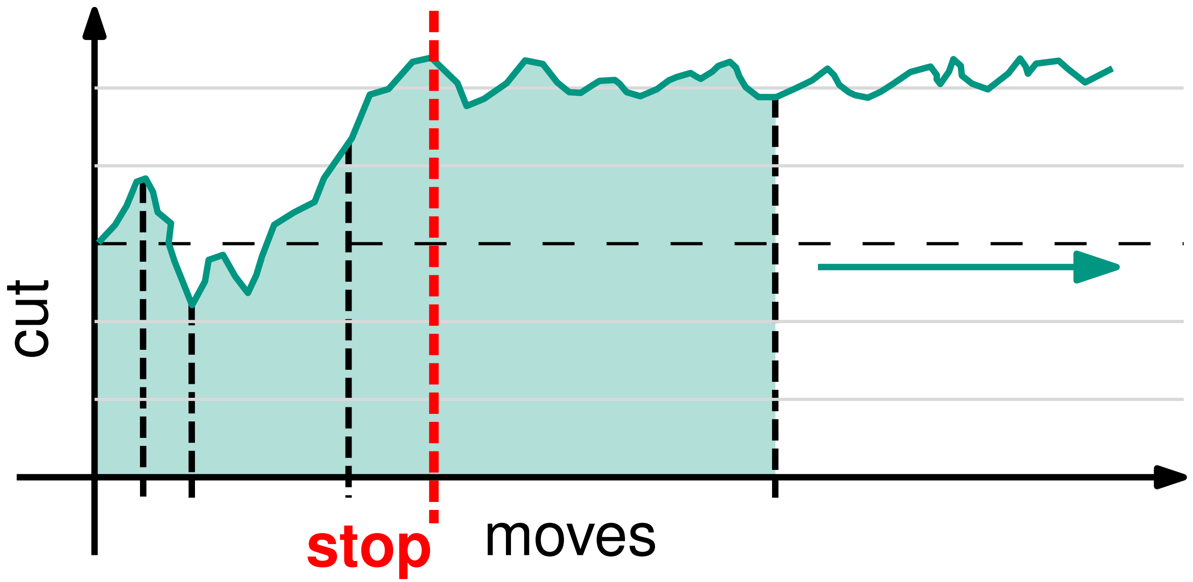
Idea: stop local search if improvement becomes **unlikely** [KaSPar]



Institute of Theoretical Informatics
Algorithmics Group

Adaptive Stopping Rule

Idea: stop local search if improvement becomes **unlikely** [KaSPar]



$$p > \frac{\sigma^2}{4\mu^2}$$

- p: # moves since last improvement
- σ^2: observed variance
- μ: avg. gain since last improvement

Institute of Theoretical Informatics
Algorithmics Group

Experiments – Benchmark Setup

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

- # Hypergraphs: [publicly available]
  - UF Sparse Matrix Collection 184
  - SAT Competition 2014 Application Track 92
  - ISPD98 VLSI Circuit Benchmark Suite 18

- $k \in \{2, 4, 8, 16, 32, 64, 128\}$ → **2058 instances**
- imbalance: $\varepsilon = 3\%$
- 8 hours time limit / instance

- Comparison with:
  - hMetis-R & hMetis-K
  - PaToH-Default & PaToH-Quality

Institute of Theoretical Informatics
Algorithmics Group

Effects of Engineering Efforts

Subset of all Instances

	cut	t_c[s]	t_{ls}[s]
Baseline	6506	1.84	56.87
+ New Coarsening	6509	0.50	*
+ Gain Caching	6505	*	31.20
+ stop early	6537	*	3.48
+ exclude nets	6537	*	3.06

Effects of Engineering Efforts

Subset of all Instances

	cut	t_c[s]	t_{ls}[s]
Baseline	6506	1.84	56.87
+ New Coarsening	6509	0.50	*
+ Gain Caching	6505	*	31.20
+ stop early	6537	*	3.48
+ exclude nets	6537	*	3.06

Effects of Engineering Efforts

Karlsruhe Institute of Technology

Subset of all Instances

	cut	t_c[s]	t_{ls}[s]
Baseline	6506	1.84	56.87
+ New Coarsening	6509	0.50	*
+ Gain Caching	6505	*	31.20
+ stop early	6537	*	3.48
+ exclude nets	6537	*	3.06

Effects of Engineering Efforts

Subset of all Instances

	cut	t_c[s]	t_{ls}[s]
Baseline	6506	1.84	56.87
+ New Coarsening	6509	0.50	*
+ Gain Caching	6505	*	31.20
+ stop early	6537	*	3.48
+ exclude nets	6537	*	3.06

Effects of Engineering Efforts

Subset of all Instances

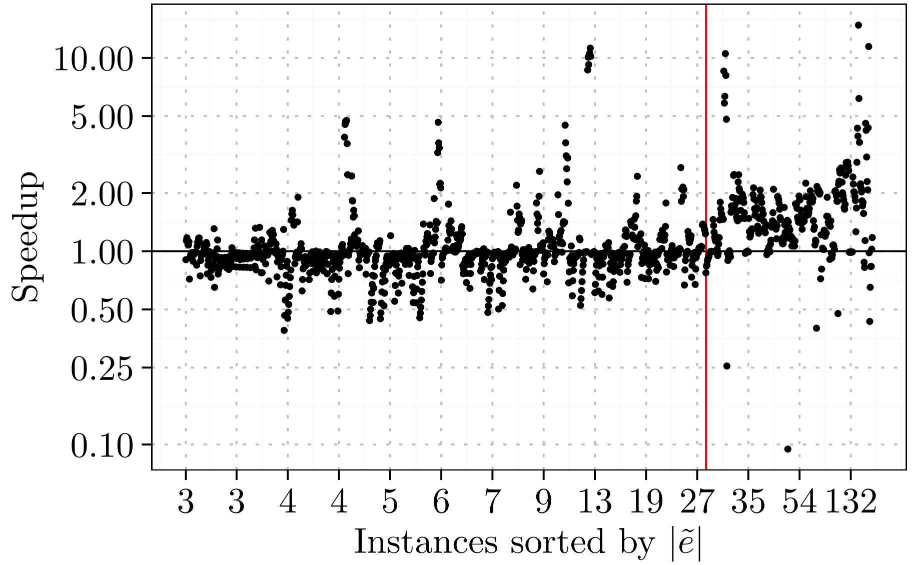
	cut	t_c[s]	t_{ls}[s]
Baseline	6506	1.84	56.87
+ New Coarsening	6509	0.50	*
+ Gain Caching	6505	*	31.20
+ stop early	6537	*	3.48
+ exclude nets	6537	*	3.06

Effects of Engineering Efforts

Subset of all Instances

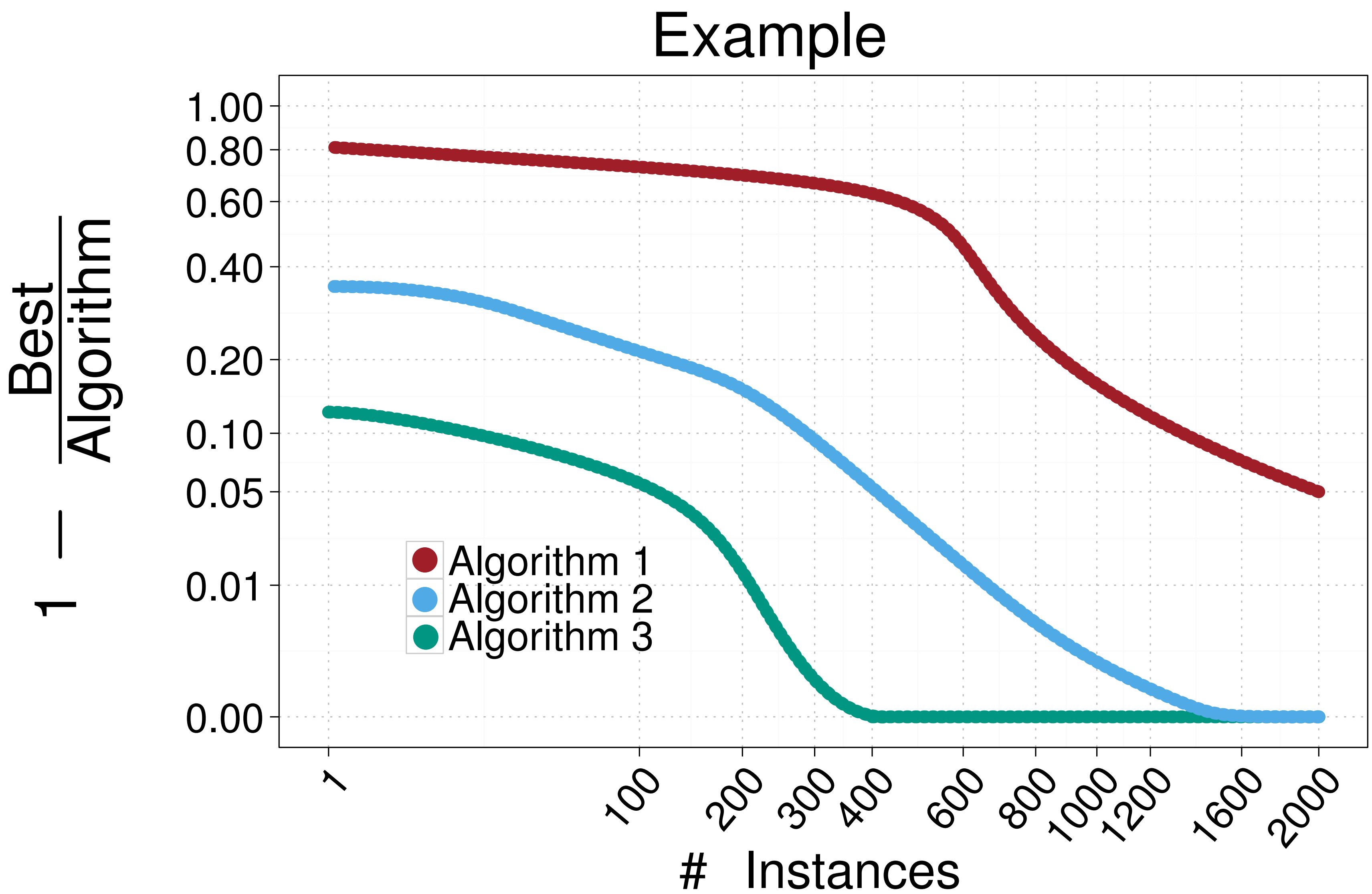
	cut	t_c[s]	t_{ls}[s]
Baseline	6506	1.84	56.87
+ New Coarsening	6509	0.50	*
+ Gain Caching	6505	*	31.20
+ stop early	6537	*	3.48
+ exclude nets	6537	*	3.06

Min-Hash Sparsifier



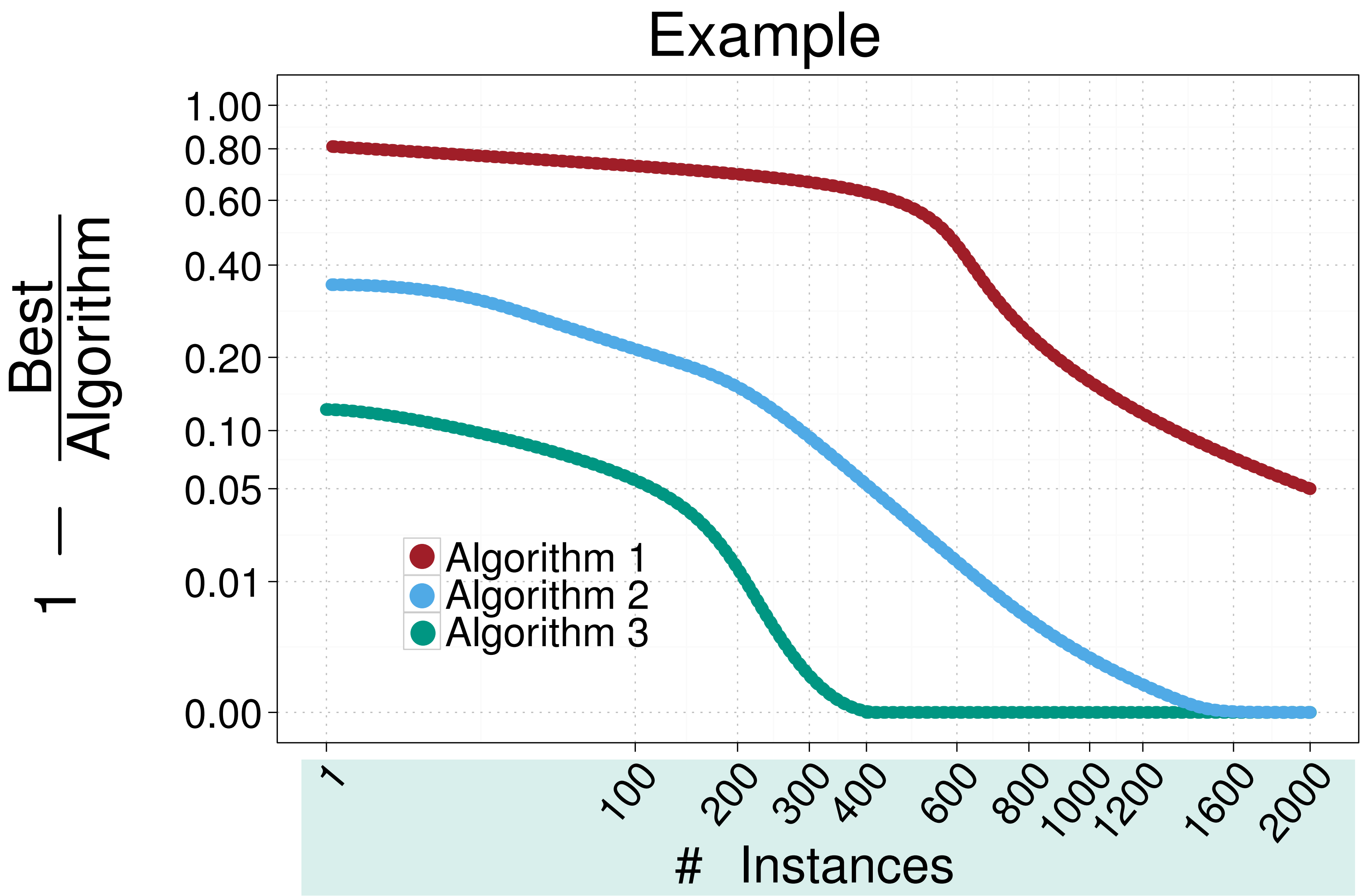
Institute of Theoretical Informatics
Algorithmics Group

Experimental Results – Partitioning Quality



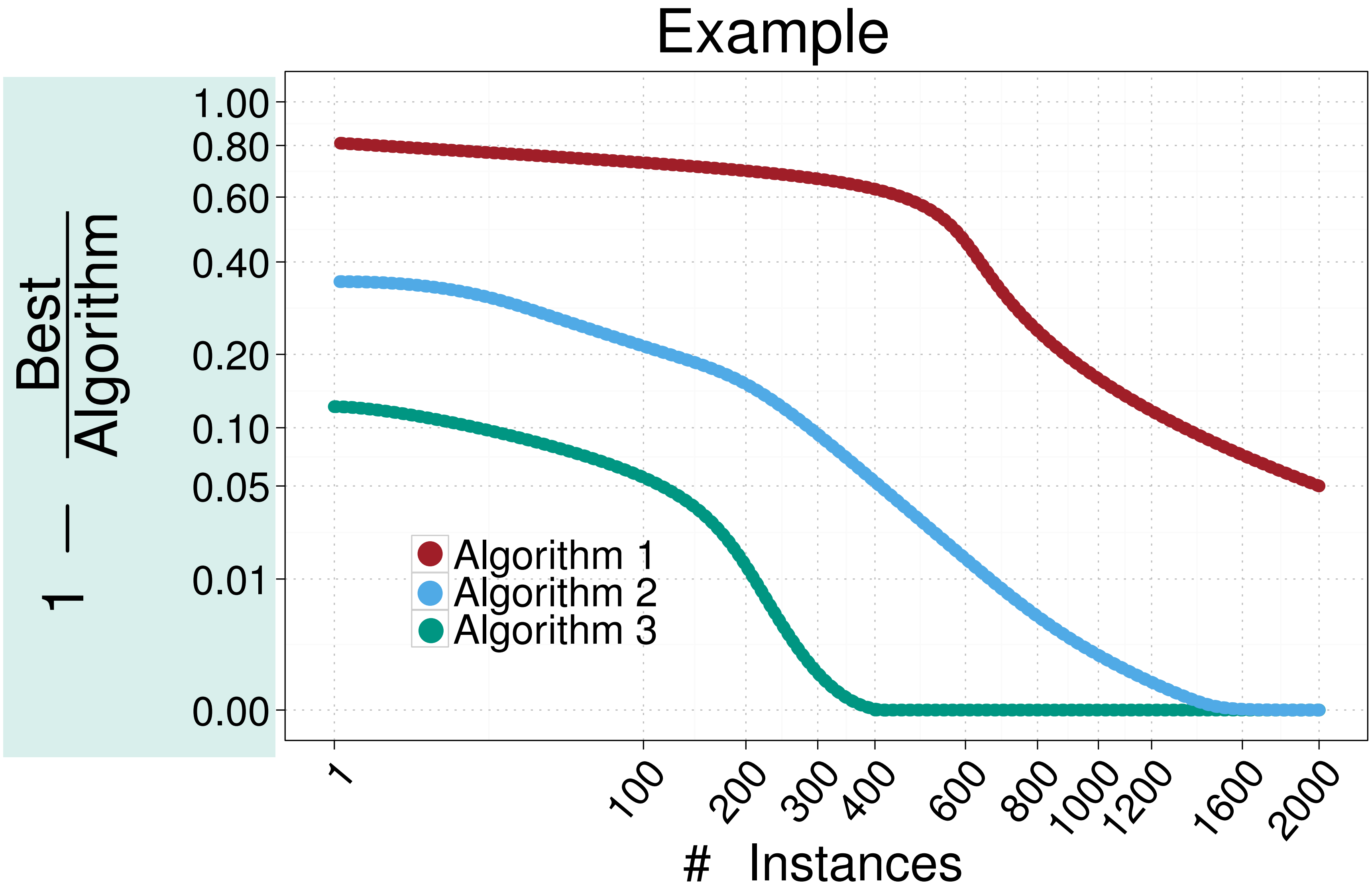
Institute of Theoretical Informatics
Algorithmics Group

Experimental Results – Partitioning Quality



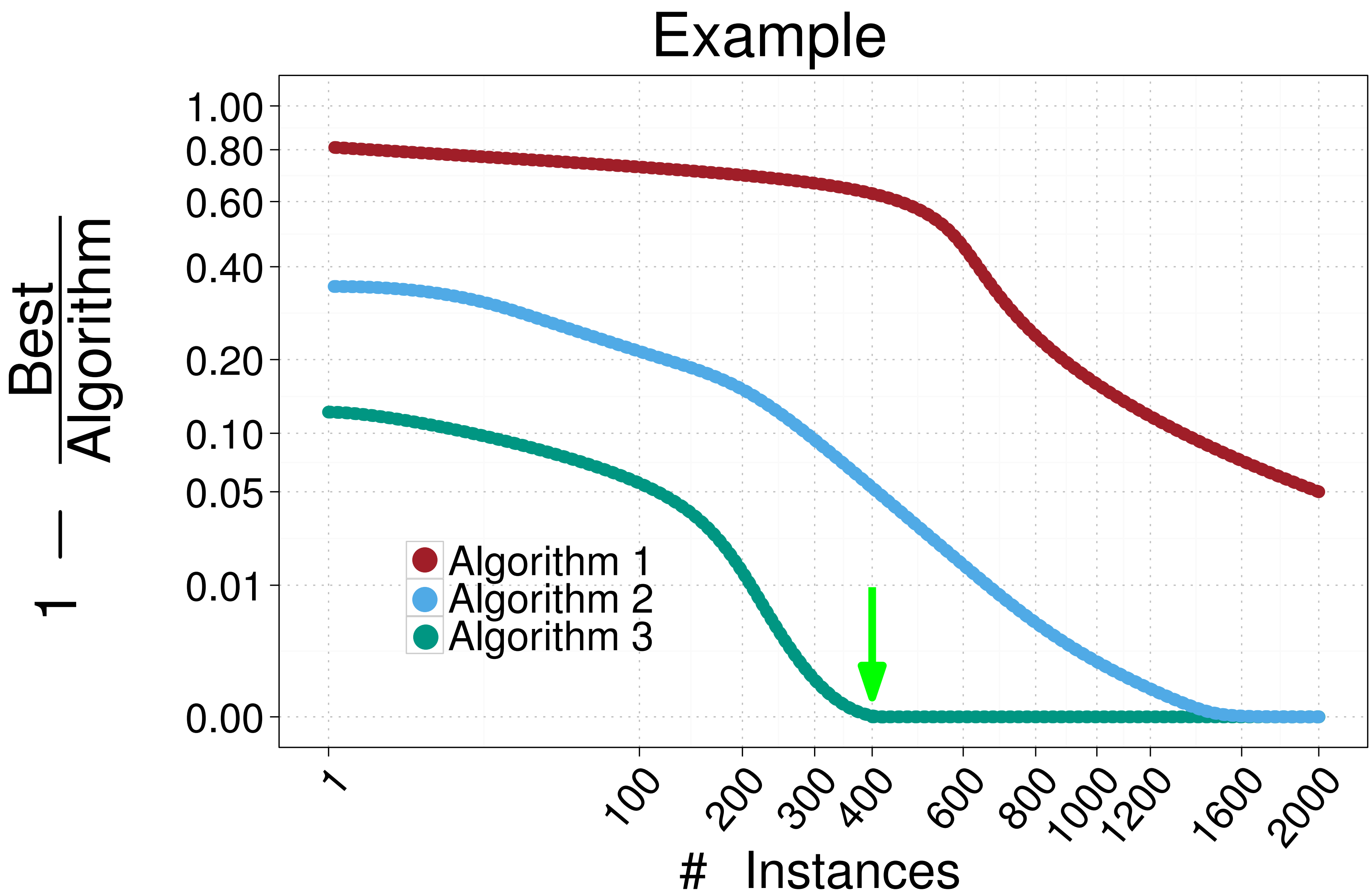
Institute of Theoretical Informatics
Algorithmics Group

Experimental Results – Partitioning Quality

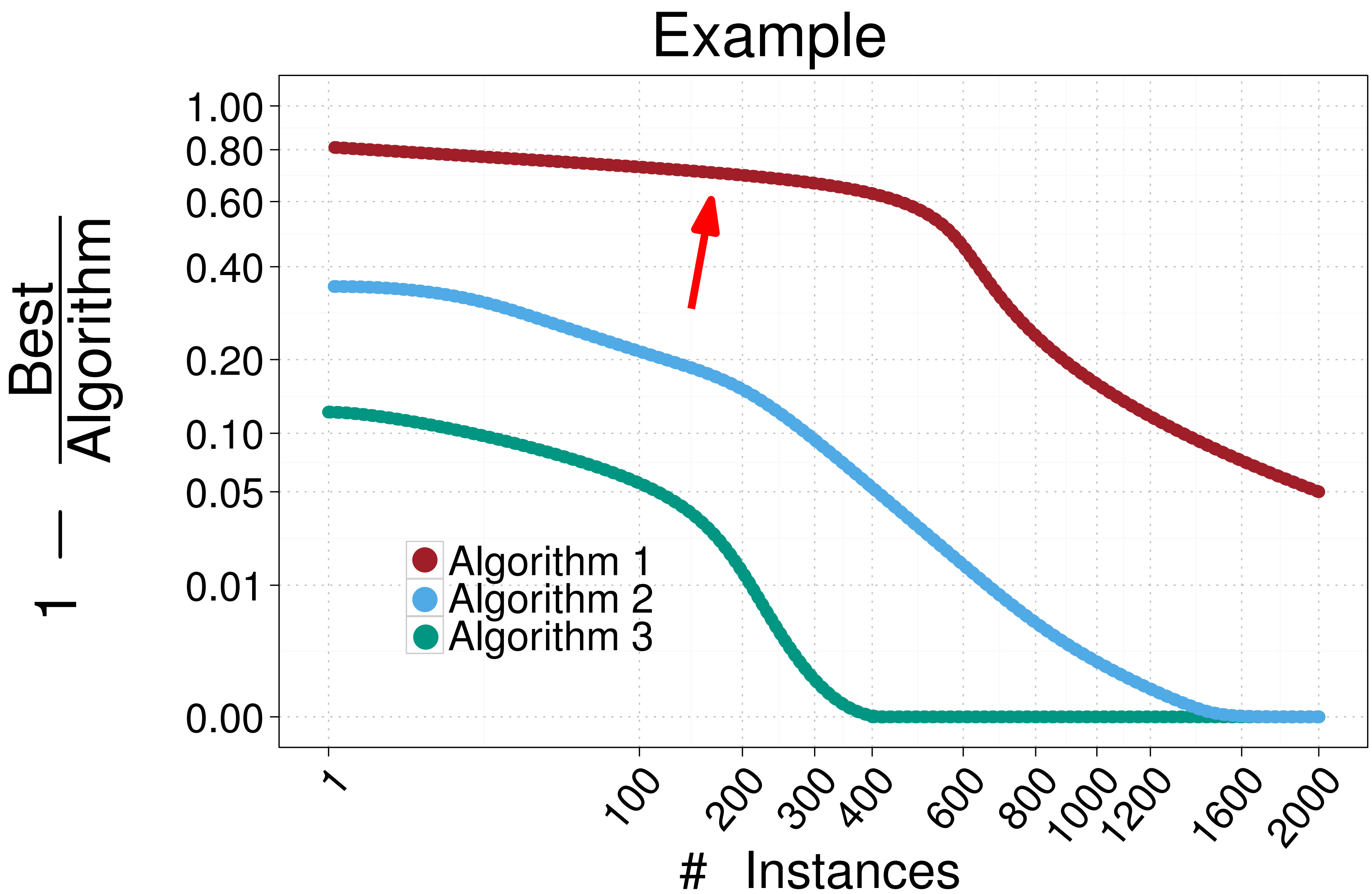


Institute of Theoretical Informatics
Algorithmics Group

Experimental Results – Partitioning Quality



Experimental Results – Partitioning Quality

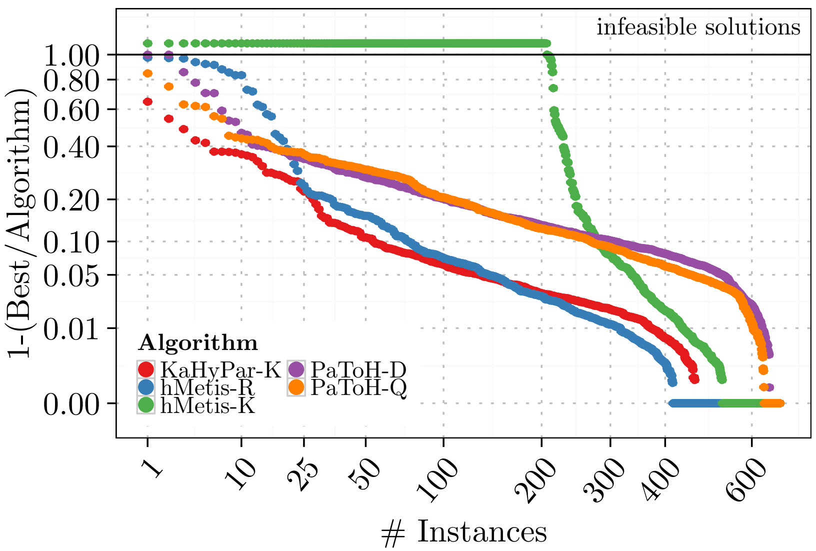


Institute of Theoretical Informatics
Algorithmics Group

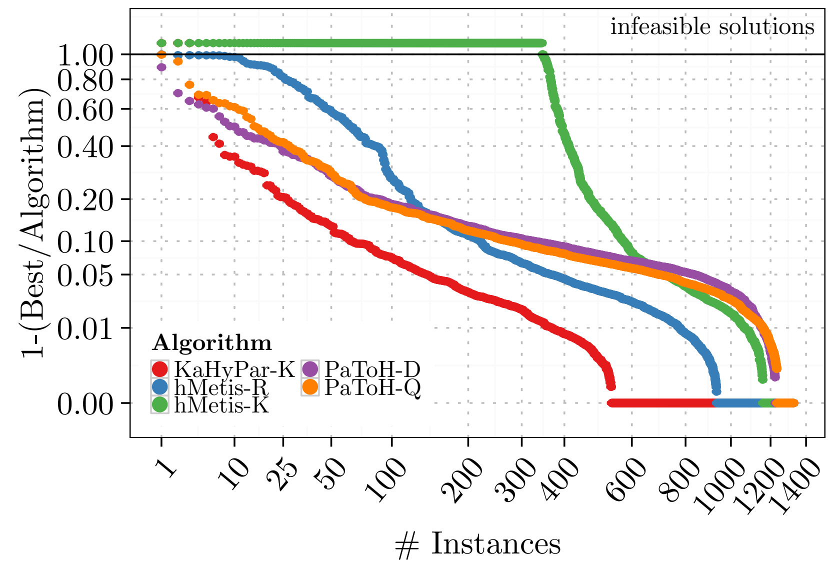
Experimental Results

Karlsruhe Institute of Technology

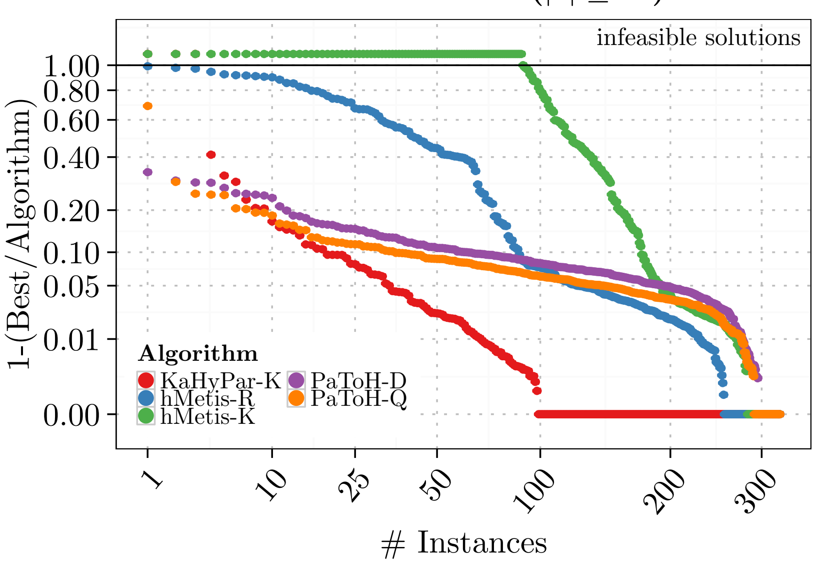
Min Cut Ratios ($|\tilde{e}| < 3$)



Min Cut Ratios ($|\tilde{e}| \geq 3$)



Min Cut Ratios ($|\tilde{e}| \geq 28$)



Running Time [s]

Algorithm	$\|\tilde{e}\| \geq 3$	$\|\tilde{e}\| < 3$	$\|\tilde{e}\| \geq 28$
KaHyPar-K	10.9	26.7	13.3
hMetis-R	45.1	103.6	90.0
hMetis-K	37.2	75.3	92.6
PaToH-Q	3.8	6.3	10.4
PaToH-D	0.8	1.1	3.1

Conclusion & Discussion

KaHyPar-K – direct k-way HGP optimizing $(\lambda - 1)$ metric

- min-hash based pin sparsifier
- fast n-level coarsening
- engineered FM-based local search

KaHyPar-Framework
Open-Source on Github:
`https://git.io/vMBaR`

In the paper:

- adaptive fingerprint construction
- fast n-level coarsening
- more experiments:
  - Comparison with KaHyPar-R
  - $k \in \{5, 23, 47, 107\}$