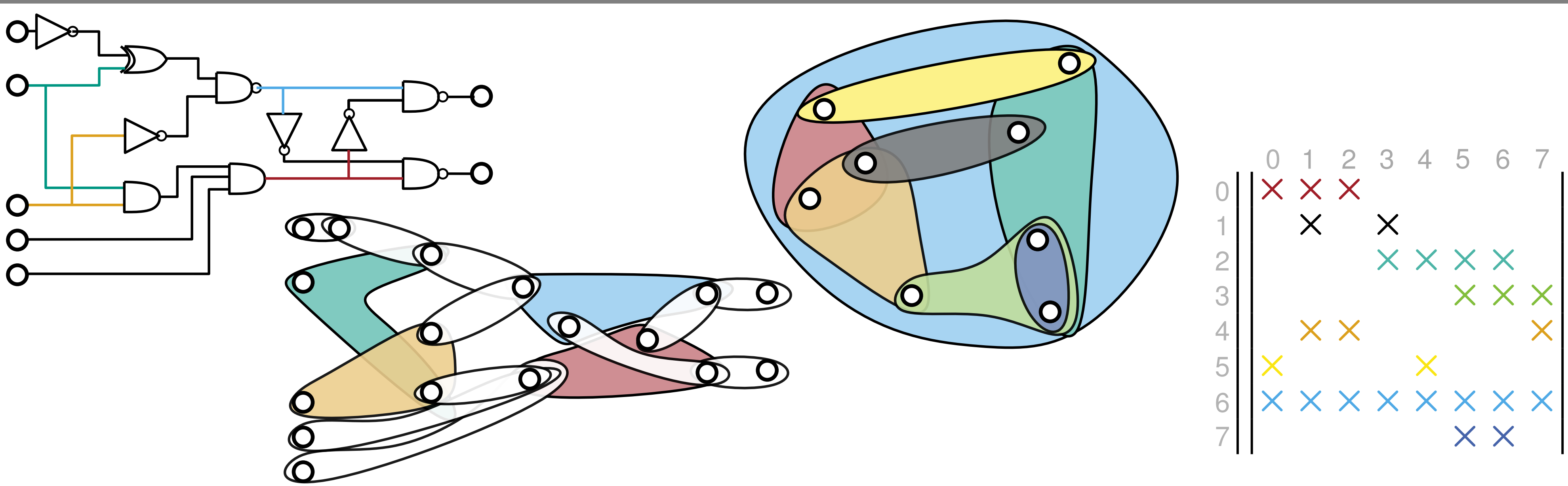
Karlsruhe Institute of Technology

k-way Hypergraph Partitioning via *n*-Level Recursive Bisection

Sebastian Schlag, Vitali Henne, Tobias Heuer, Henning Meyerhenke
Peter Sanders, Christian Schulz

January 10th, 2016 @ ALENEX'16

INSTITUTE OF THEORETICAL INFORMATICS · ALGORITHMICS GROUP

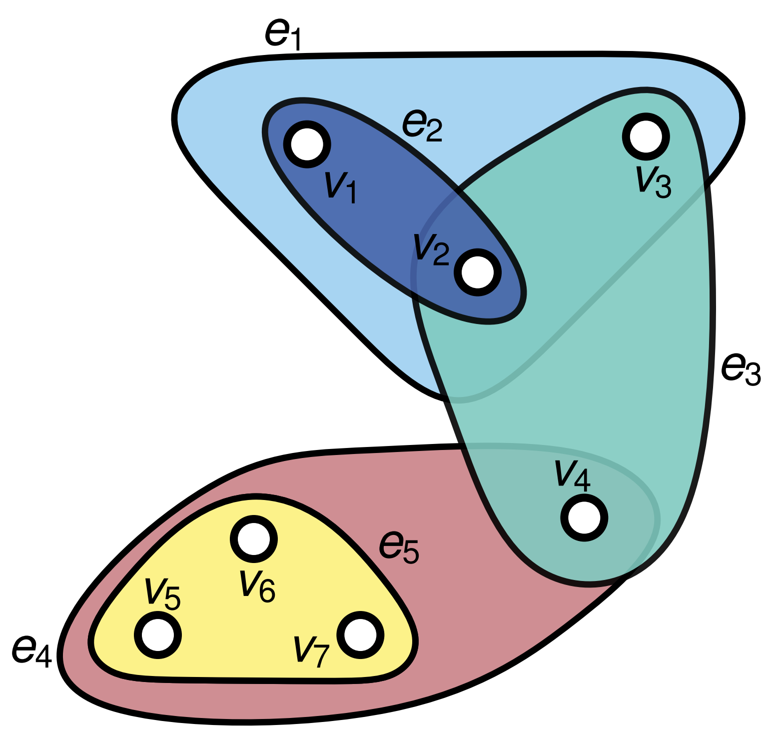


Hypergraphs

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

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

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



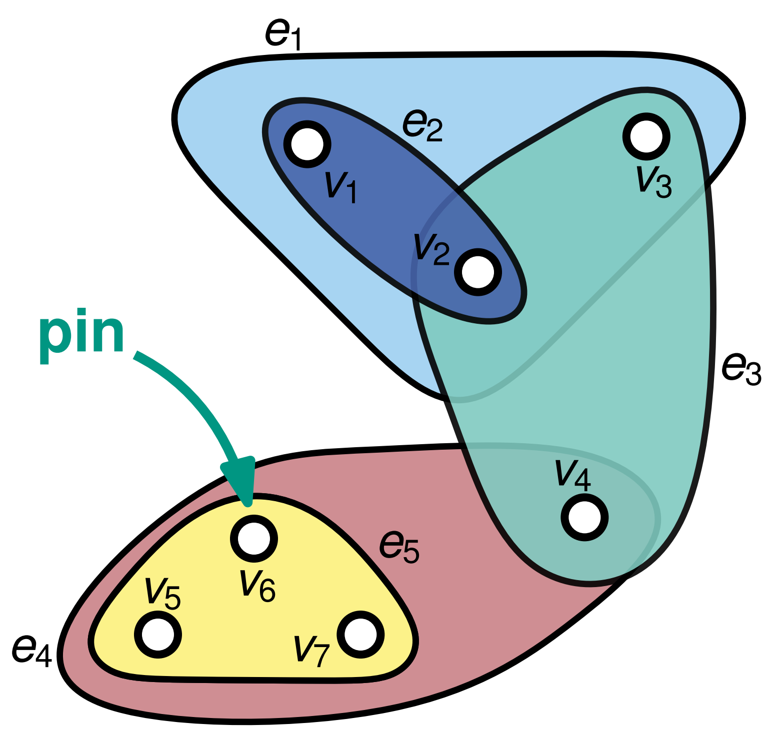
Institute of Theoretical Informatics
Algorithmics Group

Hypergraphs

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

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

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



Institute of Theoretical Informatics
Algorithmics Group

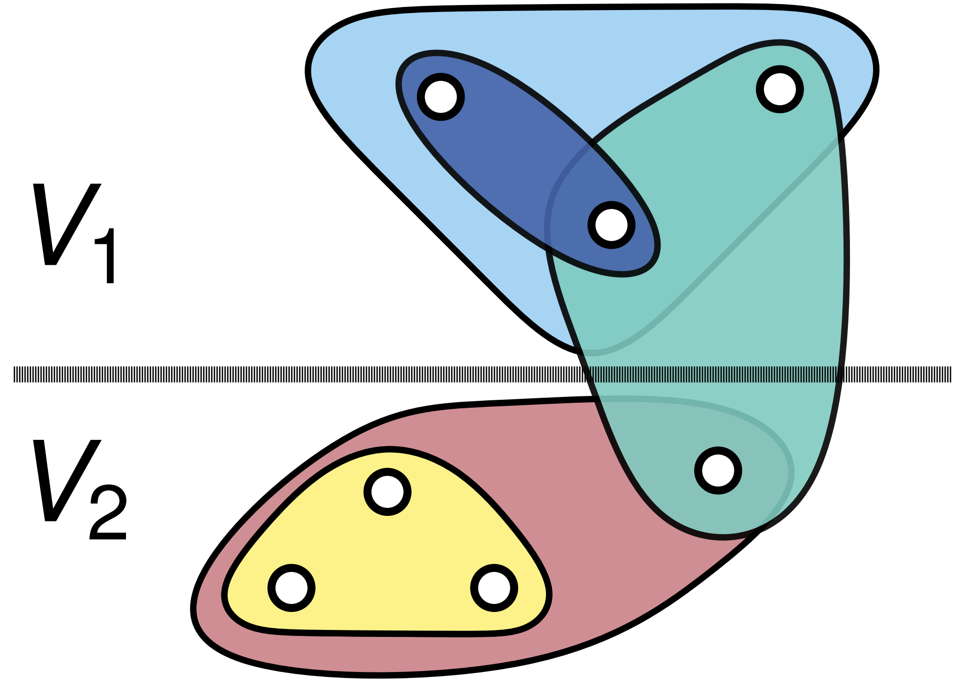
Hypergraph Partitioning Problem

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**:

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



Institute of Theoretical Informatics
Algorithmics Group

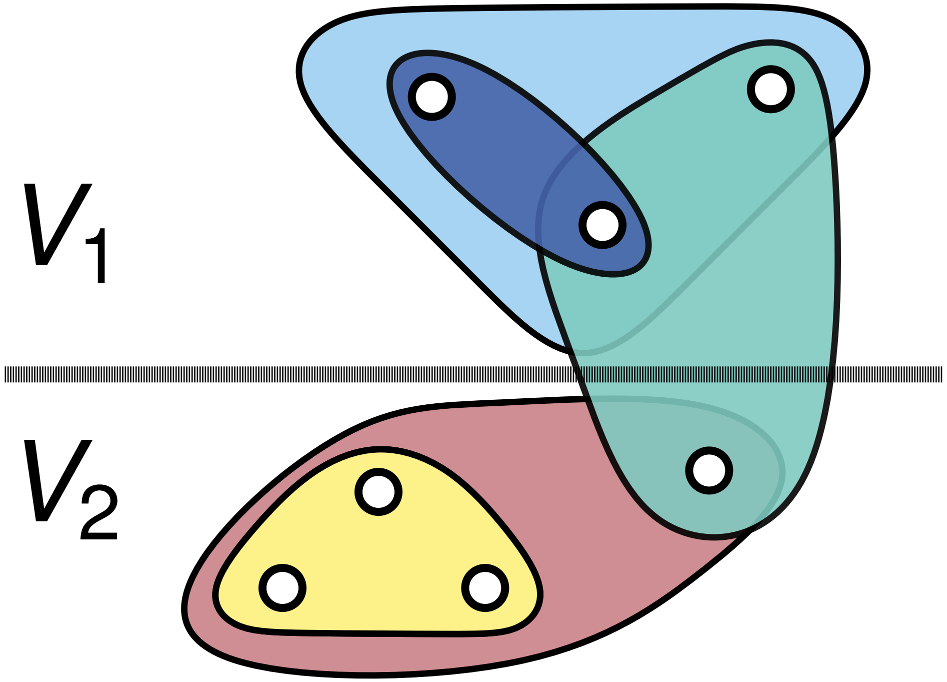
Hypergraph Partitioning Problem

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**:

imbalance parameter

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



Hypergraph Partitioning Problem

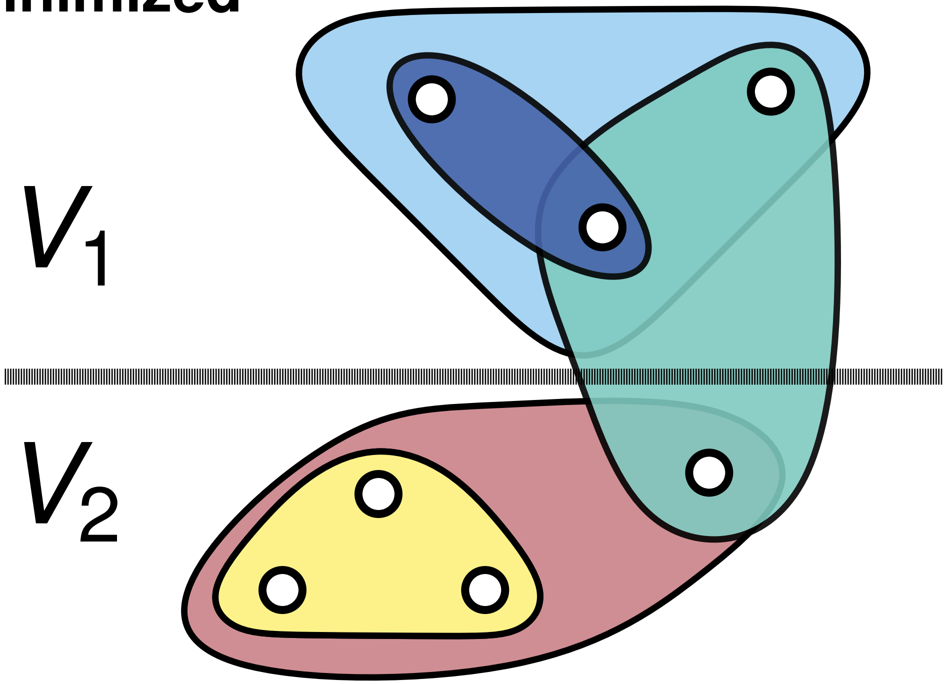
Partition hypergraph $H = (V, E, c, \omega)$ into $\mathbf{k}$ disjoint blocks $\Pi = \{V_1, \ldots, V_k\}$ such that:

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

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

imbalance parameter

- total weight of **cut** hyperedges is **minimized**



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:

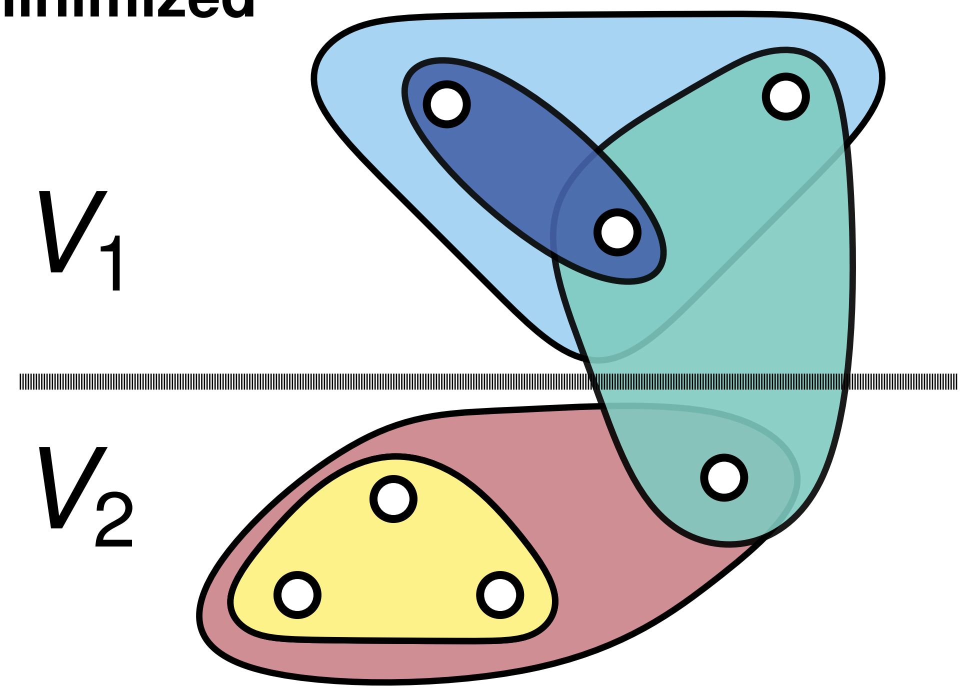
- blocks V_i are **roughly equal-sized**:

imbalance parameter

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

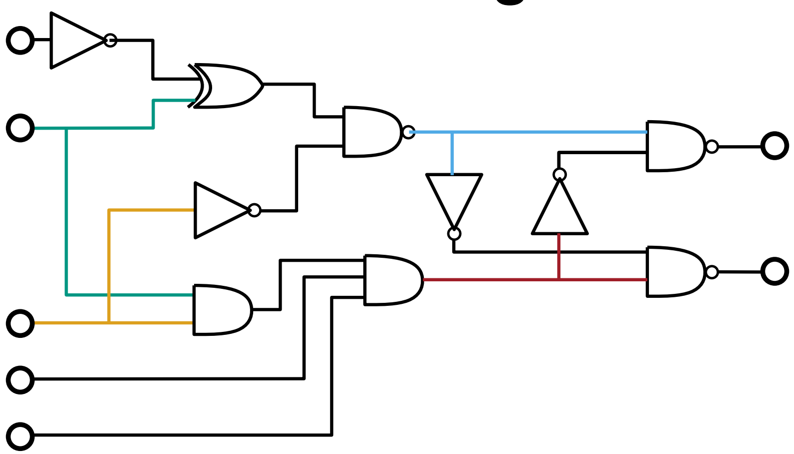
- total weight of **cut** hyperedges is **minimized**

hyperedge connecting **multiple** blocks



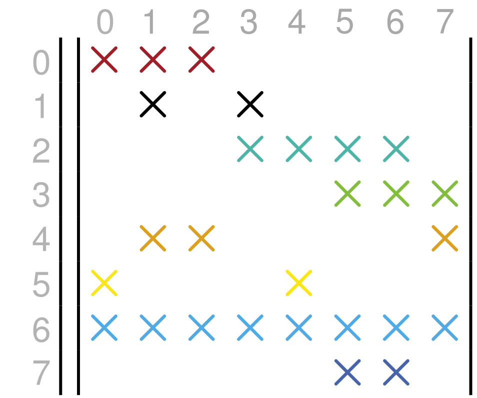
Applications

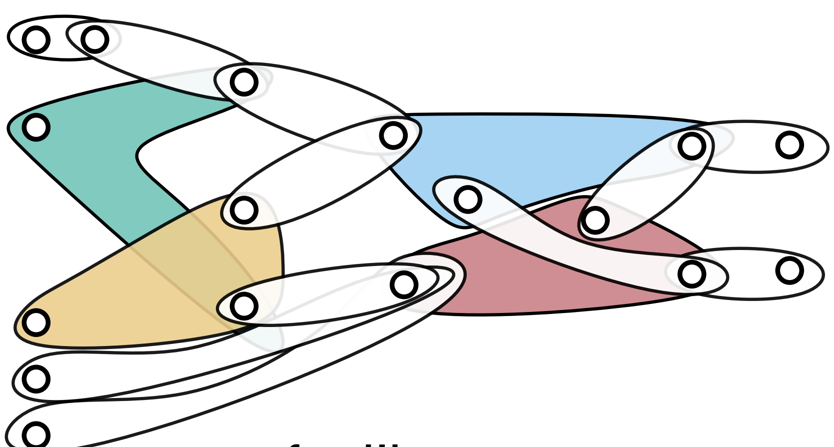
VLSI Design



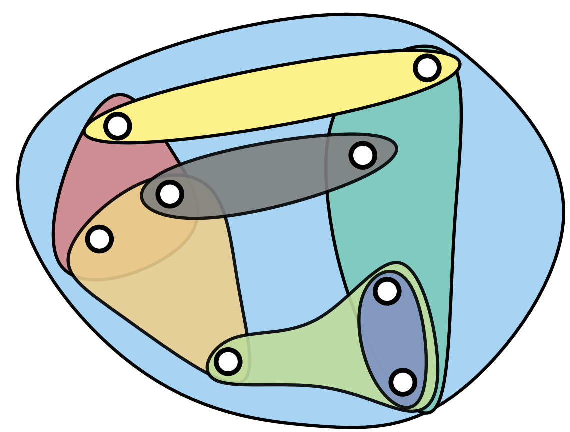
Application Domain

Scientific Computing





Hypergraph Model

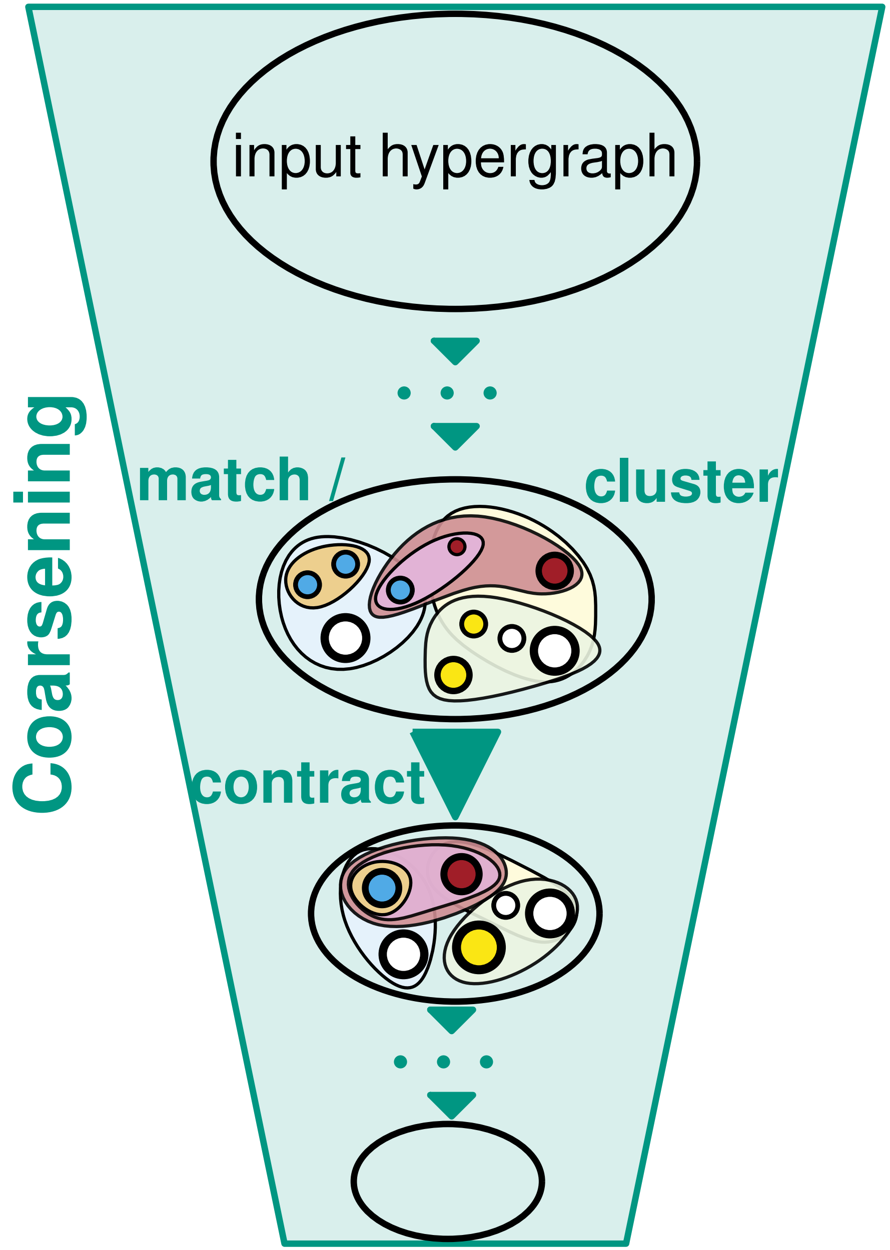


facilitate floorplanning & placement

Goal

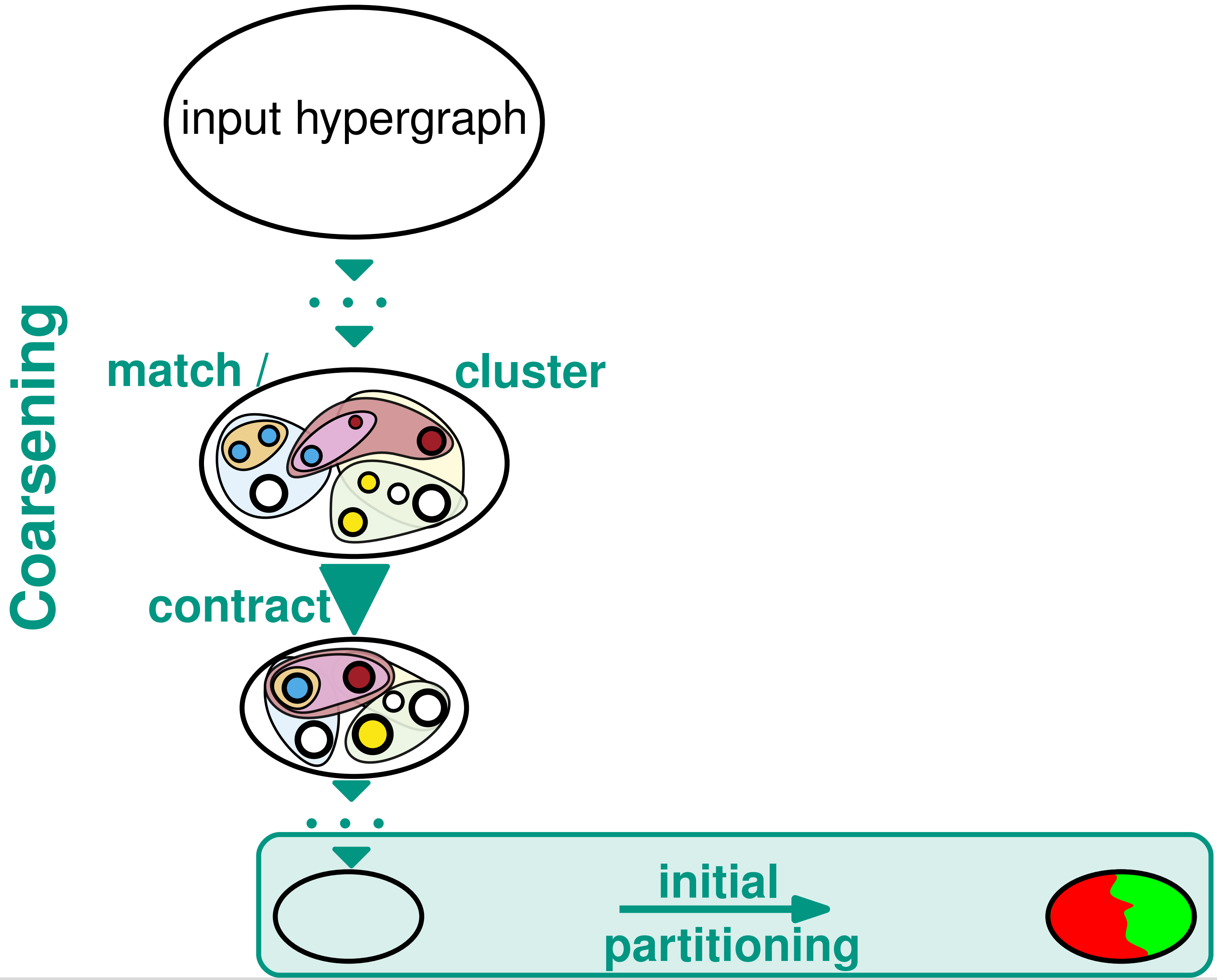
minimize communication

Multilevel Paradigm



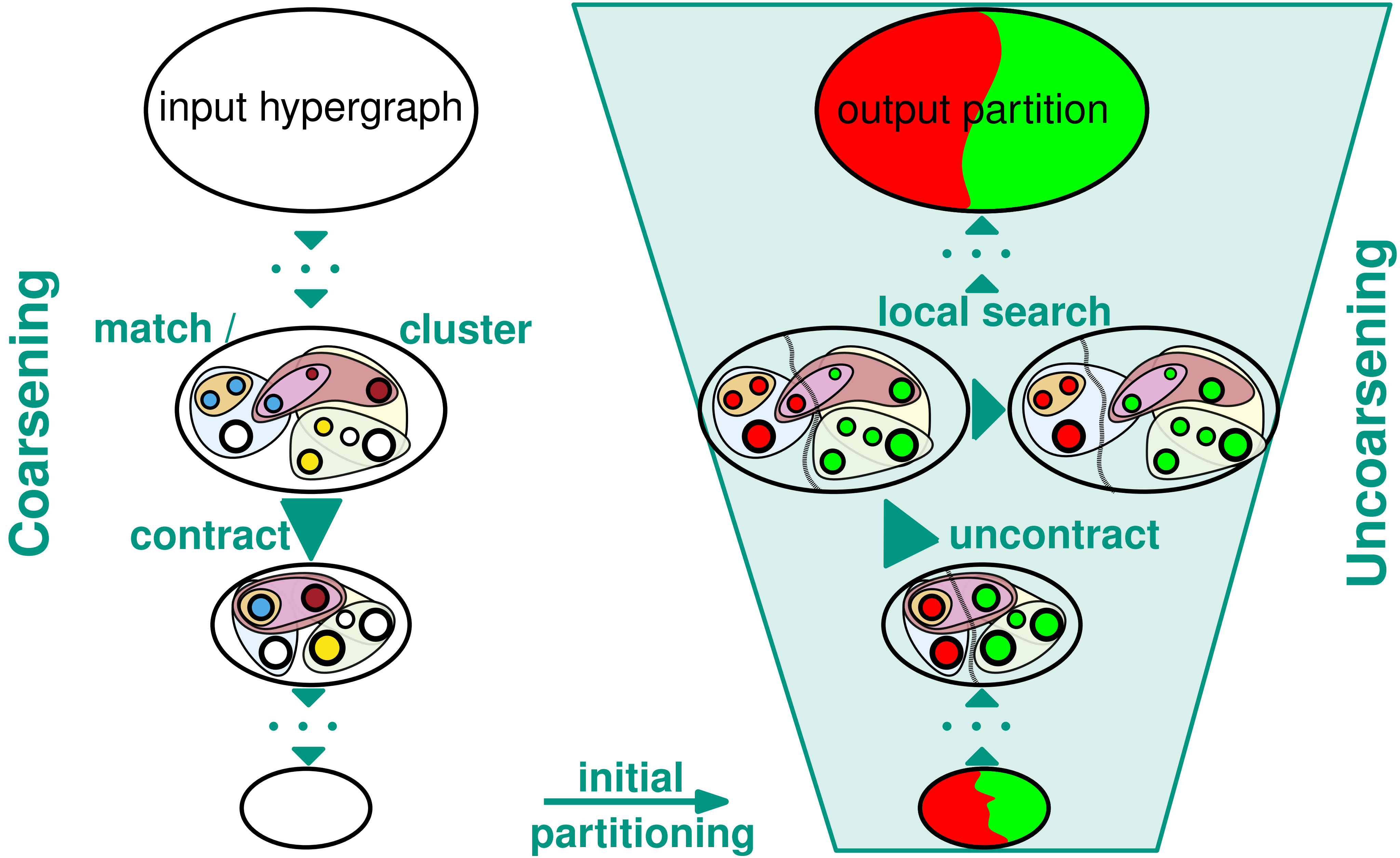
Institute of Theoretical Informatics
Algorithmics Group

Multilevel Paradigm

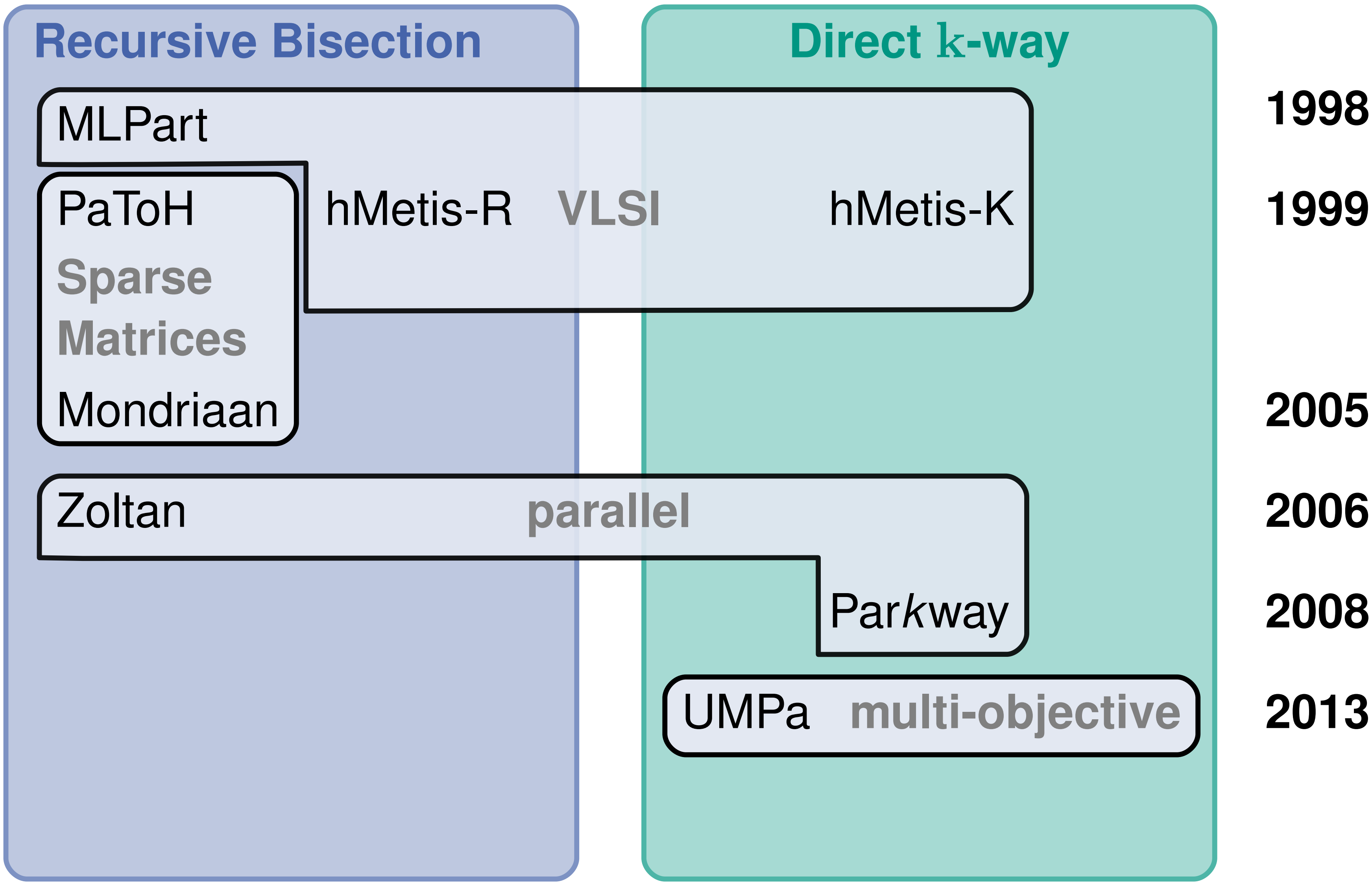


Institute of Theoretical Informatics
Algorithmics Group

Multilevel Paradigm



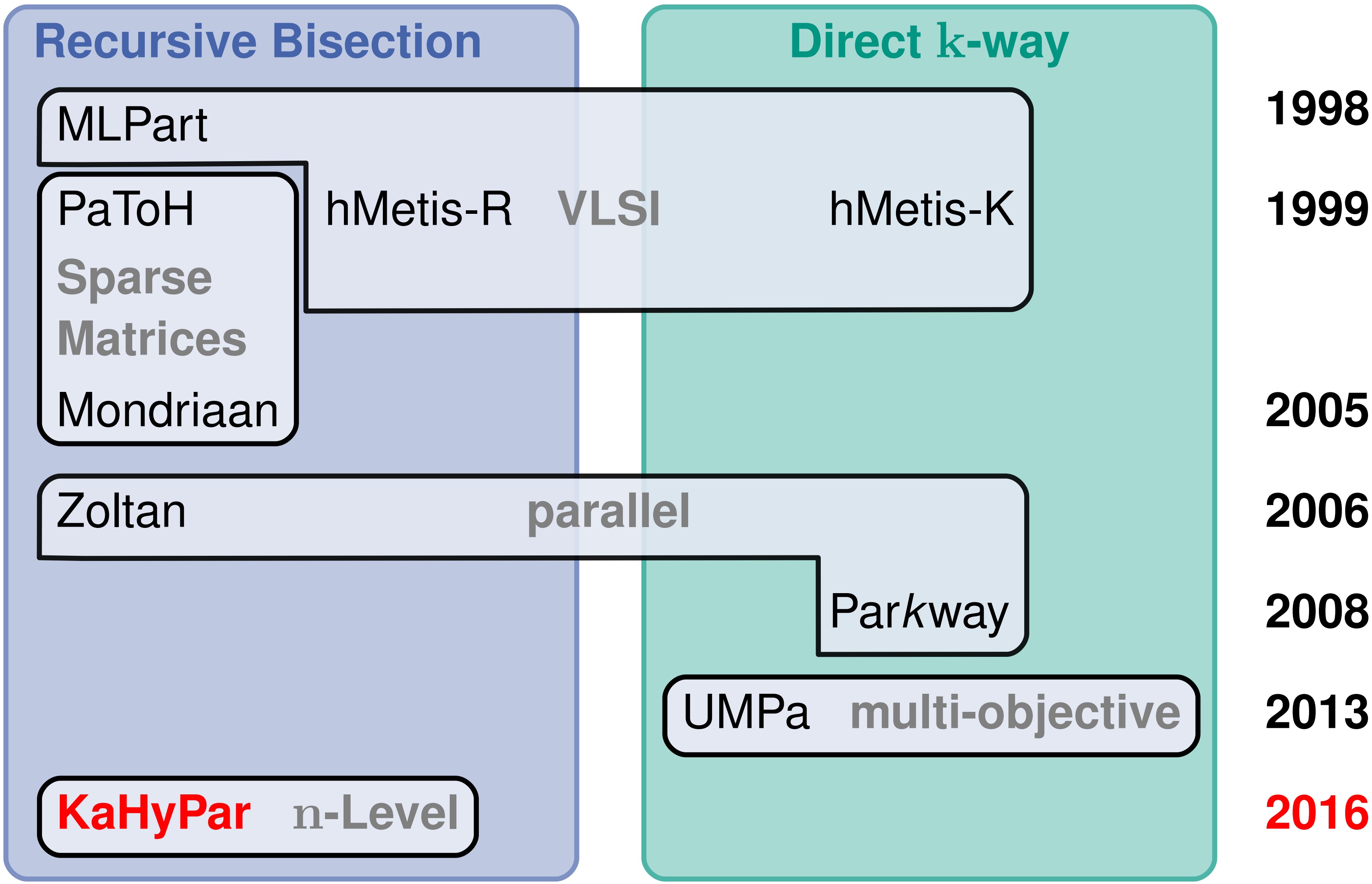
Taxonomy of Hypergraph Partitioning Tools



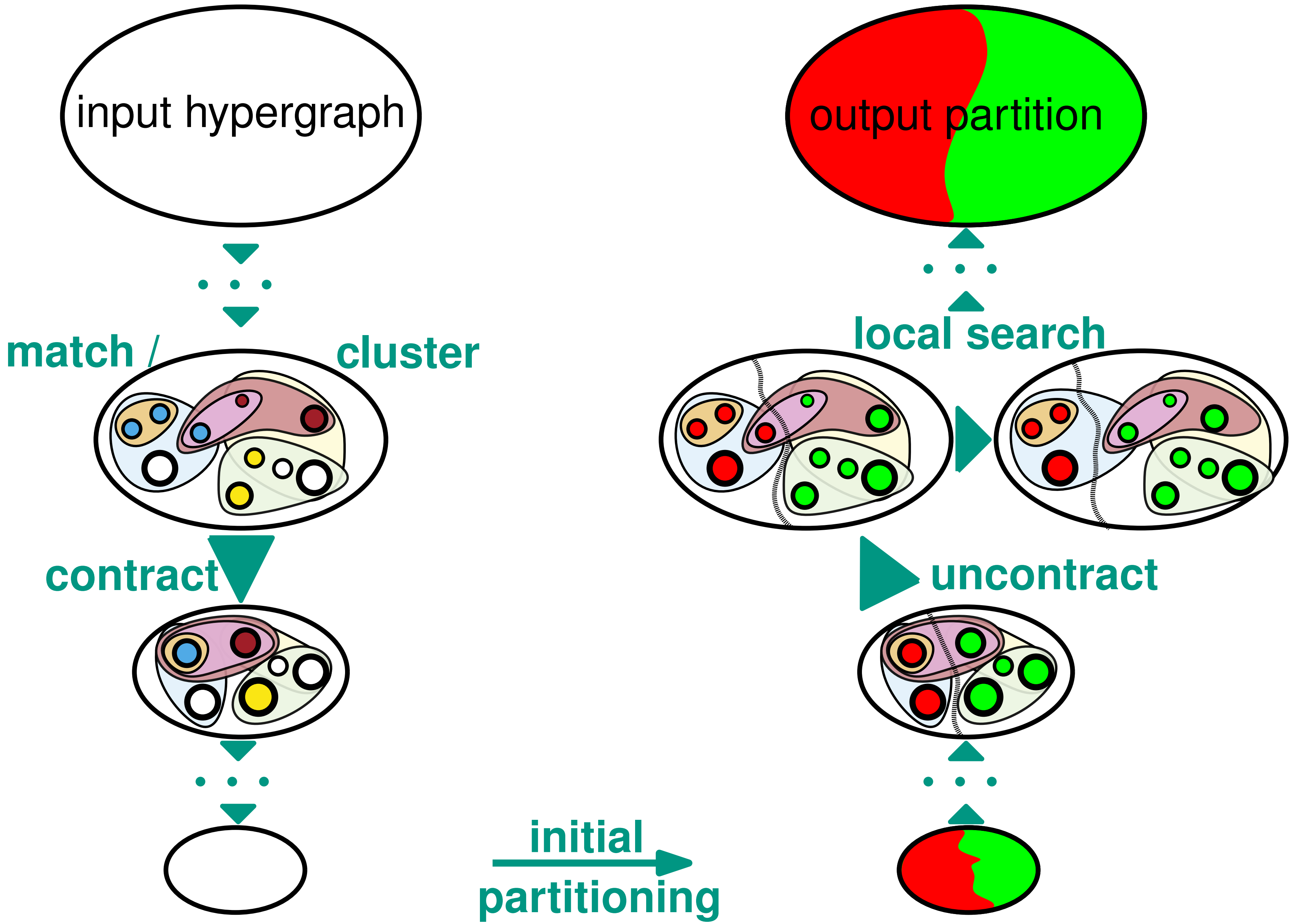
Institute of Theoretical Informatics
Algorithmics Group

Taxonomy of Hypergraph Partitioning Tools

Karlsruhe Institute of Technology

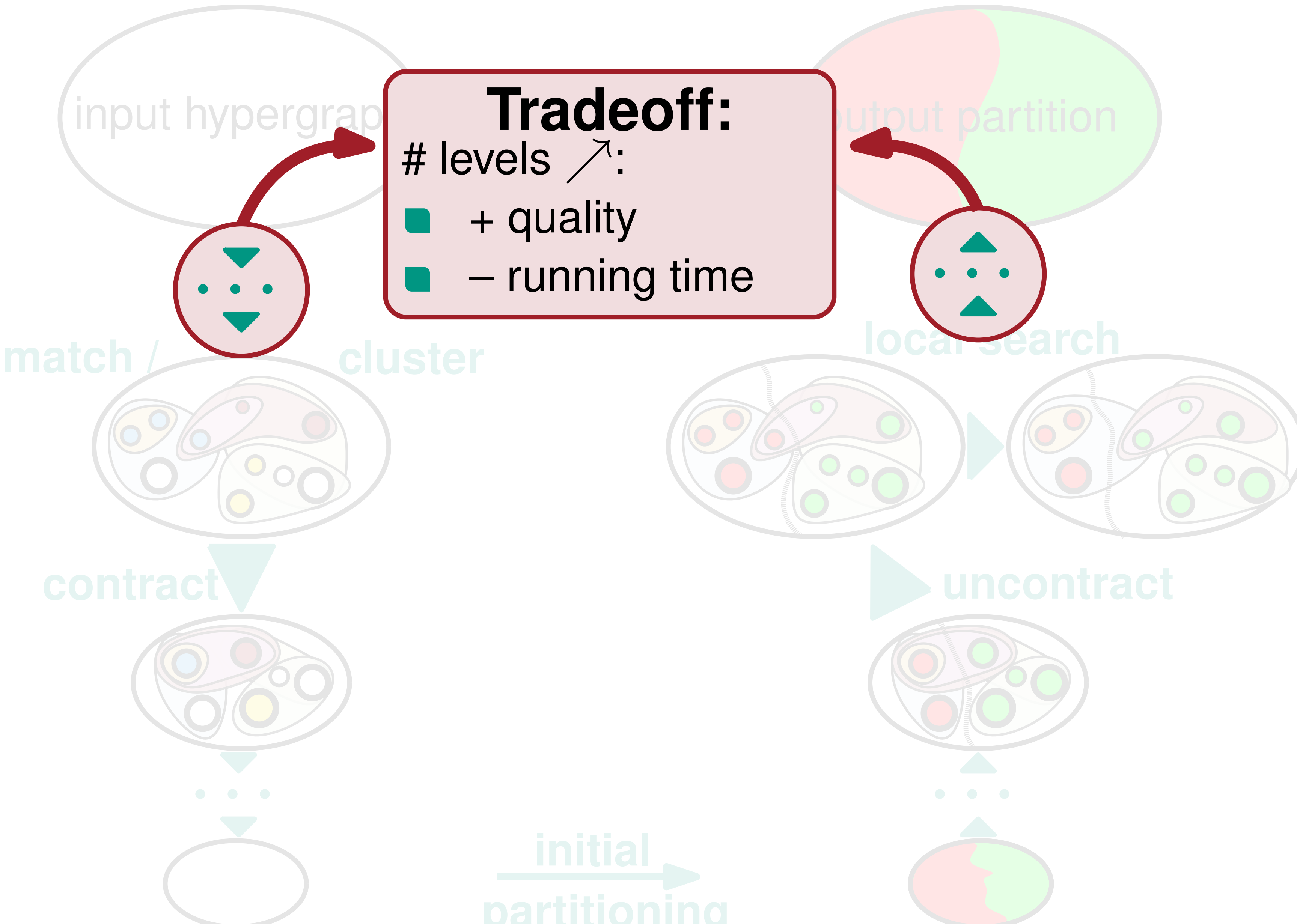


Why Yet Another Multilevel Algorithm?



Institute of Theoretical Informatics
Algorithmics Group

Why Yet Another Multilevel Algorithm?



Tradeoff:

\# levels ↗:

- \+ quality
- – running time

Institute of Theoretical Informatics
Algorithmics Group

Why Yet Another Multilevel Algorithm?

Tradeoff:

levels ↗:

- \+ quality
- – running time

Our Contribution:

evade tradeoff ⇝ n levels

⇒ combine high quality with good performance

Institute of Theoretical Informatics
Algorithmics Group

Why Yet Another Multilevel Algorithm?

Tradeoff:

\# levels ↗:

- \+ quality
- – running time

Our Contribution:

evade tradeoff ⇝ n levels

⇒ combine high quality with good performance

Motivation:

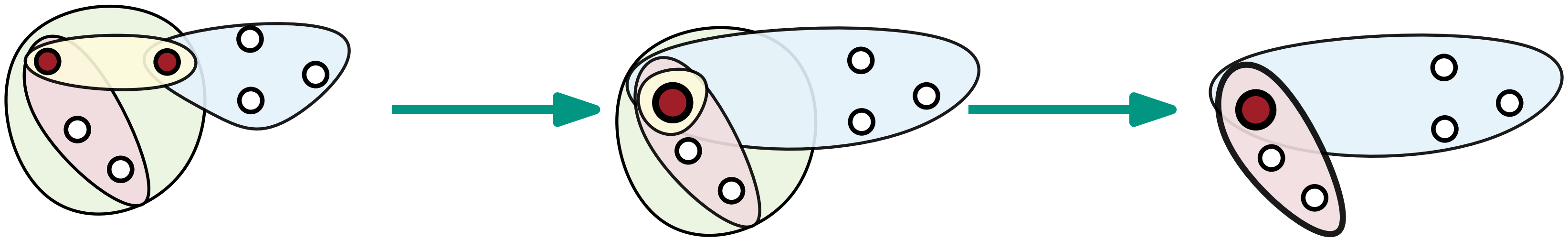
KaSPar – *n*-level graph partitioning

Coarsening

Institute of Theoretical Informatics
Algorithmics Group

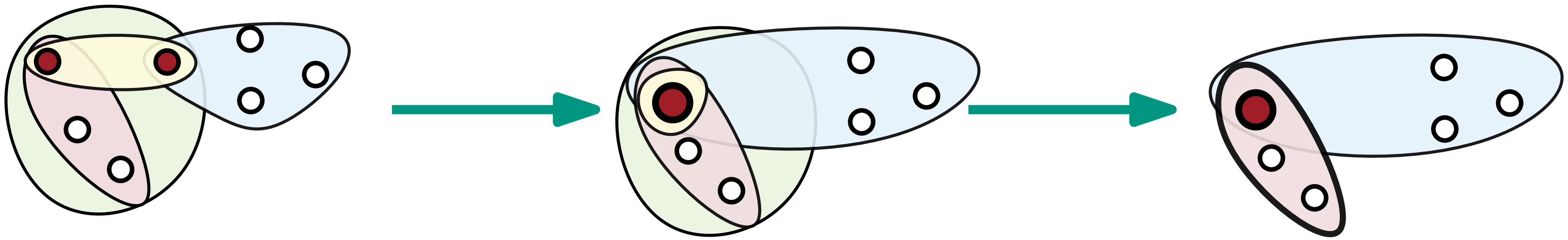
n-Level Coarsening Phase

- contract only **a single pair** of vertices at **each** level



n-Level Coarsening Phase

- contract only **a single pair** of vertices at **each** level



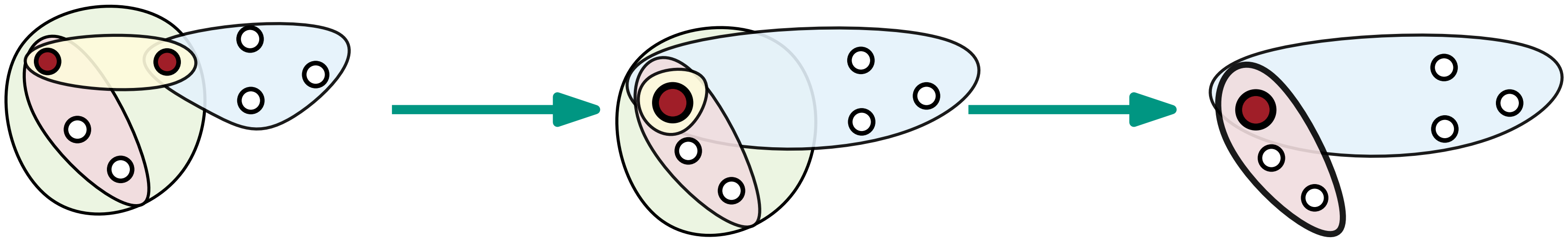
How to determine that pair?

- compute rating *r* for all pairs of adjacent hypernodes
- choose pair (*u*, *v*) with **highest** rating (priority queue)
- **update** ratings for neighbors of contracted pair

Institute of Theoretical Informatics
Algorithmics Group

n-Level Coarsening Phase

- contract only **a single pair** of vertices at **each** level



How to determine that pair?

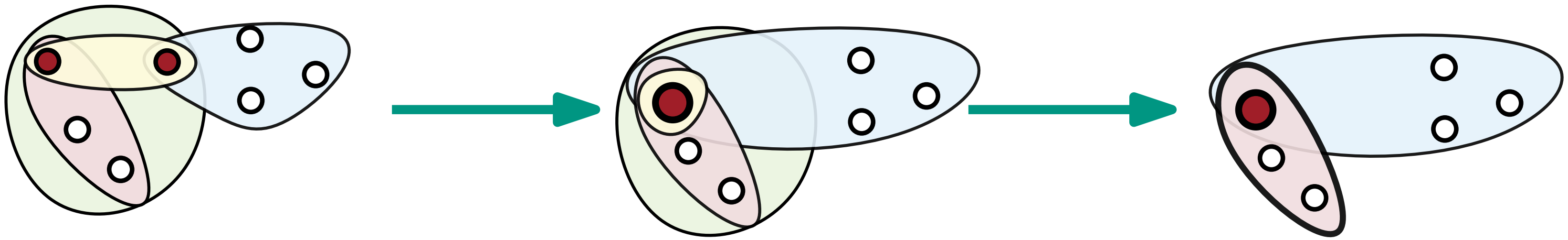
- compute rating r for all pairs of adjacent hypernodes
- choose pair (u, v) with **highest** rating (priority queue)
- **update** ratings for neighbors of contracted pair

$$r(u,v) := \frac{1}{c(v) \cdot c(u)} \sum_{\substack{\text{hyperedge } e \\ \text{containing } u,v}} \frac{\omega(e)}{|e|-1}$$

Institute of Theoretical Informatics
Algorithmics Group

n-Level Coarsening Phase

- contract only **a single pair** of vertices at **each** level



How to determine that pair?

- compute rating r for all pairs of adjacent hypernodes
- choose pair (u, v) with **highest** rating (priority queue)
- **update** ratings for neighbors of contracted pair

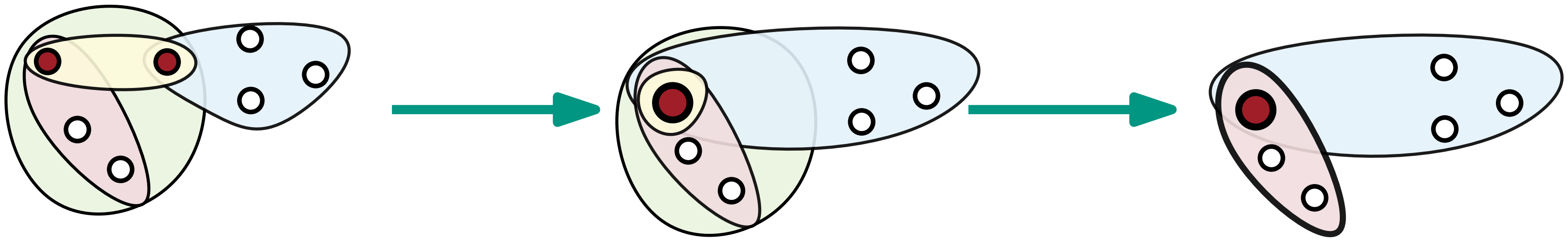
$$r(u,v) := \frac{1}{c(v)\cdot c(u)} \sum_{\substack{\text{hyperedge } e \\ \text{containing } u,v}} \frac{\omega(e)}{|e|-1}$$

large number ...

Institute of Theoretical Informatics
Algorithmics Group

n-Level Coarsening Phase

- contract only **a single pair** of vertices at **each** level



How to determine that pair?

- compute rating r for all pairs of adjacent hypernodes
- choose pair (u, v) with **highest** rating (priority queue)
- **update** ratings for neighbors of contracted pair

of heavy hyperedges ...

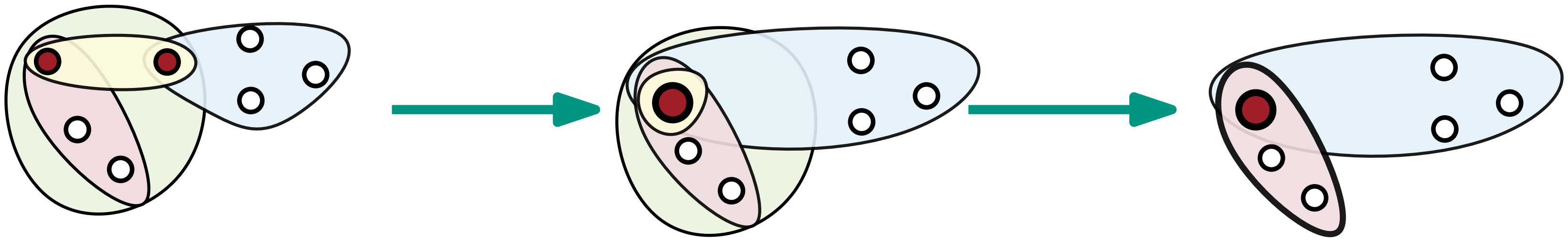
$$r(u, v) := \frac{1}{c(v) \cdot c(u)} \sum_{\substack{\text{hyperedge } e \\ \text{containing } u, v}} \frac{\omega(e)}{|e| - 1}$$

large number ...

Institute of Theoretical Informatics
Algorithmics Group

n-Level Coarsening Phase

- contract only **a single pair** of vertices at **each** level



How to determine that pair?

- compute rating *r* for all pairs of adjacent hypernodes
- choose pair (u, v) with **highest** rating (priority queue)
- **update** ratings for neighbors of contracted pair

of heavy hyperedges ...

$$r(u,v) := \frac{1}{c(v)\cdot c(u)} \sum_{\text{hyperedge } e \text{ containing } u,v} \frac{\omega(e)}{|e|-1}$$

... with small size

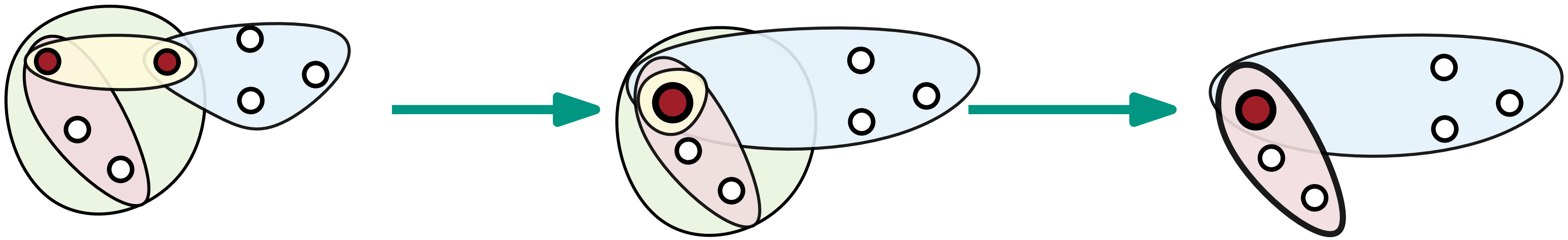
large number ...

Institute of Theoretical Informatics
Algorithmics Group

Karlsruhe Institute of Technology

n-Level Coarsening Phase

- contract only **a single pair** of vertices at **each** level



How to determine that pair?

- compute rating r for all pairs of adjacent hypernodes
- choose pair (u, v) with **highest** rating (priority queue)
- **update** ratings for neighbors of contracted pair

$$r(u,v) := \frac{1}{c(v)\cdot c(u)} \sum_{\substack{\text{hyperedge } e \\ \text{containing } u,v}} \frac{\omega(e)}{|e|-1}$$

prefer light hypernodes

large number ...

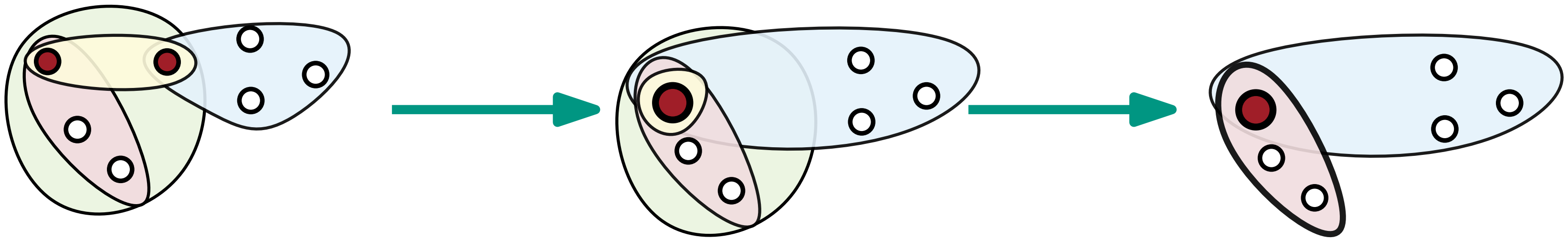
of heavy hyperedges ...

... with small size

Institute of Theoretical Informatics
Algorithmics Group

n-Level Coarsening Phase

- contract only **a single pair** of vertices at **each** level



How to determine that pair?

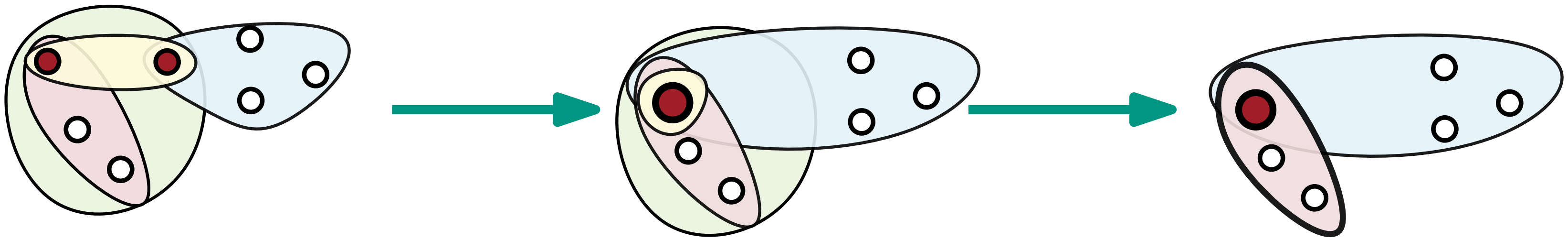
- compute rating *r* for all pairs of adjacent hypernodes
- choose pair (*u*, *v*) with **highest** rating (priority queue)
- **update** ratings for neighbors of contracted pair

- **repeat** until:
  - *t* hypernodes remain
  - no valid pair remains (size constraint on hypernodes)

Institute of Theoretical Informatics
Algorithmics Group

n-Level Coarsening Phase

- contract only **a single pair** of vertices at **each** level



How to determine that pair?

- compute rating *r* for all pairs of adjacent hypernodes
- choose pair (*u*, *v*) with **highest** rating (priority queue)
- **update** ratings for neighbors of contracted pair

- **repeat** until:
  - *t* hypernodes remain
  - no valid pair remains (size constraint on hypernodes)

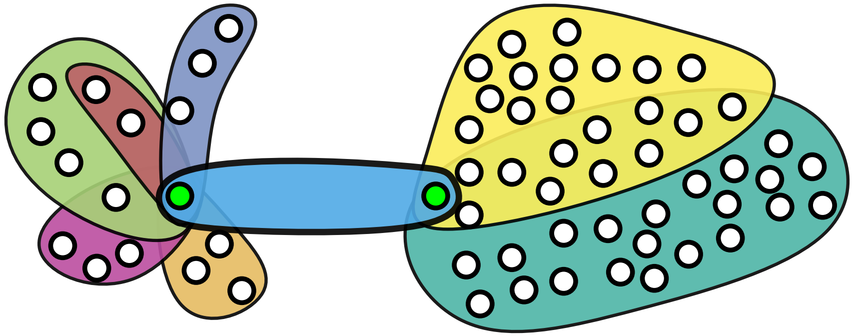
⟹ update can be **expensive!**

Institute of Theoretical Informatics
Algorithmics Group

n-Level Coarsening Phase

- **Problem:** # neighbors potentially **large**
  - high-degree hypernodes
  - large hyperedges

⇒ update **all** pins of **all** hyperedges incident to contracted pair

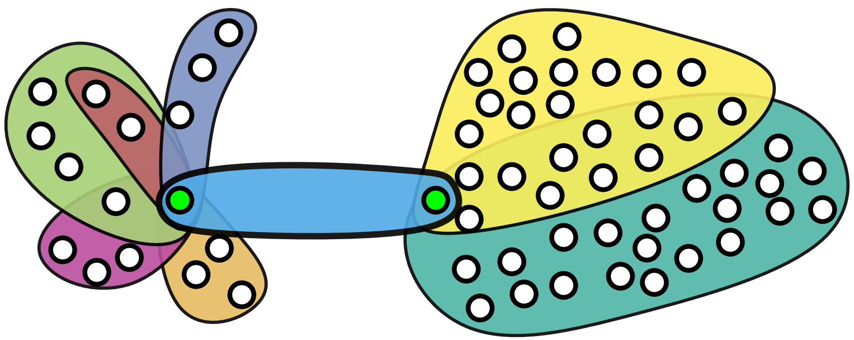


Institute of Theoretical Informatics
Algorithmics Group

n-Level Coarsening Phase

- **Problem:** # neighbors potentially **large**
  - high-degree hypernodes
  - large hyperedges

⇒ update **all** pins of **all** hyperedges incident to contracted pair



- Solution: **lazy** updates
  - **invalidate** neighboring hypernodes
  - re-calculate rating **on demand**

Initial Partitioning

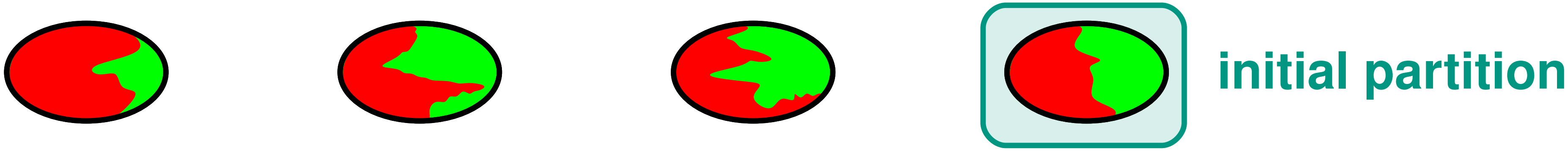
Institute of Theoretical Informatics
Algorithmics Group

Initial Partitioning

- **not** affected by *n*-level paradigm

- use **portfolio** of algorithms ⇝ diversification
  - random partitioning
  - breadth-first search
  - greedy hypergraph growing
  - size-constrained label propagation

⇒ try all algorithms multiple times
⇒ select partition with **best** cut & **lowest** imbalance as initial partition

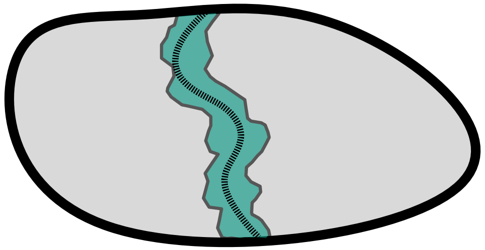


Local Search

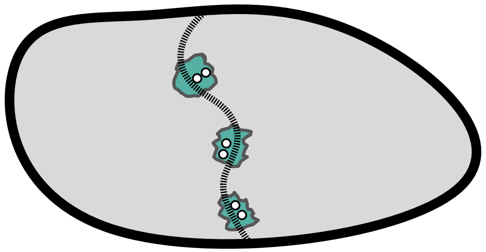
Institute of Theoretical Informatics
Algorithmics Group

Localized Local Search – Idea

- traditional multilevel algorithms
  - uncontract one **level**
  - $\rightsquigarrow$ local search around **complete** border



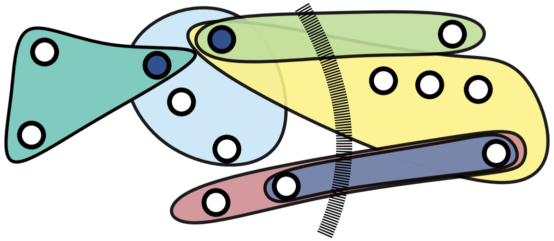
- *n*-level **localized** local search [KaSPar]
  - uncontract **a single pair** of nodes
  - $\rightsquigarrow$ local search around **2** nodes
  - $\Rightarrow$ fine-grained optimization



- limit search to **constant** # of moves **per level**
  - otherwise $\rightsquigarrow$ $|V|^2$ local search steps in total
  - $\Rightarrow$ stop pass after x fruitless moves

Institute of Theoretical Informatics
Algorithmics Group

Localized FM Local Search – Outline

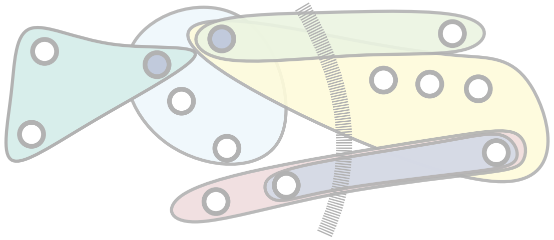


- hypernodes ⇝ unmarked, active, marked
- start around uncontracted vertex pair

Institute of Theoretical Informatics
Algorithmics Group

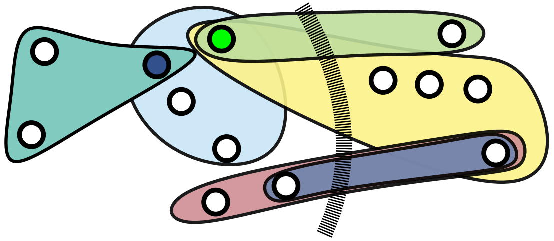
Localized FM Local Search – Outline

- hypernodes $\rightsquigarrow$ unmarked, active, marked
- start around uncontracted vertex pair



- compute gain for move to other block:

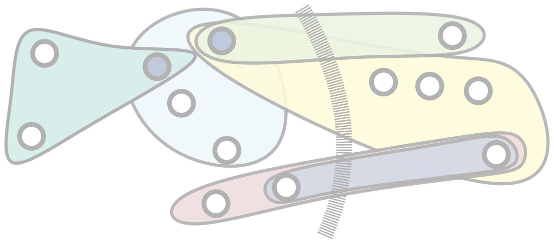
$$g(v) = \sum_{\substack{\text{hyperedge } e \\ \text{containing } v}} \begin{cases} +\omega(e) & \text{if \# pins in source} = 1 \\ -\omega(e) & \text{if \# pins in target} = 0 \end{cases}$$



  - $\rightsquigarrow$ border hypernodes become active

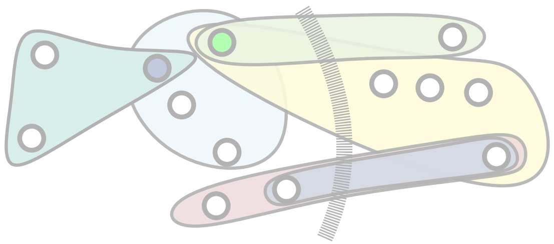
Localized FM Local Search – Outline

- hypernodes ⇝ unmarked, active, marked
- start around uncontracted vertex pair



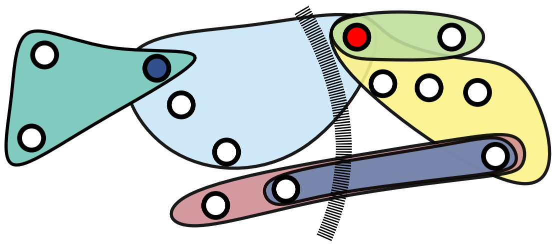
- compute gain for move to other block:

$$g(v) = \sum_{\substack{\text{hyperedge } e \\ \text{containing } v}} \begin{cases} +\omega(e) & \text{if \# pins in source} = 1 \\ -\omega(e) & \text{if \# pins in target} = 0 \end{cases}$$



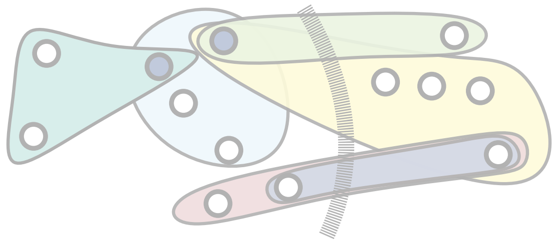
  - ⇝ border hypernodes become active

- move highest-gain node to opposite block
  - ⇝ node becomes marked



Localized FM Local Search – Outline

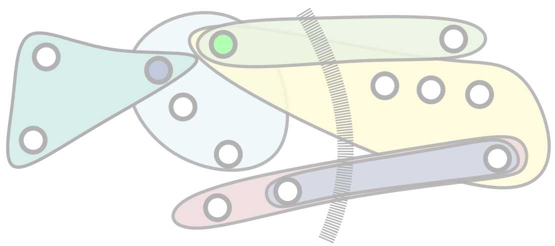
- hypernodes $\rightsquigarrow$ unmarked, active, marked
- start around uncontracted vertex pair



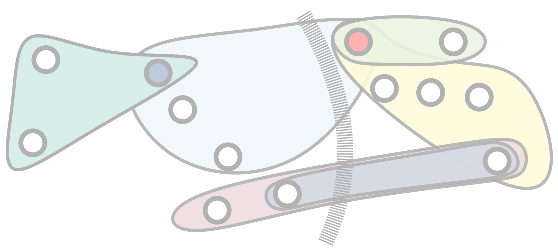
- compute gain for move to other block:

$$g(v) = \sum_{\substack{\text{hyperedge } e \\ \text{containing } v}} \begin{cases} +\omega(e) & \text{if \# pins in source} = 1 \\ -\omega(e) & \text{if \# pins in target} = 0 \end{cases}$$

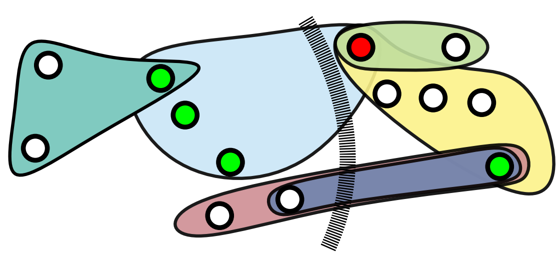
  - $\rightsquigarrow$ border hypernodes become active



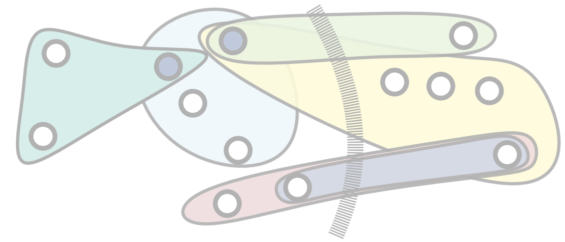
- move highest-gain node to opposite block
  - $\rightsquigarrow$ node becomes marked



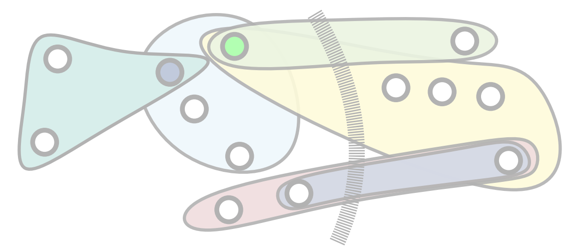
- unmarked neighbors $\rightsquigarrow$ active (if border node)
- active neighbors $\rightsquigarrow$ update gain



Localized FM Local Search – Outline



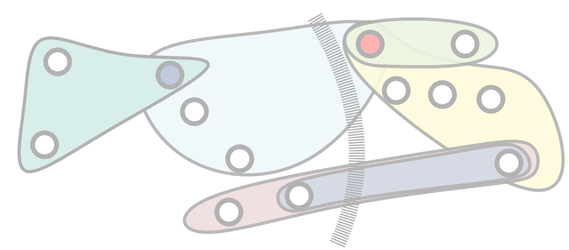
- hypernodes ⇝ unmarked, active, marked
- start around uncontracted vertex pair



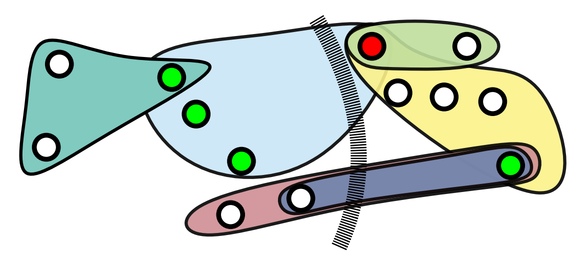
- compute gain for move to other block:

$$g(v) = \sum_{\substack{\text{hyperedge } e \\ \text{containing } v}} \begin{cases} +\omega(e) & \text{if \# pins in source} = 1 \\ -\omega(e) & \text{if \# pins in target} = 0 \end{cases}$$

  - ⇝ border hypernodes become active



- move highest-gain node to opposite block
  - ⇝ node becomes marked



- unmarked neighbors ⇝ active (if border node)
- active neighbors ⇝ update gain

⇒ update & activation can be **expensive!**

Localized FM Local Search – Engineering

Problem: # neighbors potentially **large**

- high-degree hypernodes
- large hyperedges

$\Longrightarrow$ **large** number of activations & updates on **each** level

Institute of Theoretical Informatics
Algorithmics Group

Localized FM Local Search – Engineering

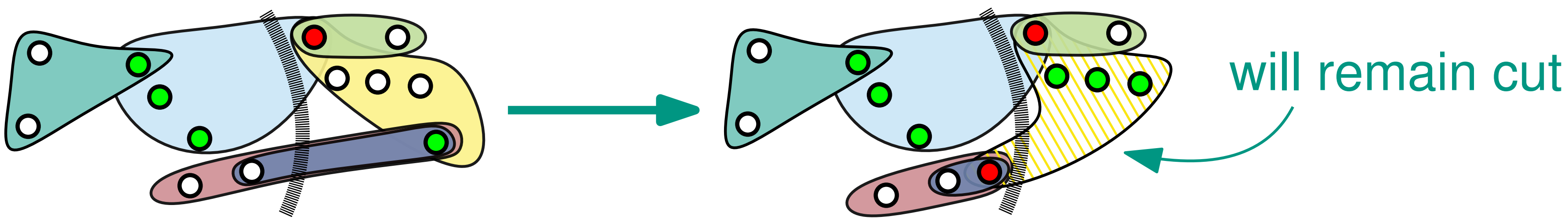
Problem: # neighbors potentially **large**

- high-degree hypernodes
- large hyperedges

⟹ **large** number of activations & updates on **each** level

Known solutions for updates:

- perform δ-gain updates [Papa, Markov]
- exclude **locked** hyperedges from gain update [Krishnamurthy]



Localized FM Local Search – Engineering

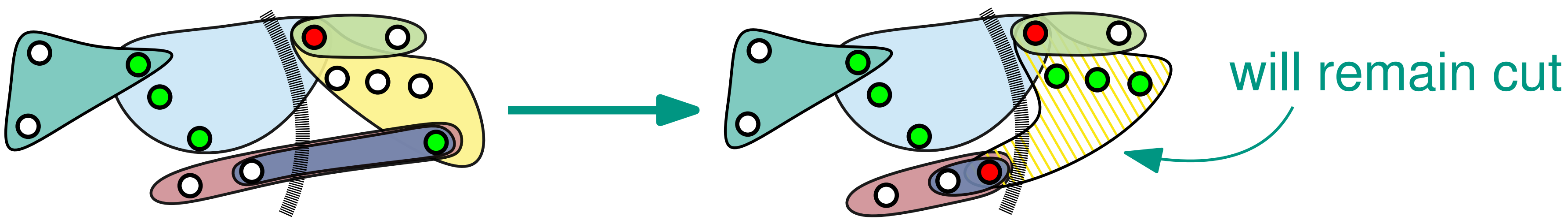
Problem: # neighbors potentially **large**

- high-degree hypernodes
- large hyperedges

⇒ **large** number of activations & updates on **each** level

Known solutions for updates:

- perform δ-gain updates [Papa, Markov]
- exclude **locked** hyperedges from gain update [Krishnamurthy]



New solution for activations:

- **cache** gain values
  - compute gain $g(v)$ at most **once** along the n-level hierarchy

Experiments – Benchmark Setup

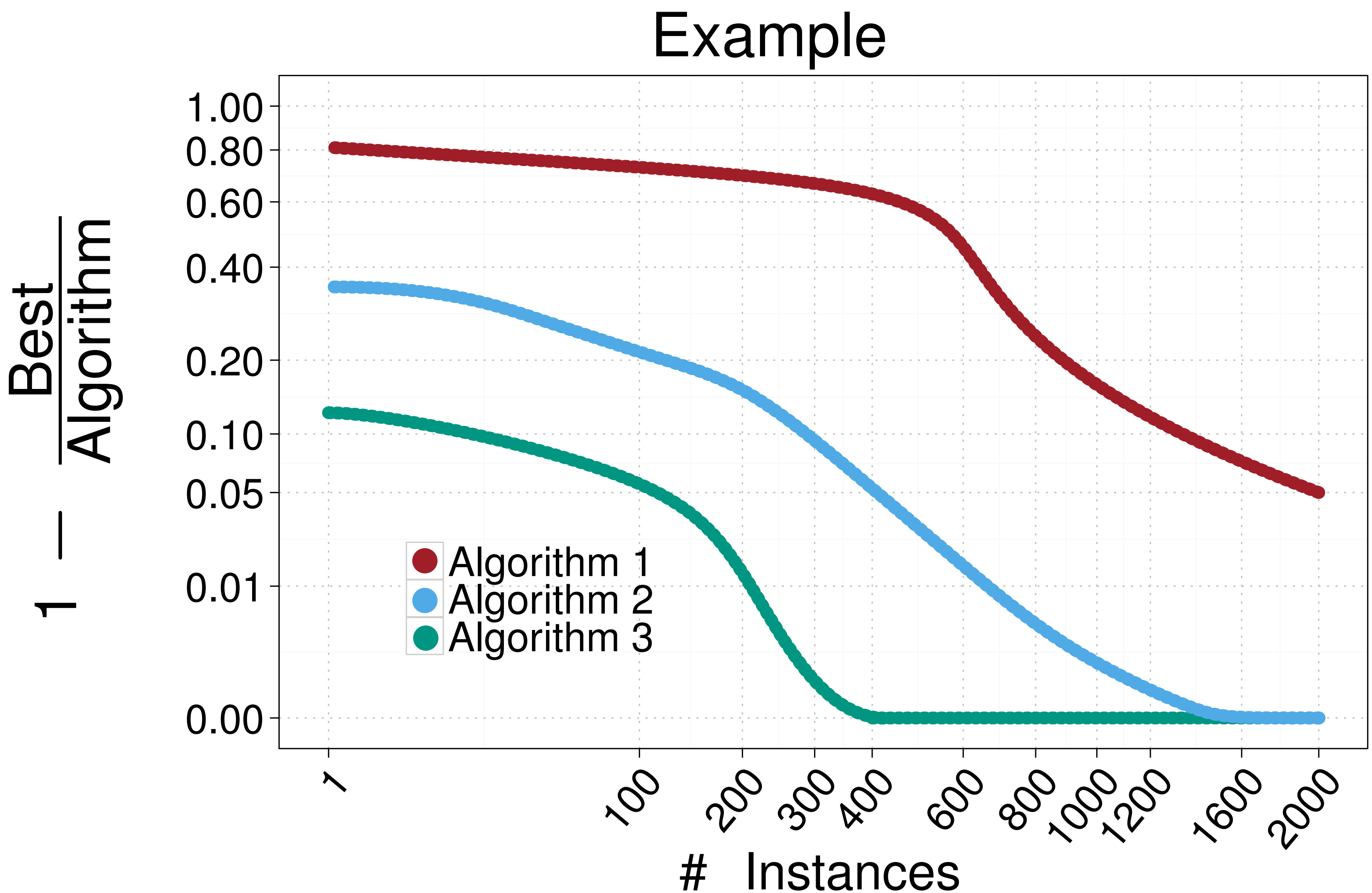
- System: 1 core of 2 Intel Xeon E5-2670 @ 2.6 Ghz, 64 GB RAM

- # Hypergraphs: [publicly available]
  - UF Sparse Matrix Collection 192
  - SAT Competition 2014 Application Track 100
  - ISPD98 VLSI Circuit Benchmark Suite 18

- $k \in \{2, 4, 8, 16, 32, 64, 128\}$ → **2170 instances**
- imbalance: $\varepsilon = 3\%$
- 250 min time limit

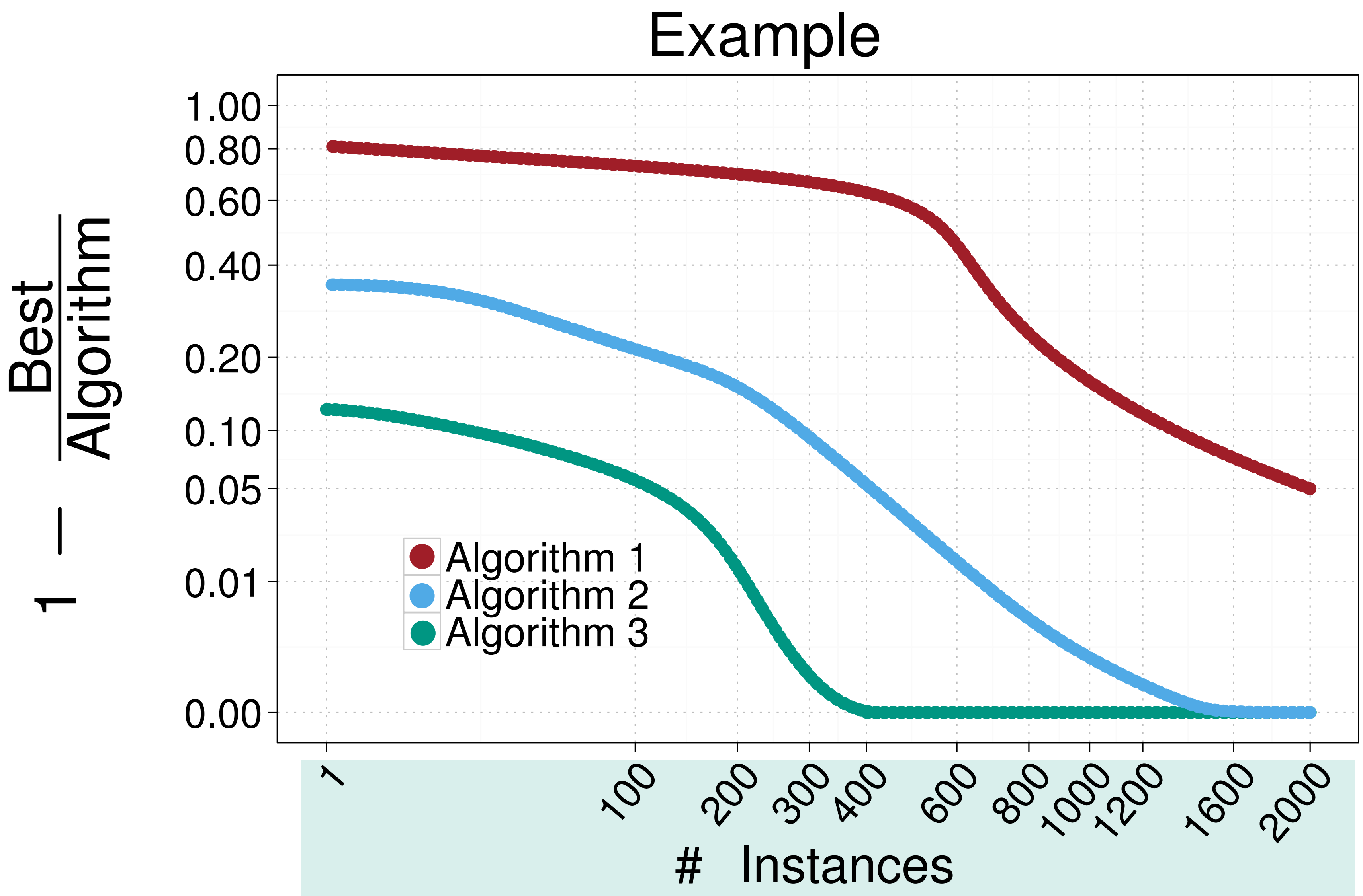
- Comparison with:
  - hMetis-R & hMetis-K
  - PaToH-Default & PaToH-Quality

Experimental Results – Partitioning Quality



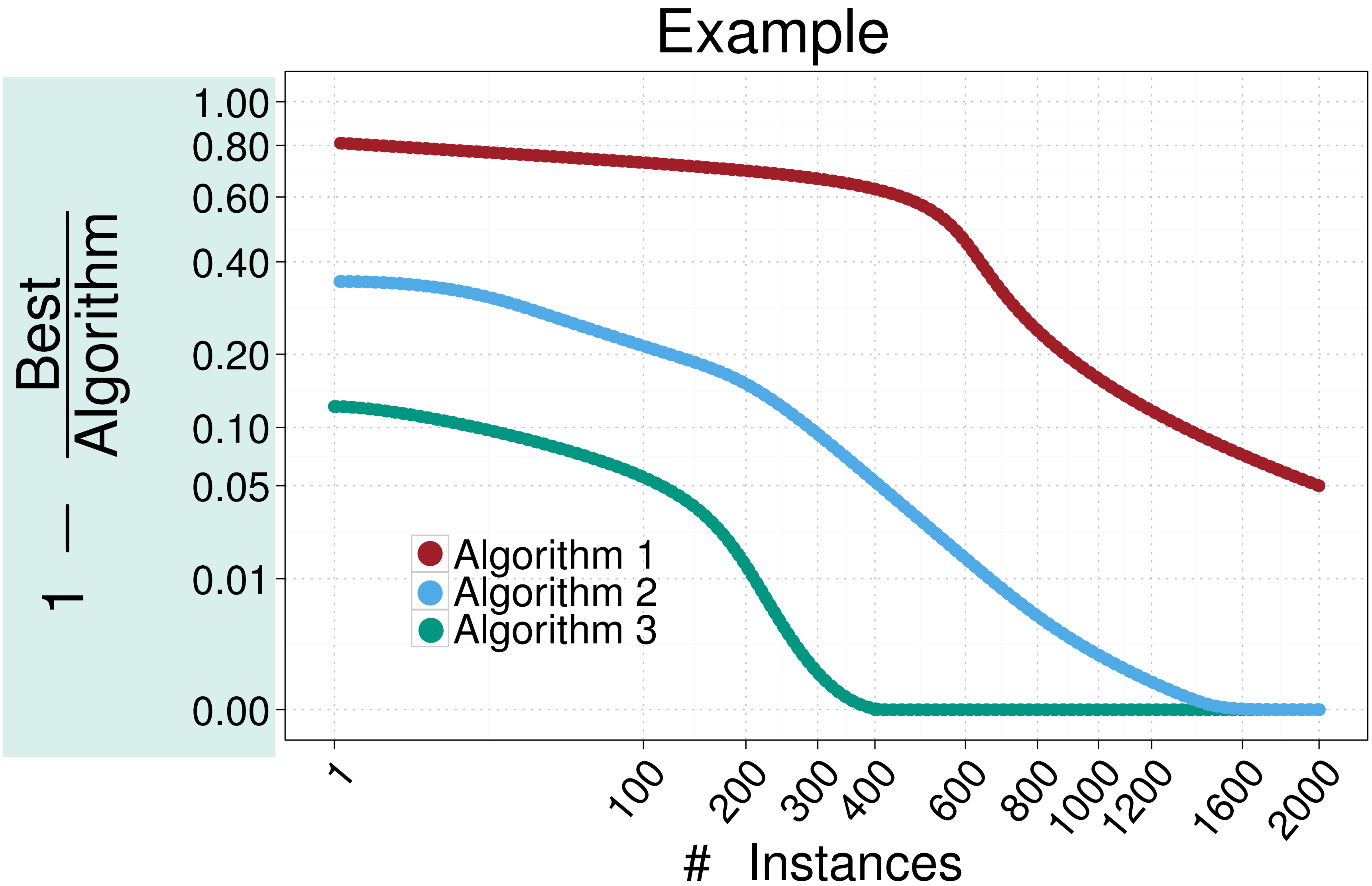
Institute of Theoretical Informatics
Algorithmics Group

Experimental Results – Partitioning Quality



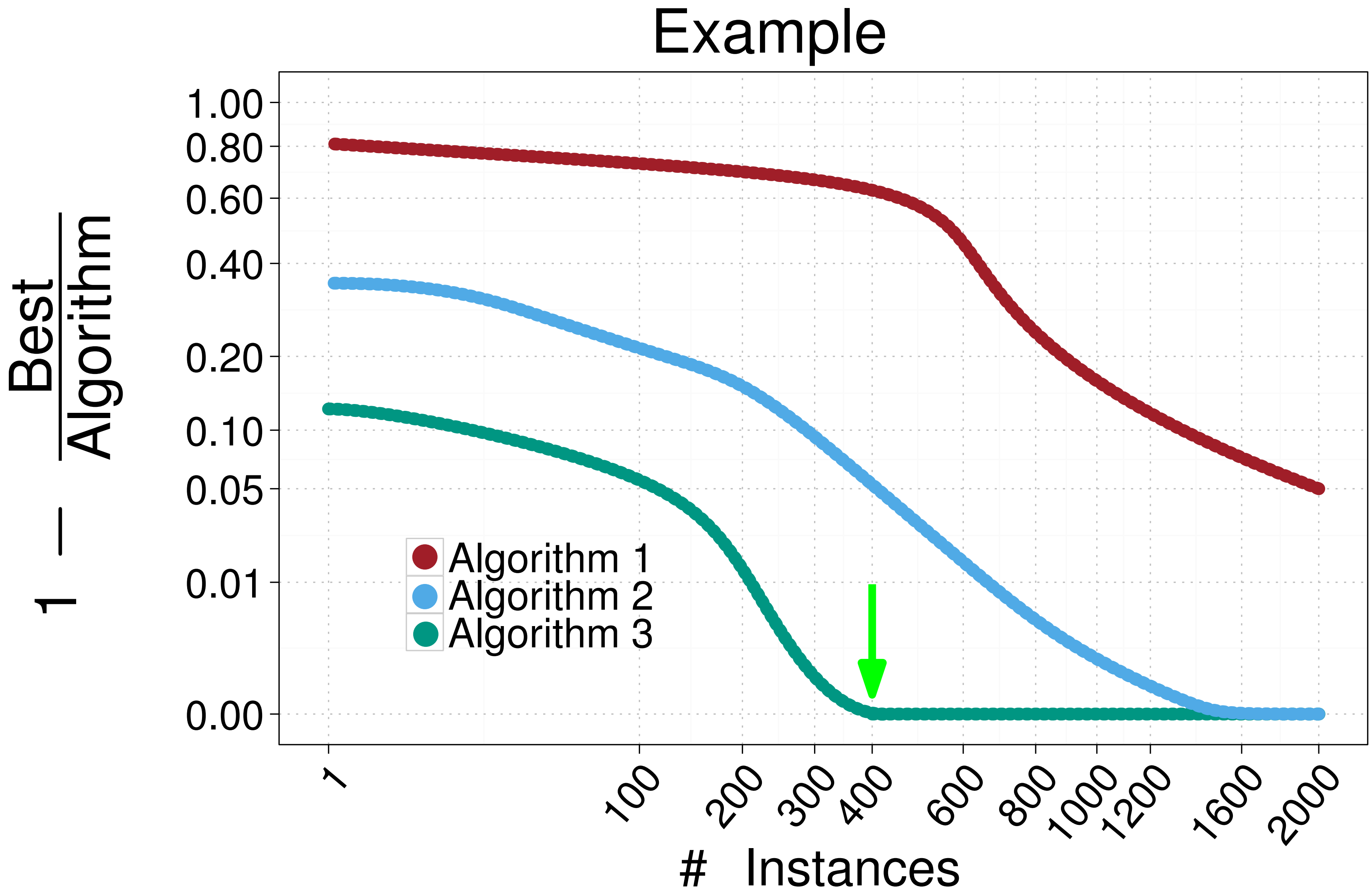
Institute of Theoretical Informatics
Algorithmics Group

Experimental Results – Partitioning Quality



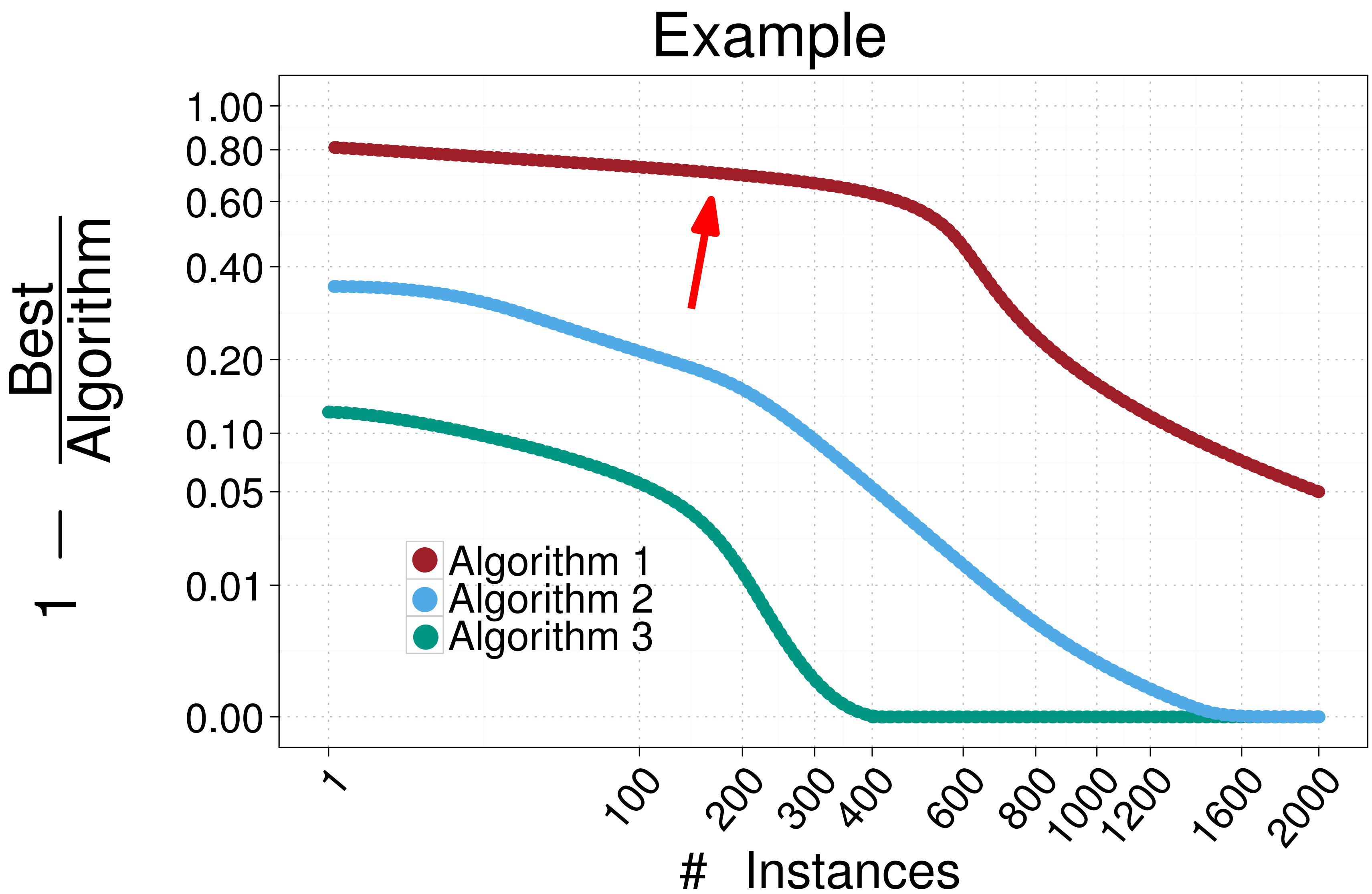
Institute of Theoretical Informatics
Algorithmics Group

Experimental Results – Partitioning Quality

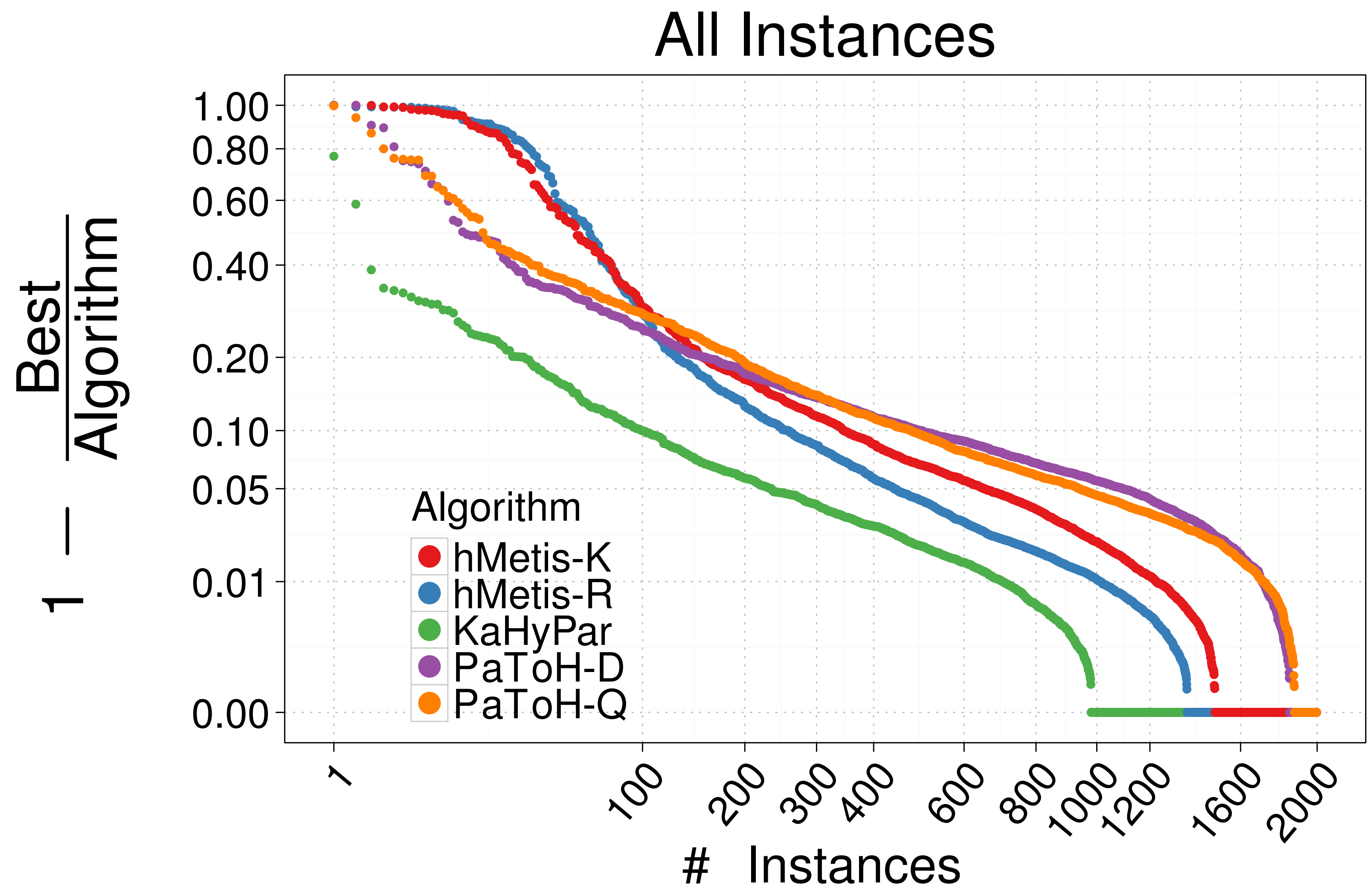


Institute of Theoretical Informatics
Algorithmics Group

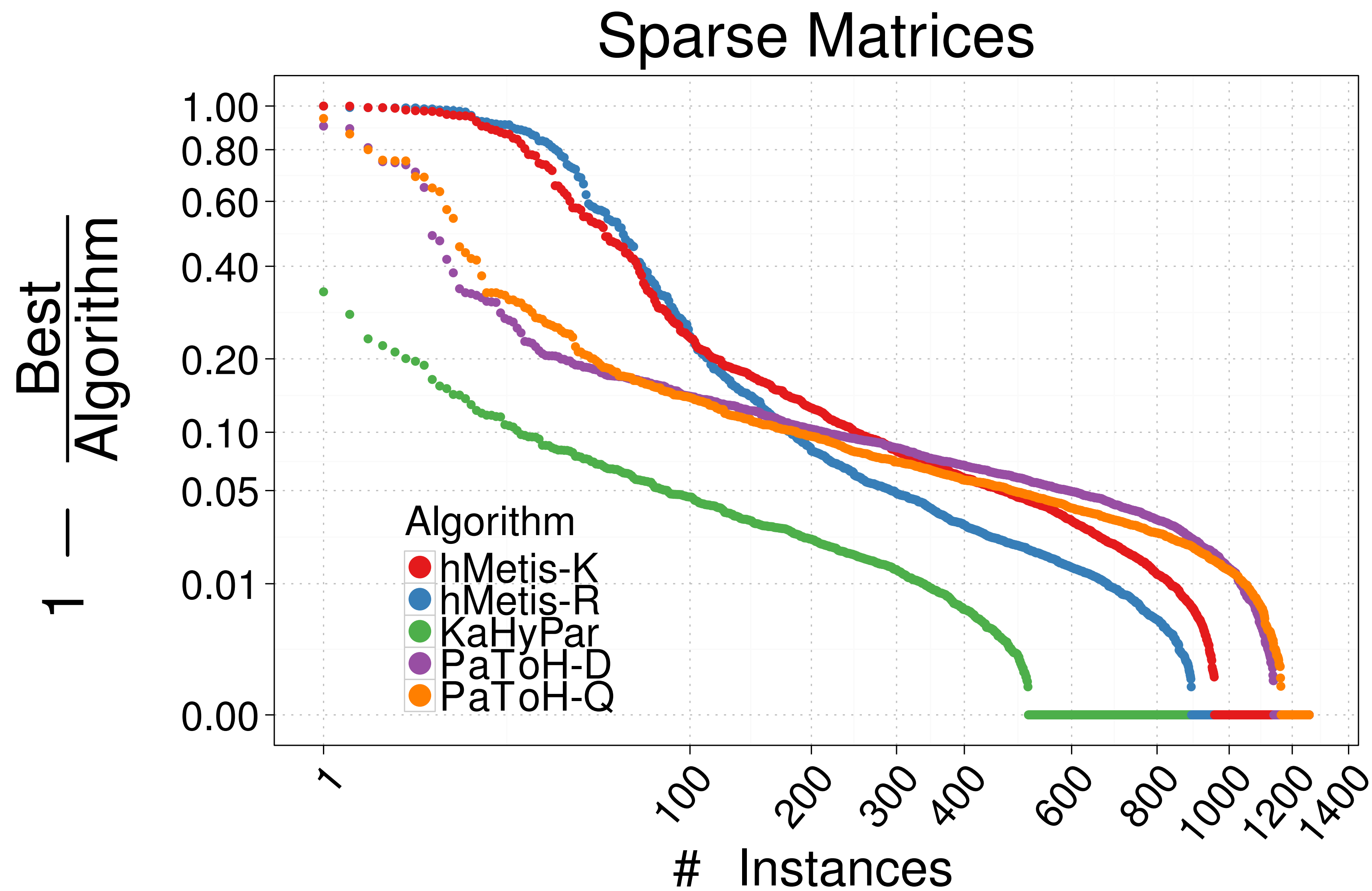
Experimental Results – Partitioning Quality



Experimental Results – Partitioning Quality

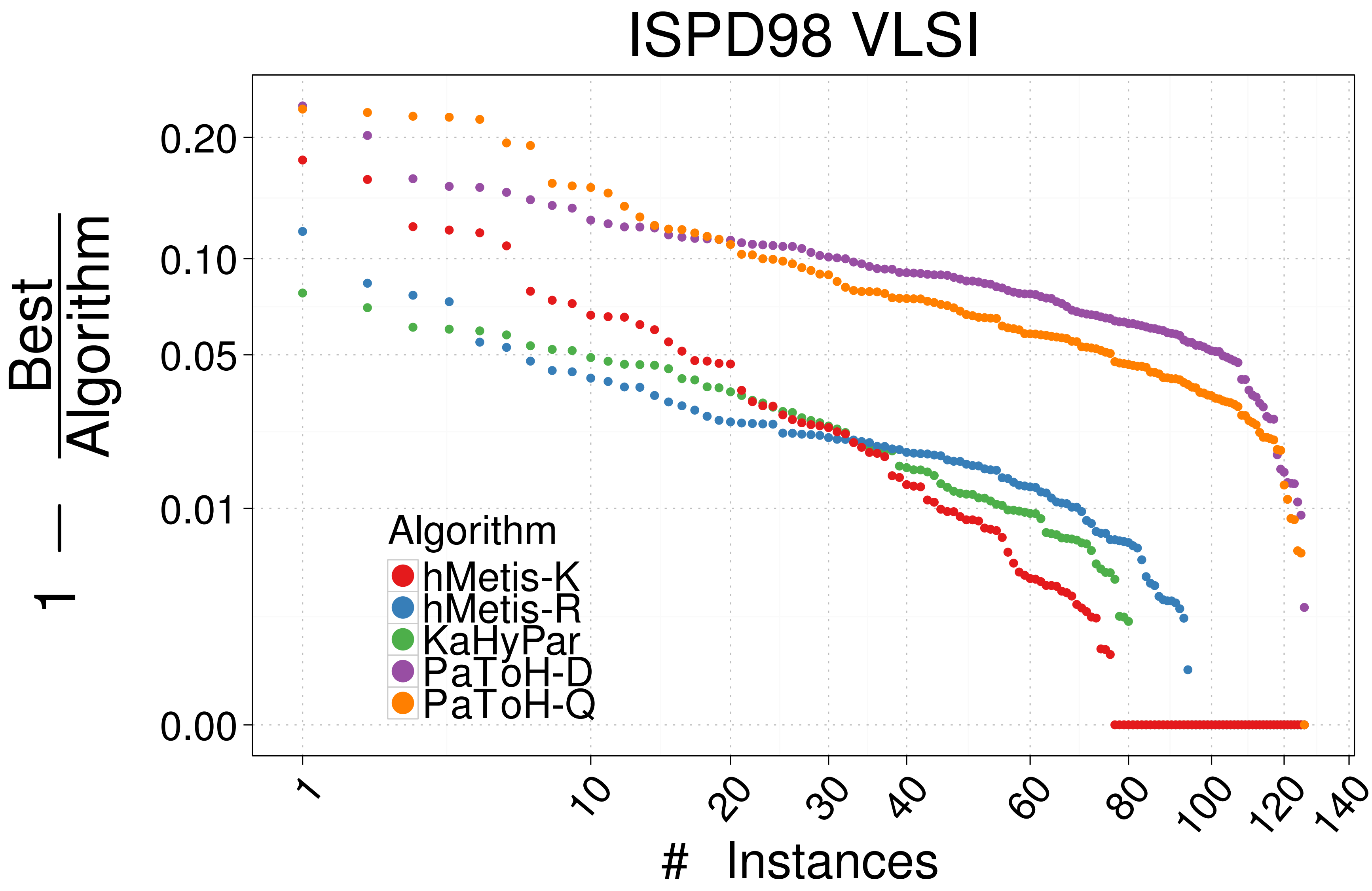


Experimental Results – Partitioning Quality



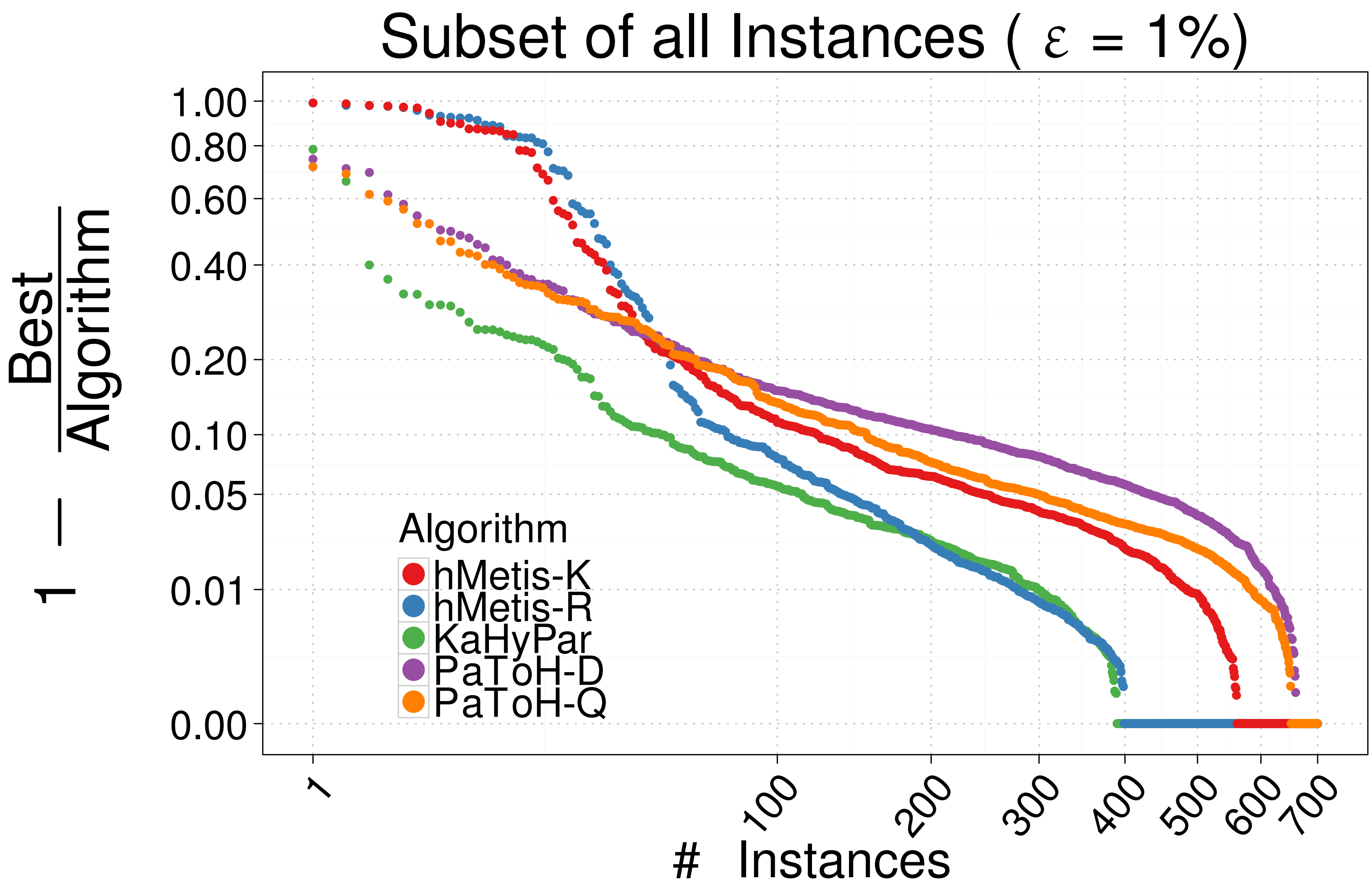
Institute of Theoretical Informatics
Algorithmics Group

Experimental Results – Partitioning Quality



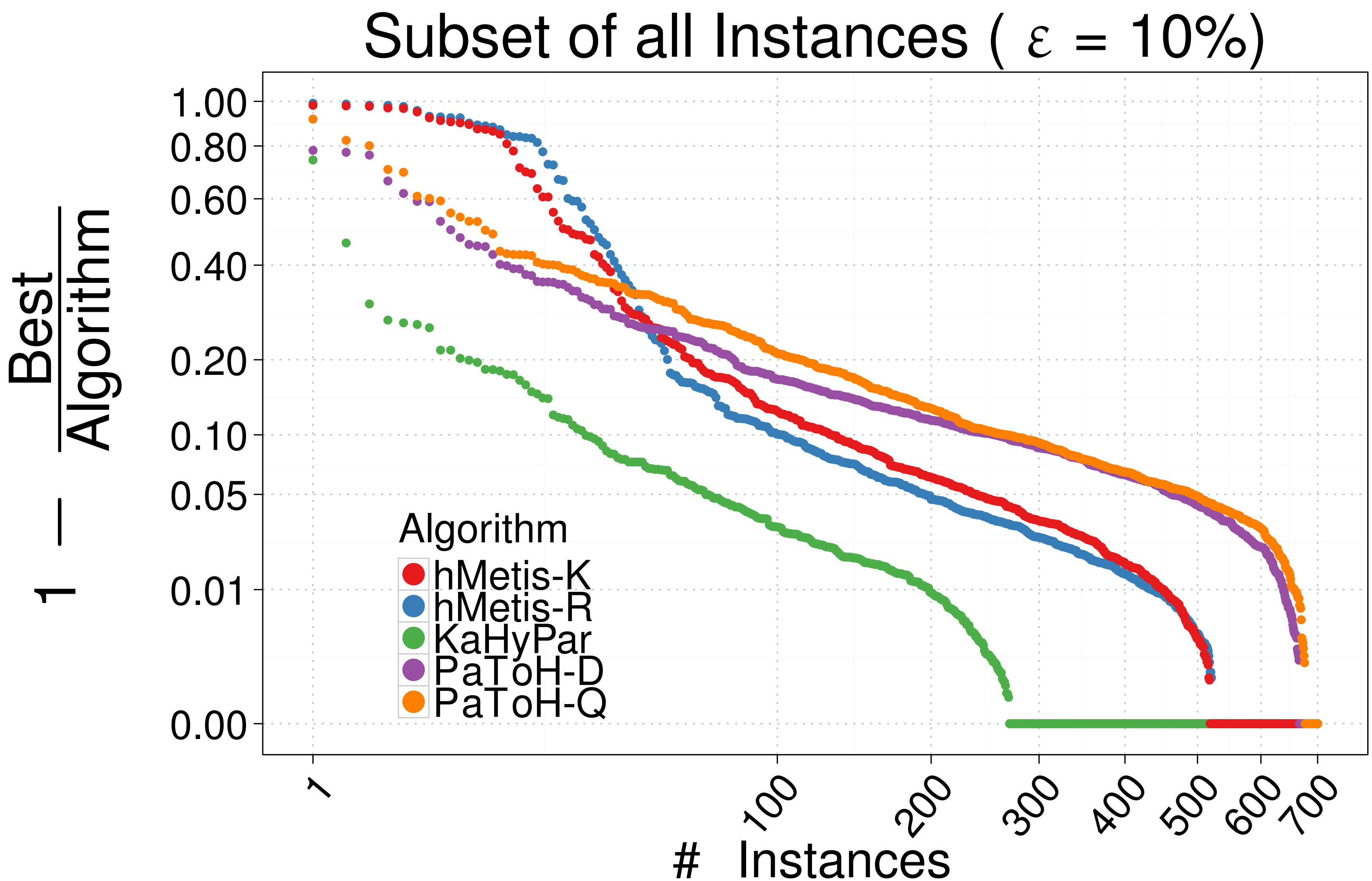
Institute of Theoretical Informatics
Algorithmics Group

Experimental Results – Smaller Imbalance

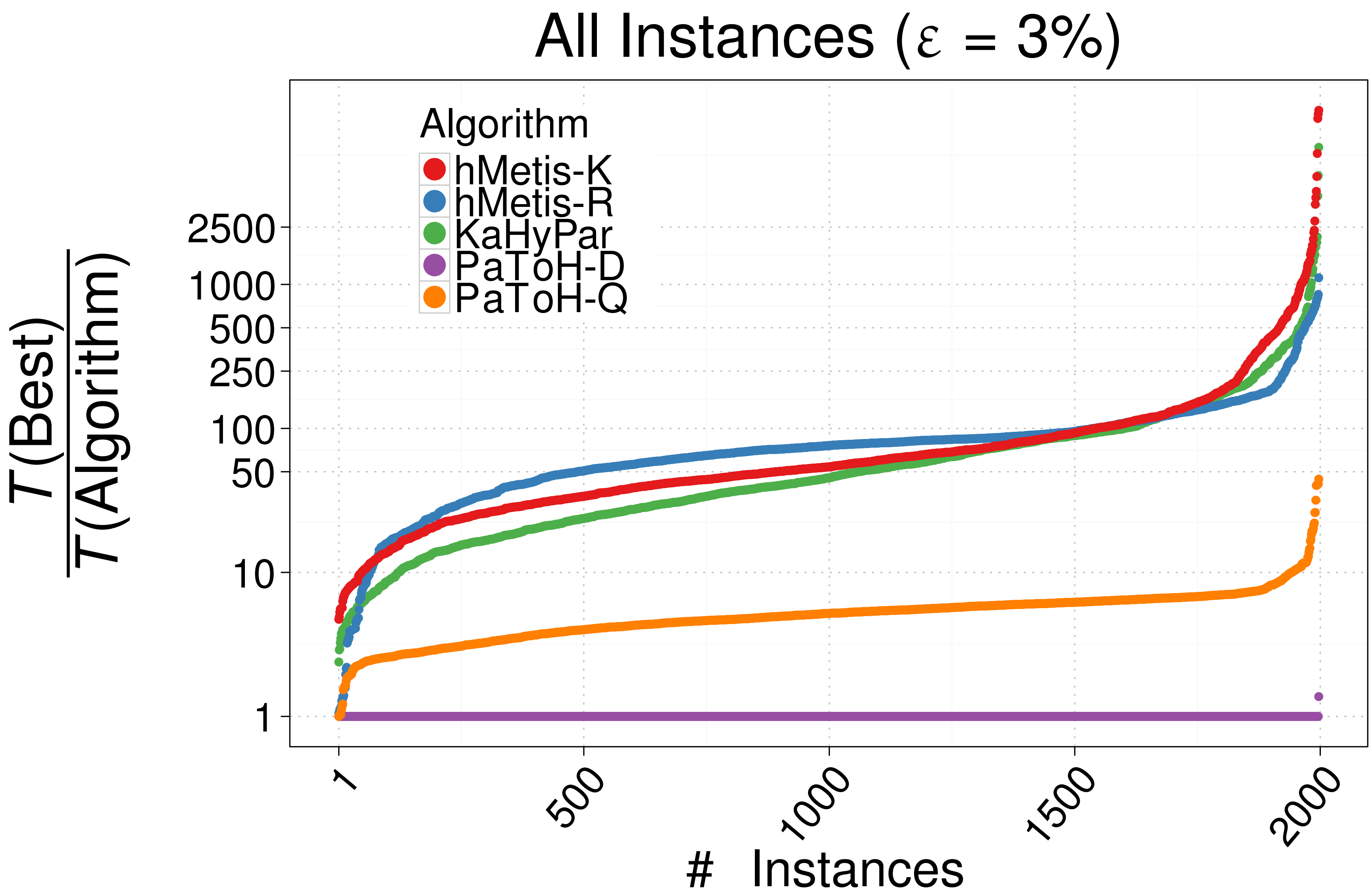


Institute of Theoretical Informatics
Algorithmics Group

Experimental Results – Larger Imbalance



Experimental Results – Running Time

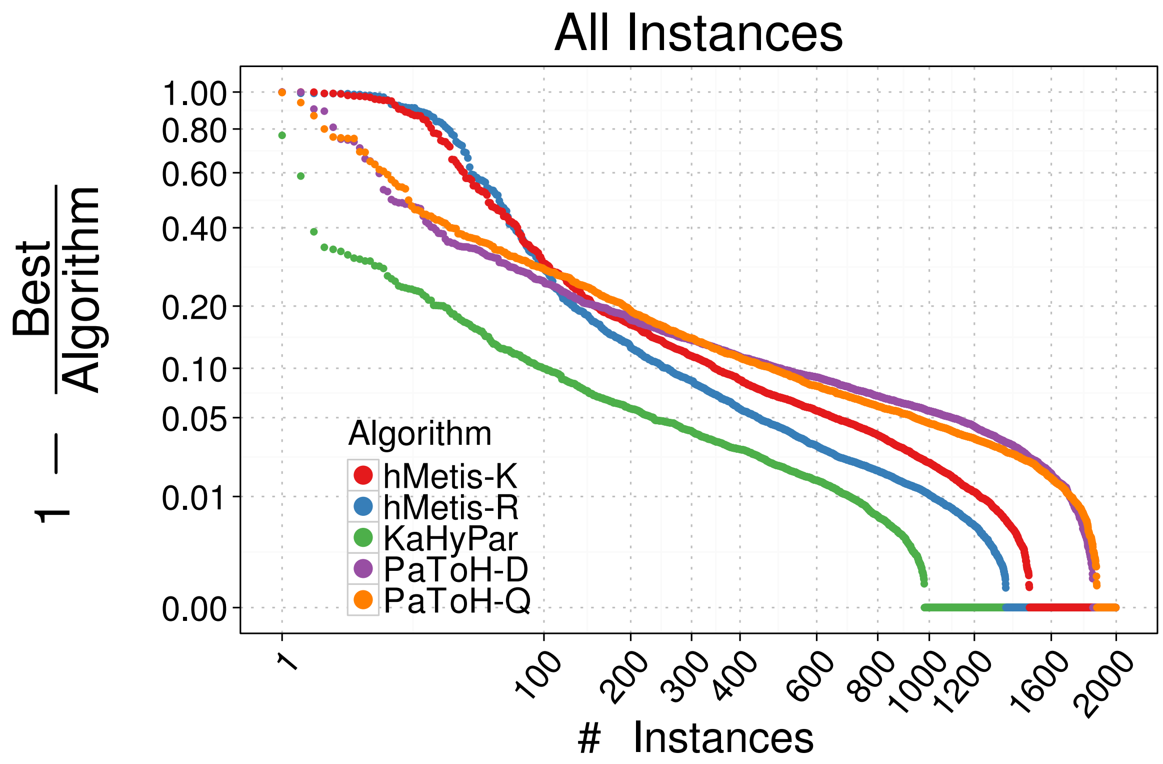


Future Work

- **improve running time:**
  - ignore “large” hyperedges [PaToH]
  - stop local search if improvement becomes unlikely [KaSPar]
- **improve quality:**
  - introduce V-cycles
  - evolutionary algorithm [KaHIP]
- **improve balancing:**
  - optimize locally - rebalance globally

Conclusion & Discussion

- **evade** running time / quality tradeoff of multilevel algorithms
  $\rightsquigarrow$ $\mathbf{n}$-level hierarchy
  - engineered coarsening phase
  - portfolio-based approach to initial partitioning
  - highly tuned local search algorithm

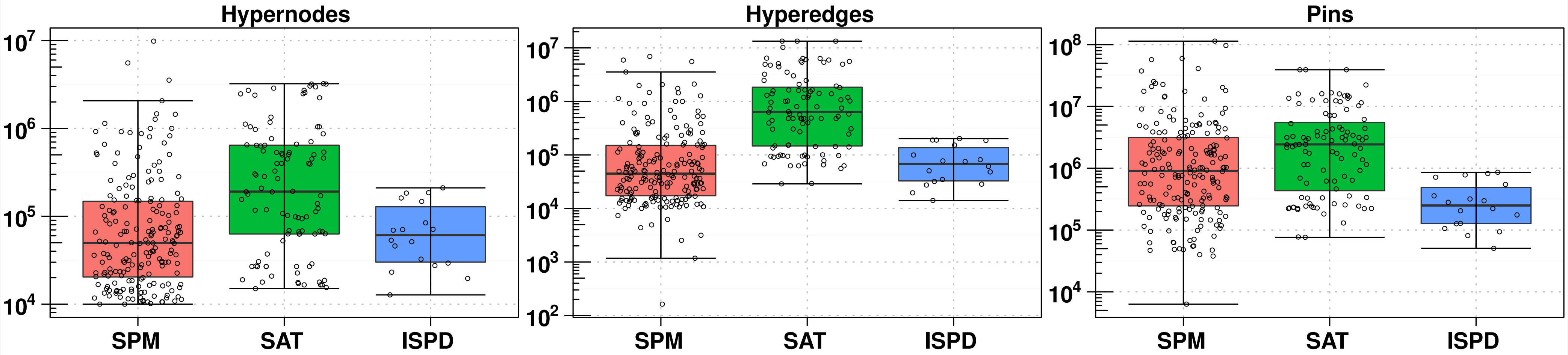


Institute of Theoretical Informatics
Algorithmics Group

Coffee Break!

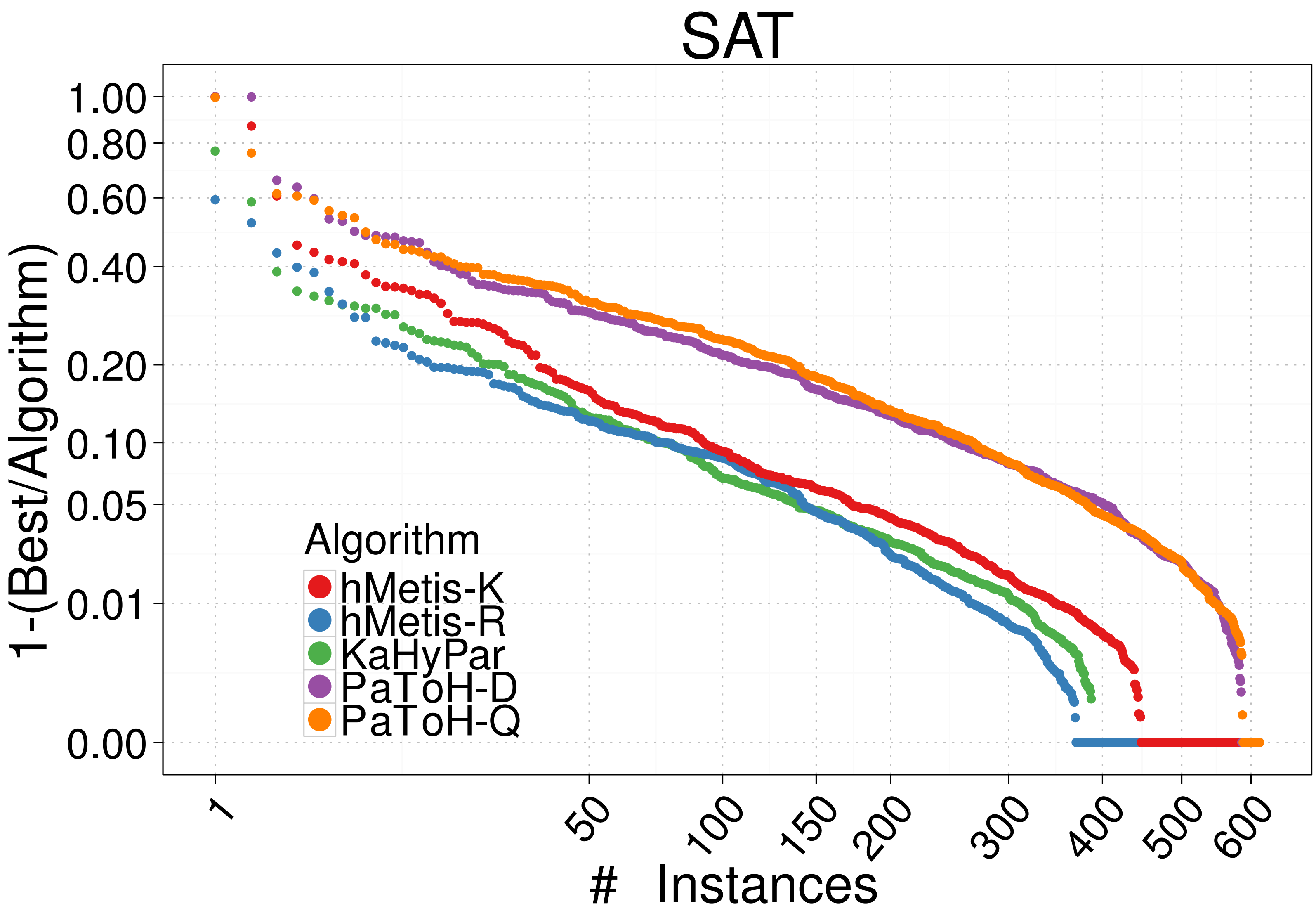
Institute of Theoretical Informatics
Algorithmics Group

Benchmark Set Details



Institute of Theoretical Informatics
Algorithmics Group

Benchmark Results – Partitioning Quality



Institute of Theoretical Informatics
Algorithmics Group