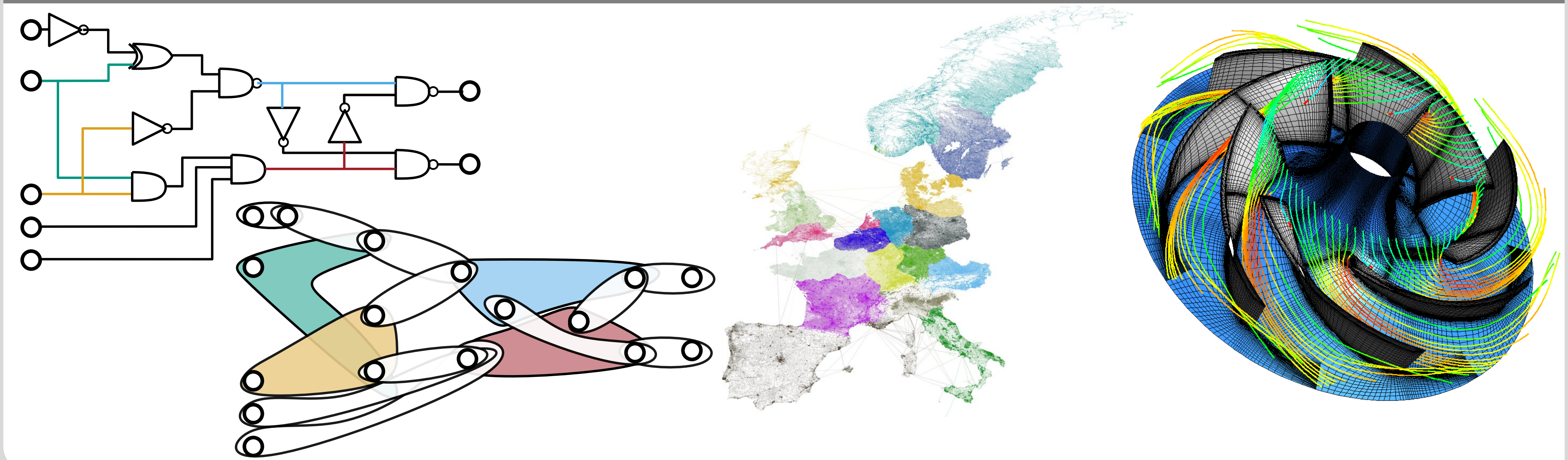
High Quality Graph and Hypergraph Partitioning

2nd BMBF Big Data All Hands Meeting · October 11, 2017

Yaroslav Akhremtsev, Peter Sanders, Sebastian Schlag, Christian Schulz

INSTITUTE OF THEORETICAL INFORMATICS · ALGORITHMICS GROUP

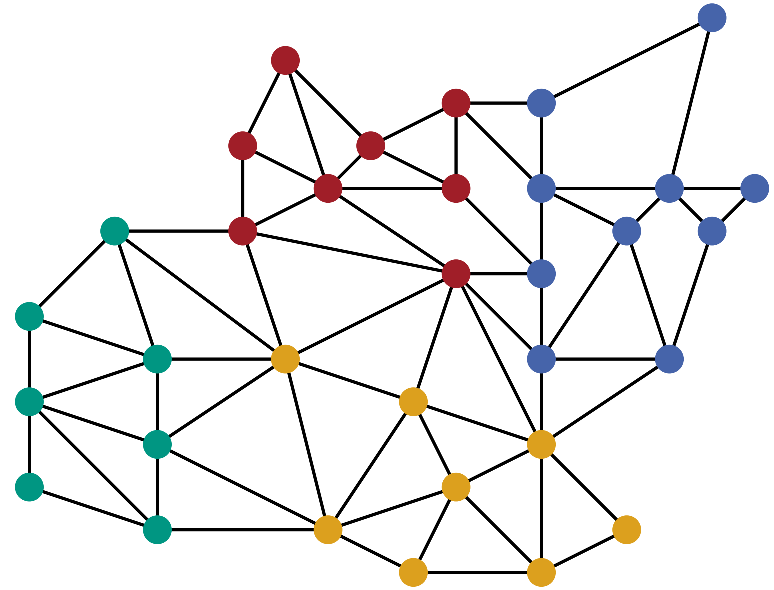


Graphs and Hypergraphs

Graph $G = (V, E)$

vertices → V, edges → E

- models **relationships** between **objects**
- dyadic (**2-ary**) relationships

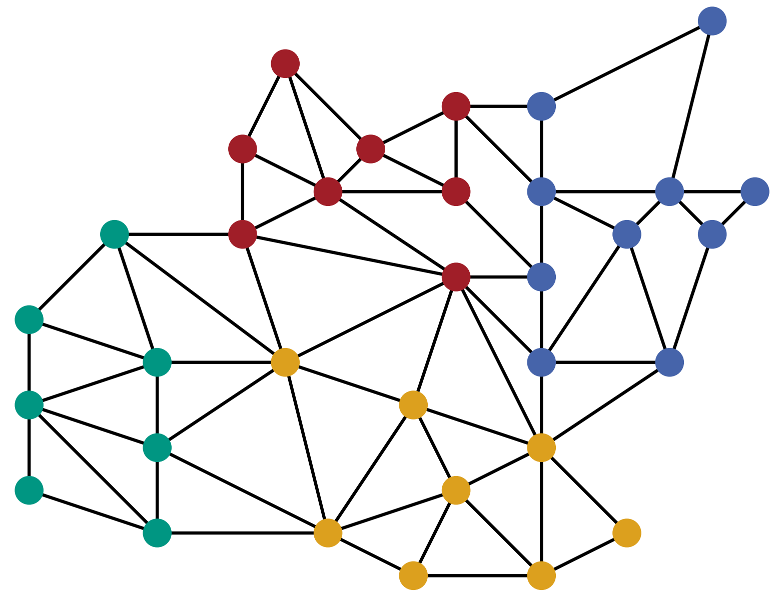


Graphs and Hypergraphs

Karlsruhe Institute of Technology

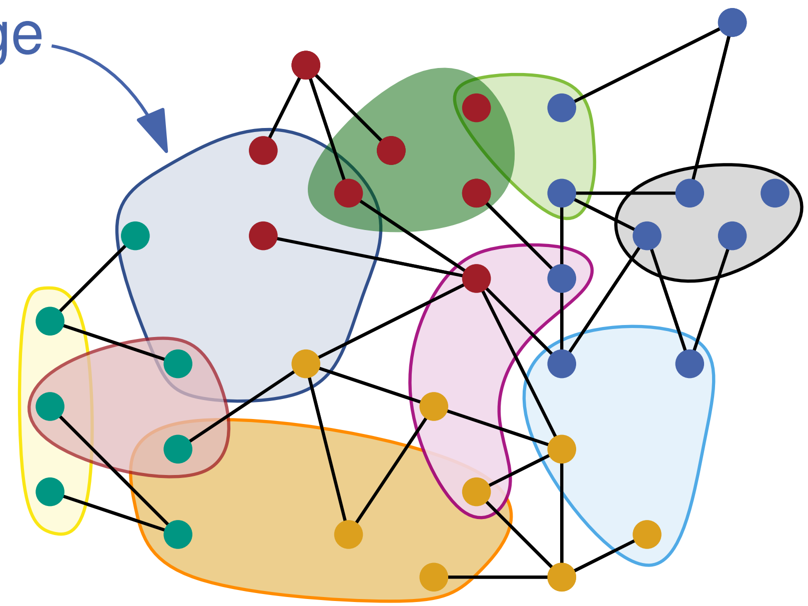
Graph $G = (V, E)$

vertices → V, edges → E



- models **relationships** between **objects**
- dyadic (**2-ary**) relationships

hyperedge



Hypergraph $H = (V, E)$

- Generalization of a graph
  $\Rightarrow$ hyperedges connect ≥ 2 nodes
- arbitrary (**d-ary**) relationships
- Edge set $E \subseteq \mathcal{P}(V) \setminus \emptyset$

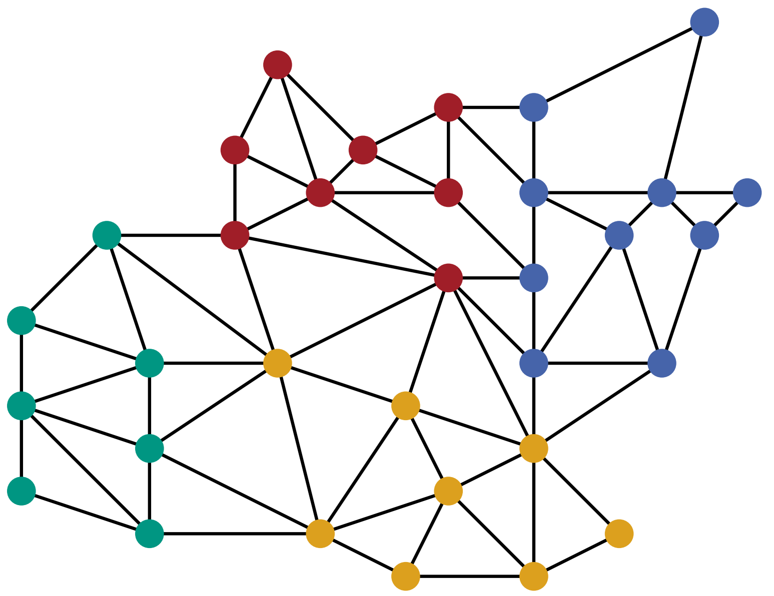
ε-Balanced Graph and Hypergraph Partitioning

Partition (hyper)graph $G = (V, E, c : V \to \mathbb{R}_{>0}, \omega : E \to \mathbb{R}_{>0})$ into **k** disjoint blocks $V_1, \ldots, V_k$ s.t.

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

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

- **objective** function on edges is **minimized**



Institute of Theoretical Informatics
Algorithmics Group

ε-Balanced Graph and Hypergraph Partitioning

Karlsruhe Institute of Technology

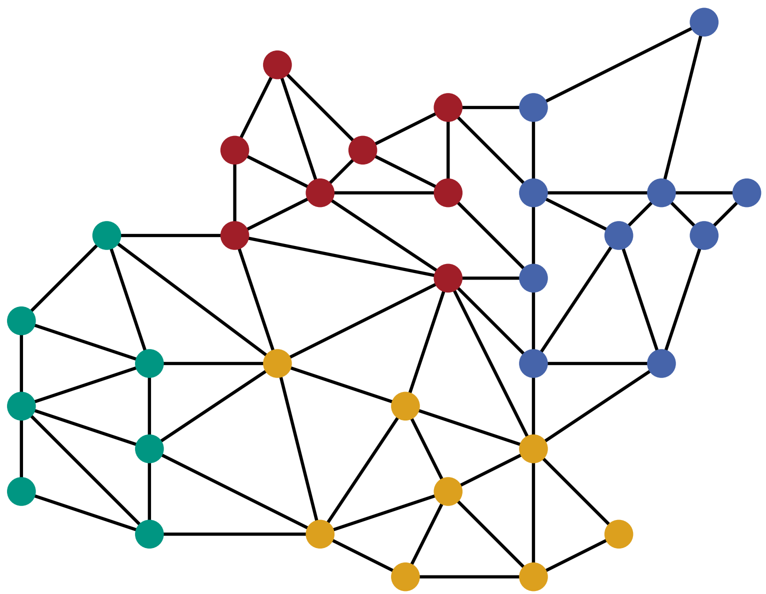
Partition (hyper)graph $G = (V, E, c : V \to \mathbb{R}_{>0}, \omega : E \to \mathbb{R}_{>0})$ into **k** disjoint blocks $V_1, \ldots, V_k$ s.t.

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

imbalance parameter

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

- **objective** function on edges is **minimized**



ε-Balanced Graph and Hypergraph Partitioning

Partition (hyper)graph $G = (V, E, c : V \to \mathbb{R}_{>0}, \omega : E \to \mathbb{R}_{>0})$ into **k** disjoint blocks $V_1, \ldots, V_k$ s.t.

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

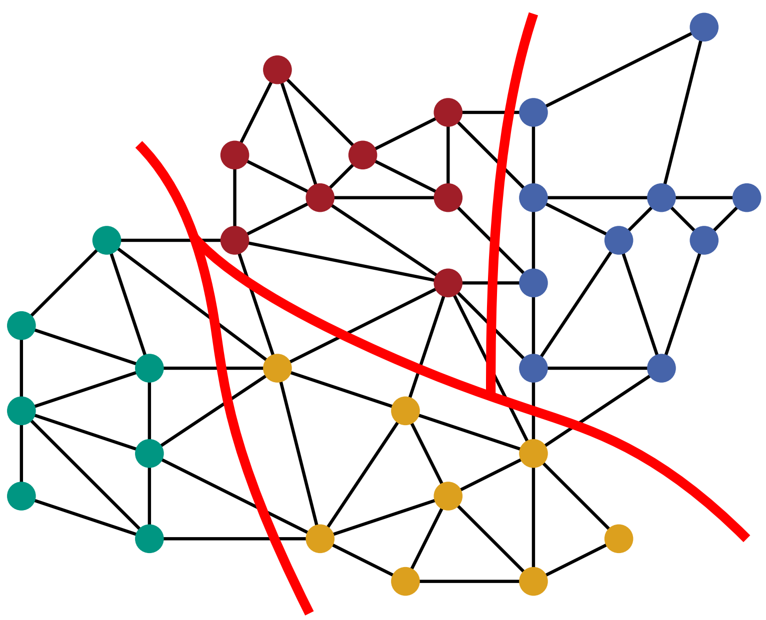
imbalance parameter

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

- **objective** function on edges is **minimized**

Common Objectives:

- Graphs:
  - cut: $\sum_{e \in \text{cut}} \omega(e)$



ε-Balanced Graph and Hypergraph Partitioning

Partition (hyper)graph $G = (V, E, c : V \to \mathbb{R}_{>0}, \omega : E \to \mathbb{R}_{>0})$
into **k** disjoint blocks $V_1, \ldots, V_k$ s.t.

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

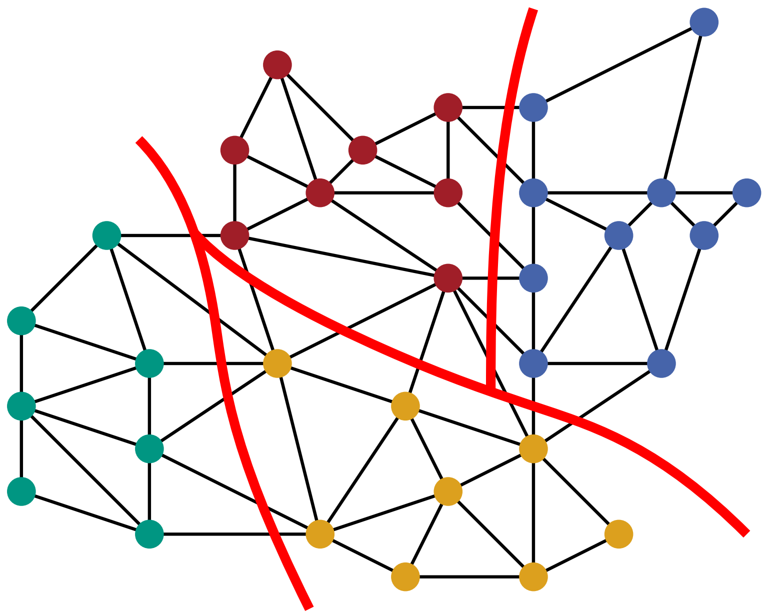
imbalance parameter

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

- **objective** function on edges is **minimized**

Common Objectives:

- Graphs:
  - cut: $\sum_{e \in \text{cut}} \omega(e) = 17$



ε-Balanced Graph and Hypergraph Partitioning

Partition (hyper)graph $G = (V, E, c : V \to \mathbb{R}_{>0}, \omega : E \to \mathbb{R}_{>0})$ into **k** disjoint blocks $V_1, \ldots, V_k$ s.t.

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

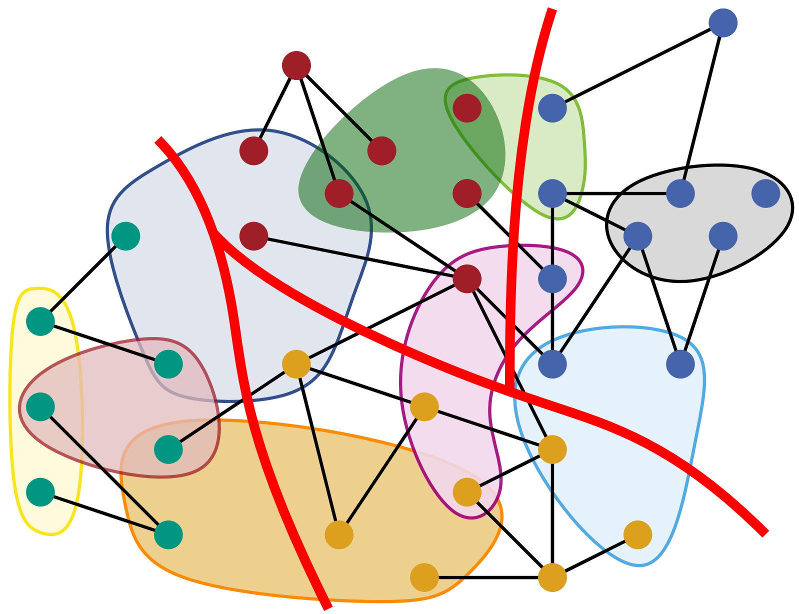
imbalance parameter

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

- **objective** function on edges is **minimized**

Common Objectives:

- Graphs:
  - cut: $\sum_{e \in \text{cut}} \omega(e) = 17$



ε-Balanced Graph and Hypergraph Partitioning

Karlsruhe Institute of Technology

Partition (hyper)graph $G = (V, E, c : V \to \mathbb{R}_{>0}, \omega : E \to \mathbb{R}_{>0})$ into **k** disjoint blocks $V_1, \ldots, V_k$ s.t.

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

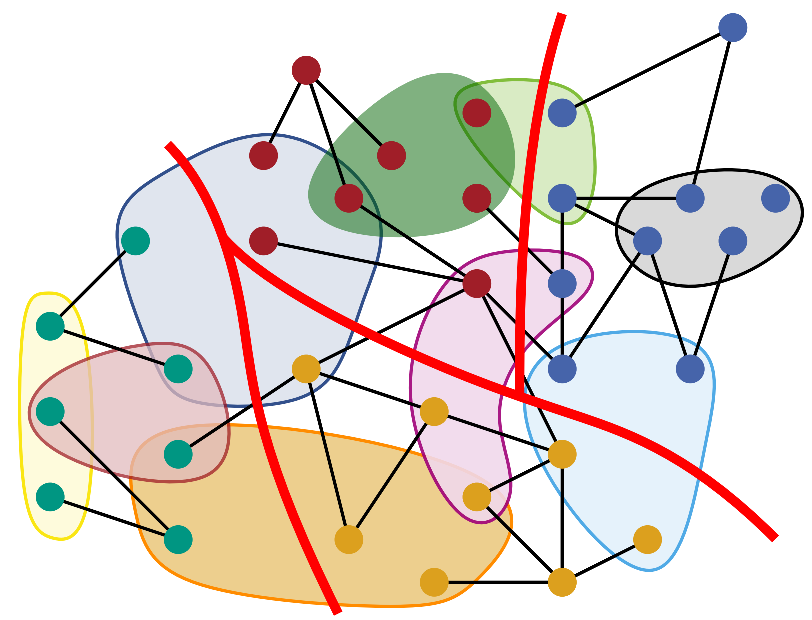
imbalance parameter

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

- **objective** function on edges is **minimized**

Common Objectives:

- Graphs:
  - cut: $\sum_{e \in \text{cut}} \omega(e) = 17$
- Hypergraphs:
  - cut: $\sum_{e \in \text{cut}} \omega(e)$



Institute of Theoretical Informatics
Algorithmics Group

ε-Balanced Graph and Hypergraph Partitioning

Partition (hyper)graph $G = (V, E, c : V \to \mathbb{R}_{>0}, \omega : E \to \mathbb{R}_{>0})$ into **k** disjoint blocks $V_1, \ldots, V_k$ s.t.

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

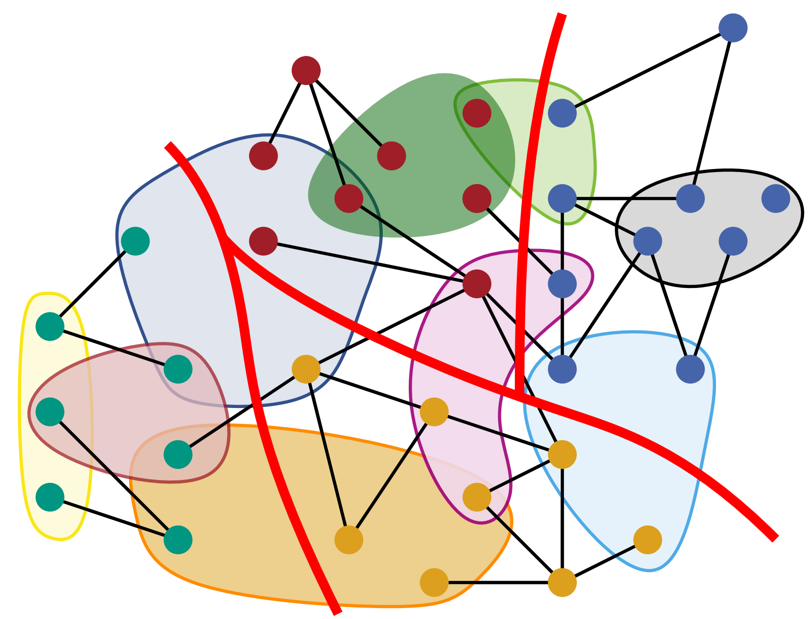
imbalance parameter

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

- **objective** function on edges is **minimized**

Common Objectives:

- Graphs:
  - cut: $\sum_{e \in \text{cut}} \omega(e) = 17$
- Hypergraphs:
  - cut: $\sum_{e \in \text{cut}} \omega(e) = 10$



Institute of Theoretical Informatics
Algorithmics Group

ε-Balanced Graph and Hypergraph Partitioning

Partition (hyper)graph $G = (V, E, c : V \to \mathbb{R}_{>0}, \omega : E \to \mathbb{R}_{>0})$ into **k** disjoint blocks $V_1, \ldots, V_k$ s.t.

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

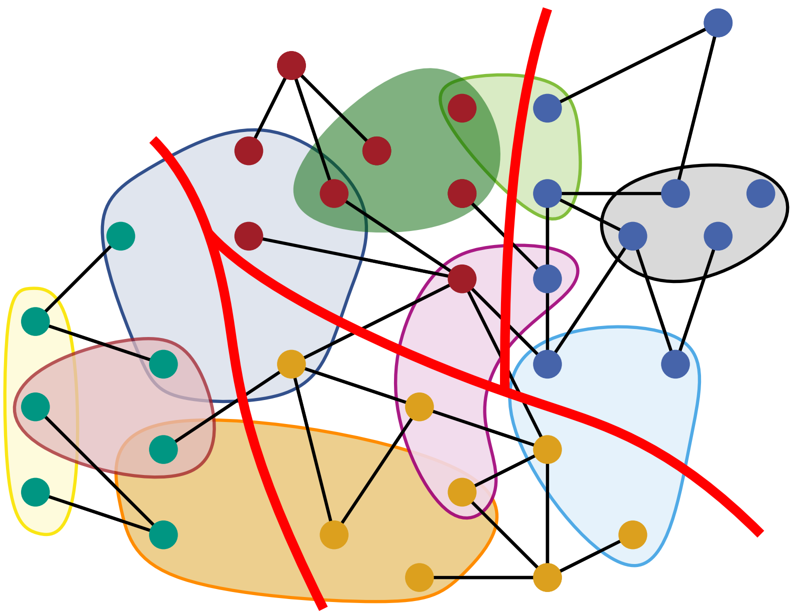
imbalance parameter

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

- **objective** function on edges is **minimized**

Common Objectives:

- Graphs:
  - cut: $\sum_{e \in \text{cut}} \omega(e) = \mathbf{17}$
- Hypergraphs:
  - cut: $\sum_{e \in \text{cut}} \omega(e) = \mathbf{10}$
  - connectivity: $\sum_{e \in \text{cut}} (\lambda - 1)\omega(e)$



ε-Balanced Graph and Hypergraph Partitioning

Partition (hyper)graph $G = (V, E, c : V \to \mathbb{R}_{>0}, \omega : E \to \mathbb{R}_{>0})$ into **k** disjoint blocks $V_1, \ldots, V_k$ s.t.

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

imbalance parameter

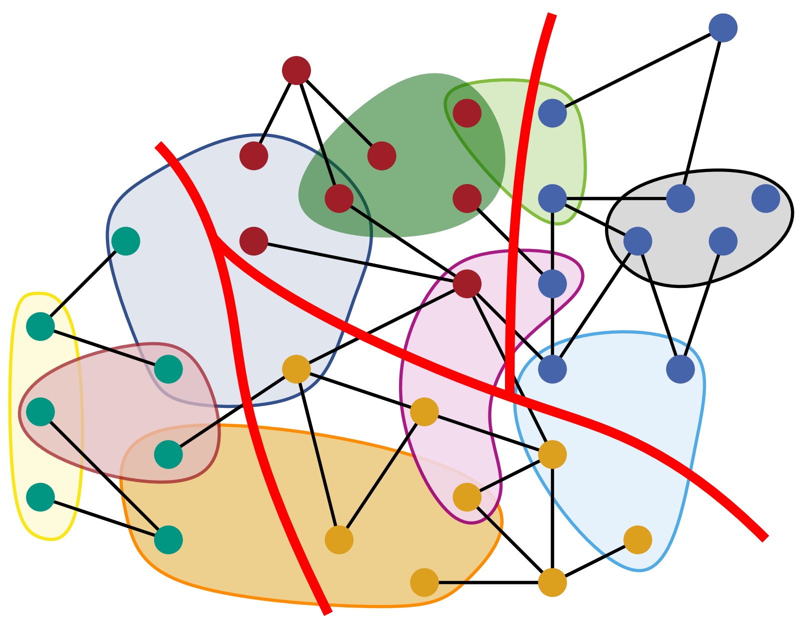
$$c(V_i) \leq (1 + \varepsilon) \left\lceil \frac{c(V)}{k} \right\rceil$$

- **objective** function on edges is **minimized**

Common Objectives:

- Graphs:
  - cut: $\sum_{e \in \text{cut}} \omega(e) = \mathbf{17}$
- Hypergraphs:
  - cut: $\sum_{e \in \text{cut}} \omega(e) = \mathbf{10}$
  - connectivity: $\sum_{e \in \text{cut}} (\lambda - 1)\omega(e)$

 # blocks connected by e



Institute of Theoretical Informatics
Algorithmics Group

ε-Balanced Graph and Hypergraph Partitioning

Partition (hyper)graph $G = (V, E, c : V \to \mathbb{R}_{>0}, \omega : E \to \mathbb{R}_{>0})$ into **k** disjoint blocks $V_1, \ldots, V_k$ s.t.

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

imbalance parameter

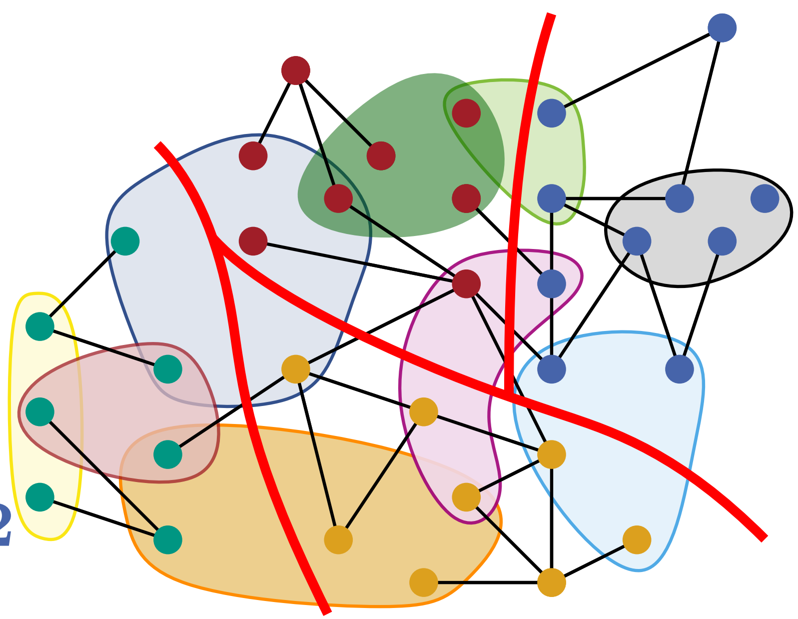
$$c(V_i) \leq (1 + \varepsilon) \left\lceil \frac{c(V)}{k} \right\rceil$$

- **objective** function on edges is **minimized**

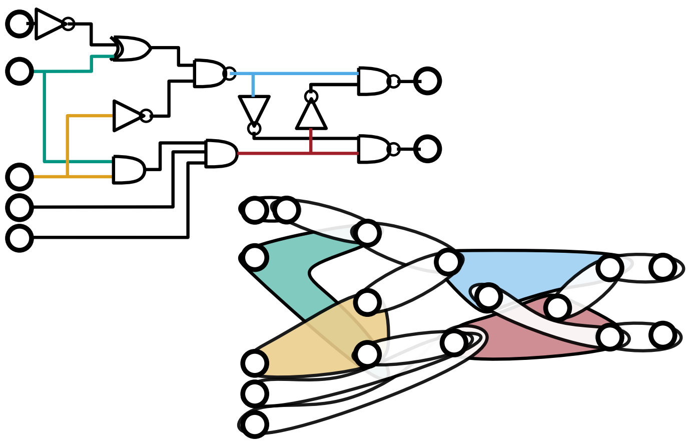
Common Objectives:

- Graphs:
  - cut: $\sum_{e \in \text{cut}} \omega(e) = \mathbf{17}$
- Hypergraphs:
  - cut: $\sum_{e \in \text{cut}} \omega(e) = \mathbf{10}$
  - connectivity: $\sum_{e \in \text{cut}} (\lambda - 1)\omega(e) = \mathbf{12}$

 # blocks connected by e



Applications



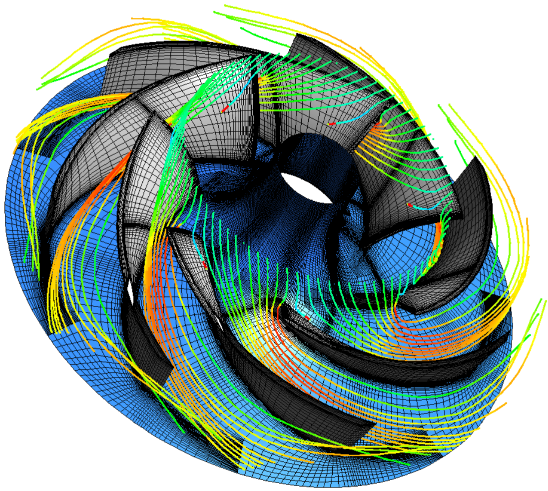
VLSI Design

Warehouse Optimization

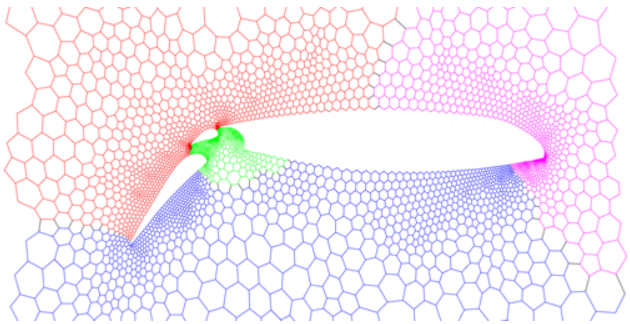
[Martin Grandjean, via Wikimedia Commons]

Complex Networks

Route Planning



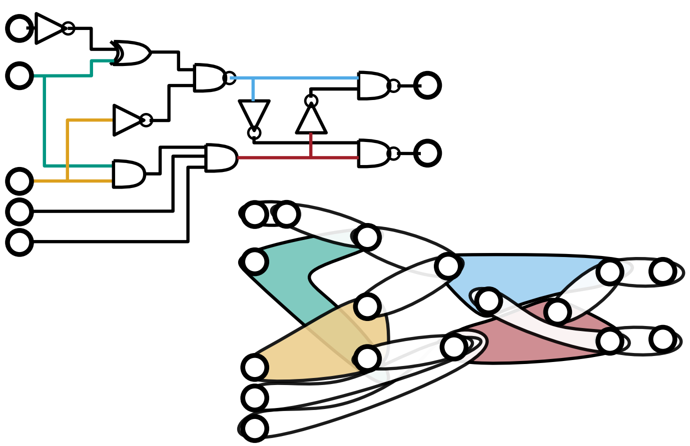
Simulation



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

Scientific Computing

Applications



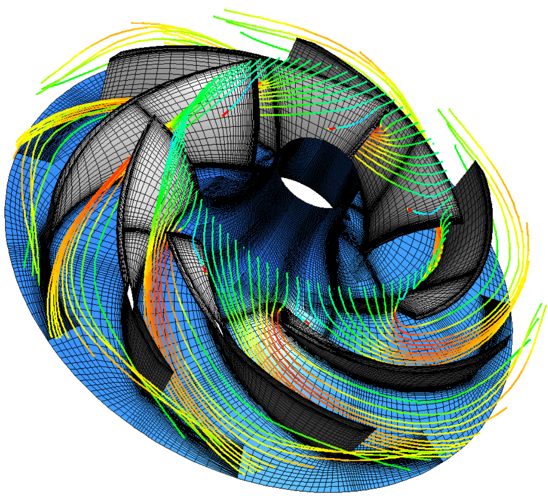
VLSI Design

Warehouse Optimization

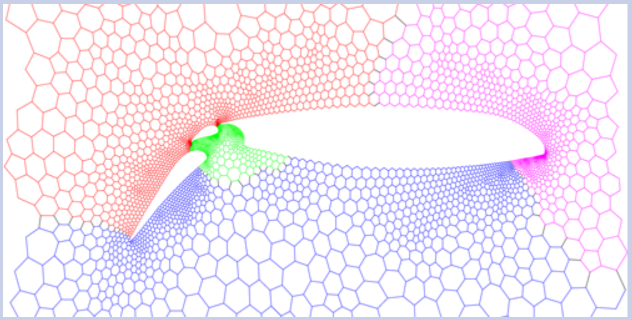
[Martin Grandjean, via Wikimedia Commons]

Complex Networks

Route Planning



Simulation

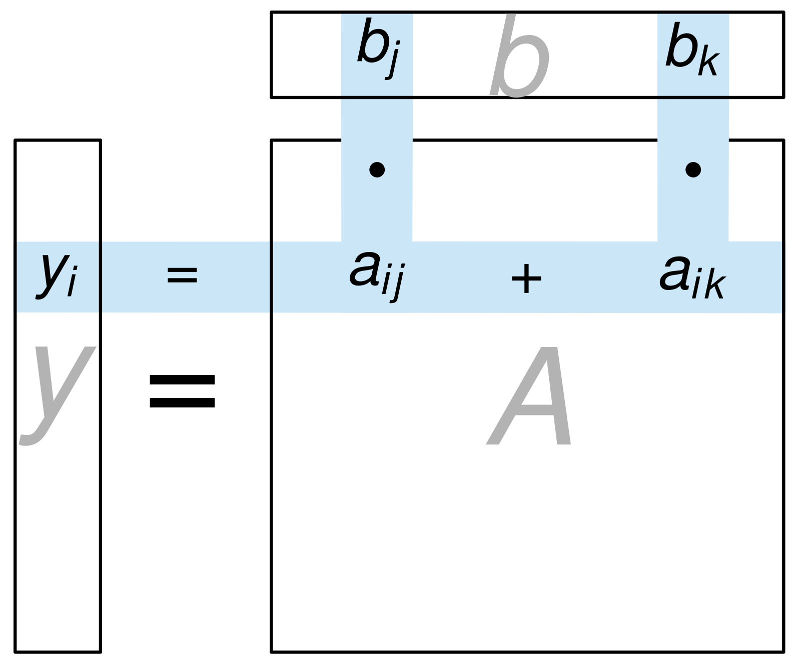


$$\mathbf{R}^{n \times n} \ni Ax = b \in \mathbf{R}^n$$

Scientific Computing

Parallel Sparse-Matrix Vector Product (SpM×V)

$$y = A\,b$$

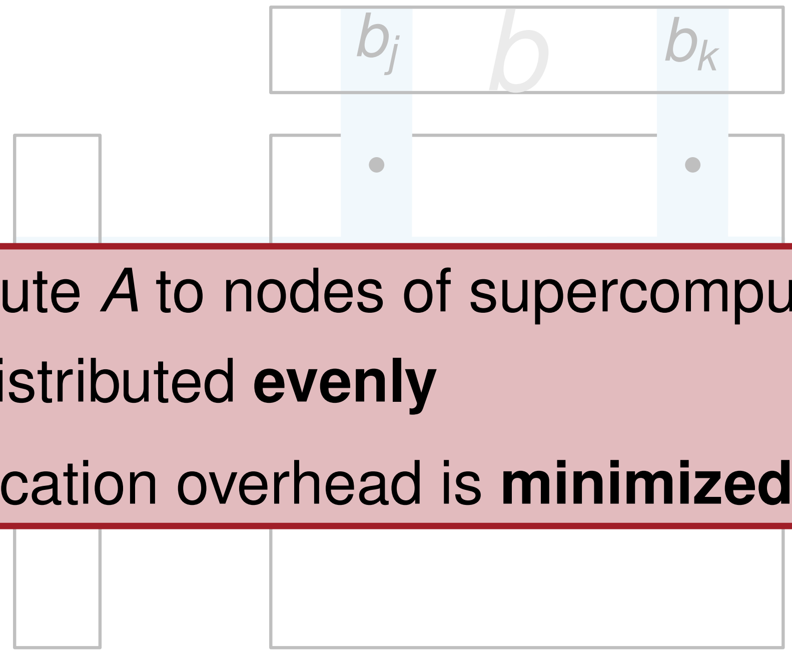


Setting:

- repeated SpM×V on supercomputer
- A is large $\Rightarrow$ distribute on multiple nodes
- symmetric partitioning $\Rightarrow$ y & b divided conformally with A

Parallel Sparse-Matrix Vector Product (SpM×V)

$$y = A\,b$$



Task: distribute *A* to nodes of supercomputer such that

- work is distributed **evenly**
- communication overhead is **minimized**

Setting:

- repeated SpM×V on supercomputer
- *A* is large ⇒ distribute on multiple nodes
- symmetric partitioning ⇒ *y* & *b* divided conformally with *A*

Institute of Theoretical Informatics
Algorithmics Group

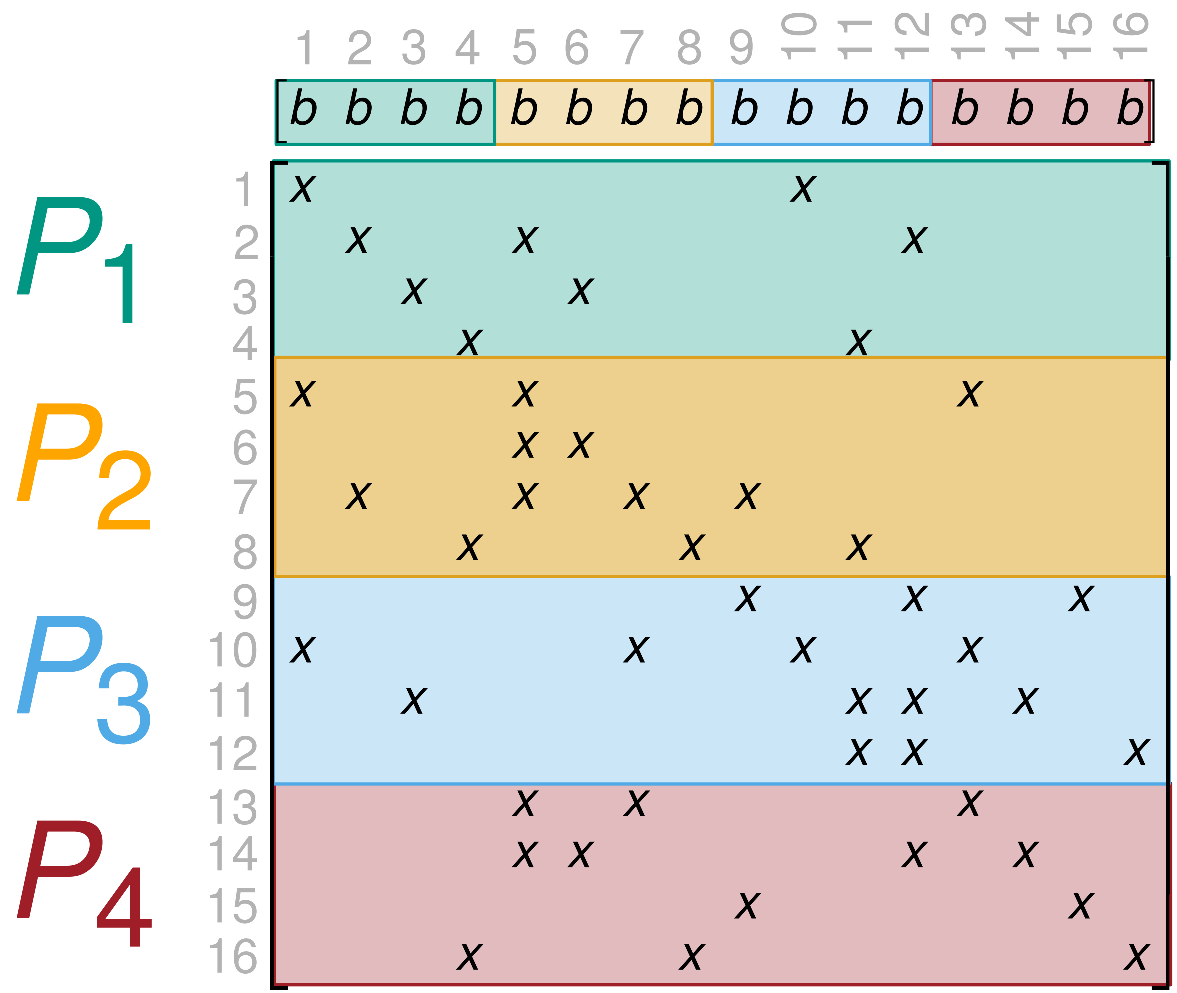
Naive Approach: Rowwise Decomposition

$A \in \mathbf{R}^{16 \times 16}$

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
	b	*b*	*b*	*b*	*b*	*b*	*b*	*b*	*b*	*b*	*b*	*b*	*b*	*b*	*b*	*b*
1	*x*									*x*						
2		*x*			*x*							*x*				
3			*x*			*x*										
4				*x*							*x*					
5	*x*				*x*								*x*			
6					*x*	*x*										
7		*x*			*x*		*x*		*x*							
8				*x*				*x*			*x*					
9									*x*			*x*			*x*	
10	*x*						*x*			*x*			*x*			
11			*x*								*x*	*x*		*x*		
12											*x*	*x*				*x*
13					*x*		*x*						*x*			
14					*x*	*x*						*x*		*x*		
15									*x*						*x*	
16				*x*				*x*								*x*

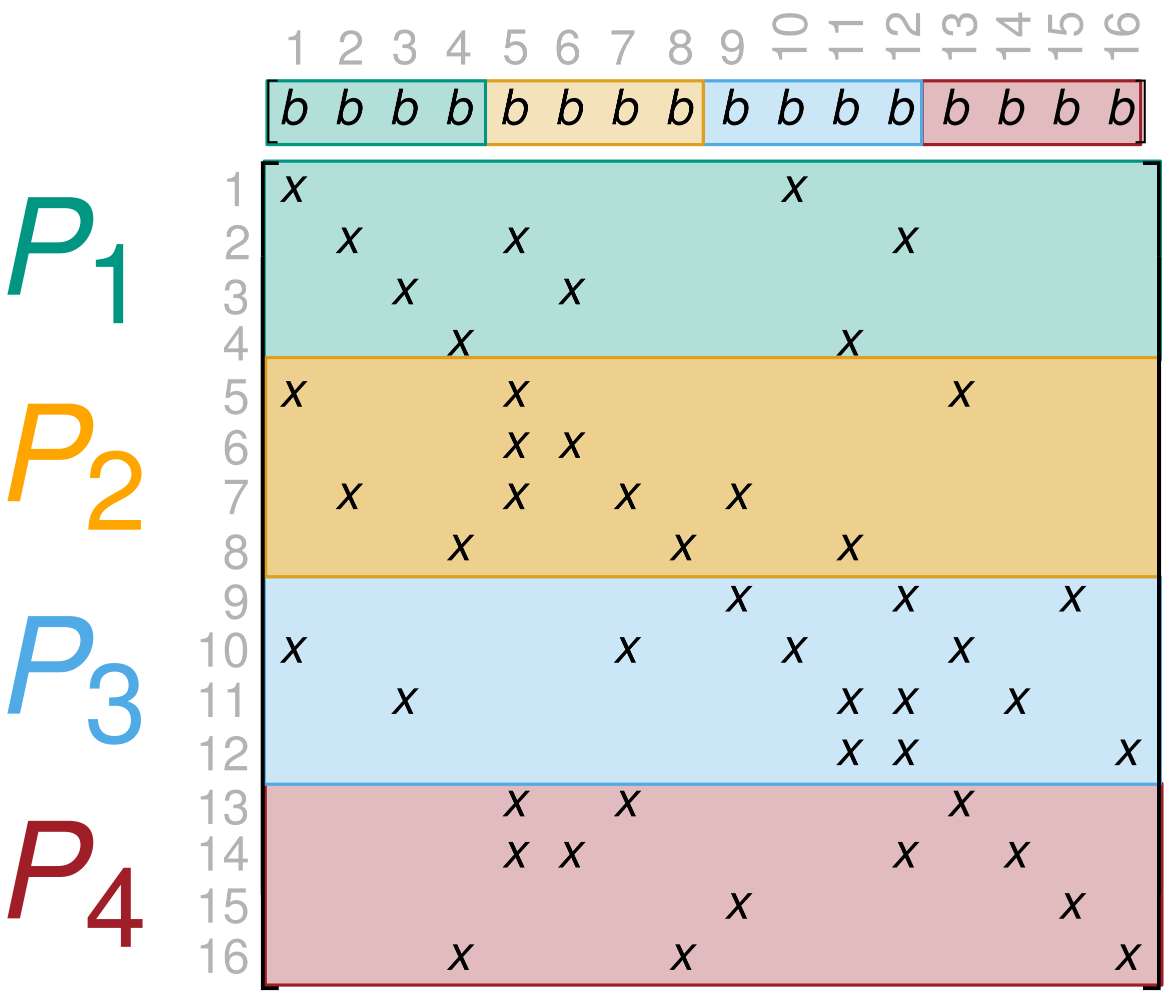
Naive Approach: Rowwise Decomposition

$A \in \mathbb{R}^{16 \times 16}$



Naive Approach: Rowwise Decomposition

$A \in \mathbb{R}^{16 \times 16}$



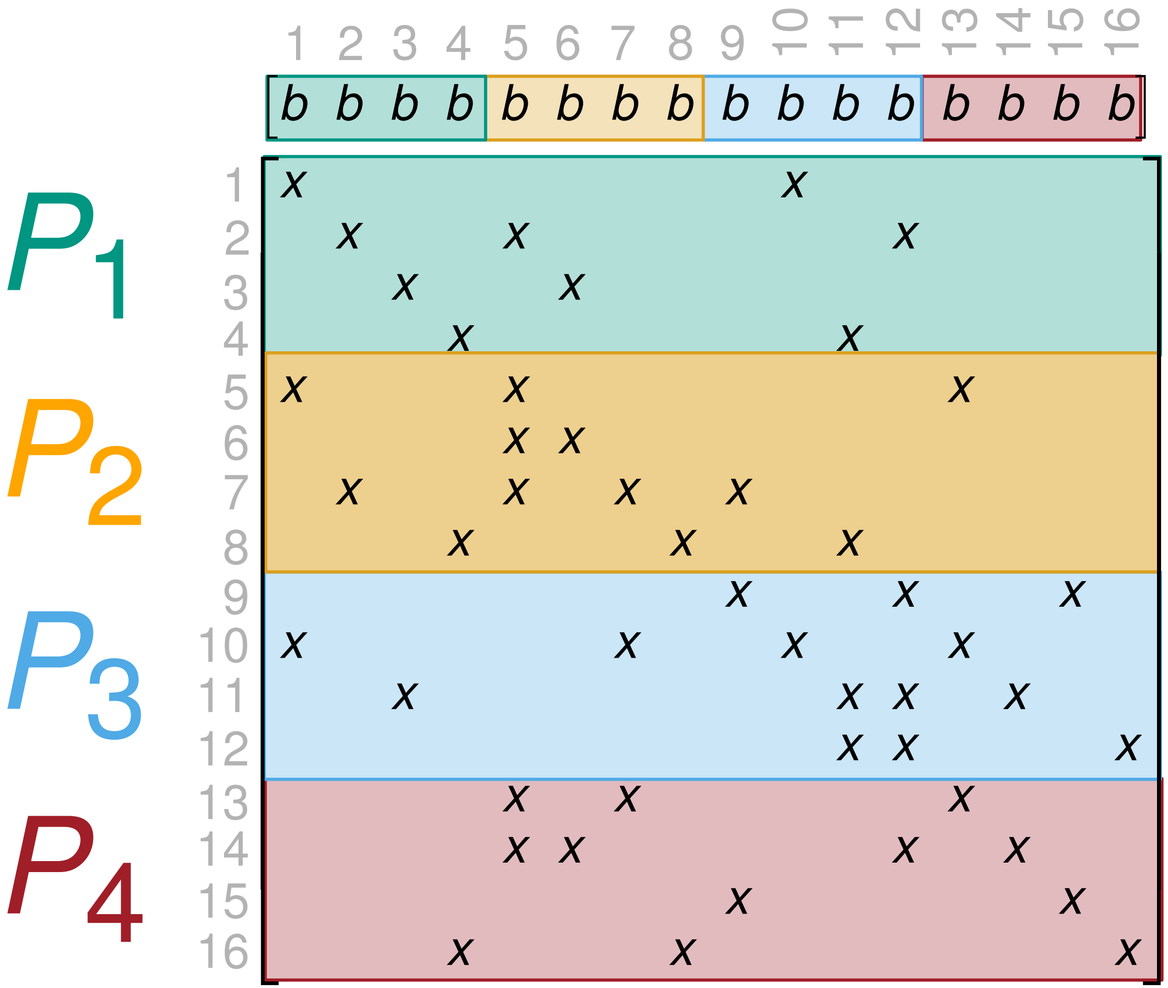
Load Balancing?

- $P_1 \Rightarrow 9$
- $P_2 \Rightarrow 12$
- $P_3 \Rightarrow 14$
- $P_4 \Rightarrow 12$

Institute of Theoretical Informatics
Algorithmics Group

Naive Approach: Rowwise Decomposition

$A \in \mathbb{R}^{16 \times 16}$



Load Balancing?

$\Rightarrow$ 9

$\Rightarrow$ 12

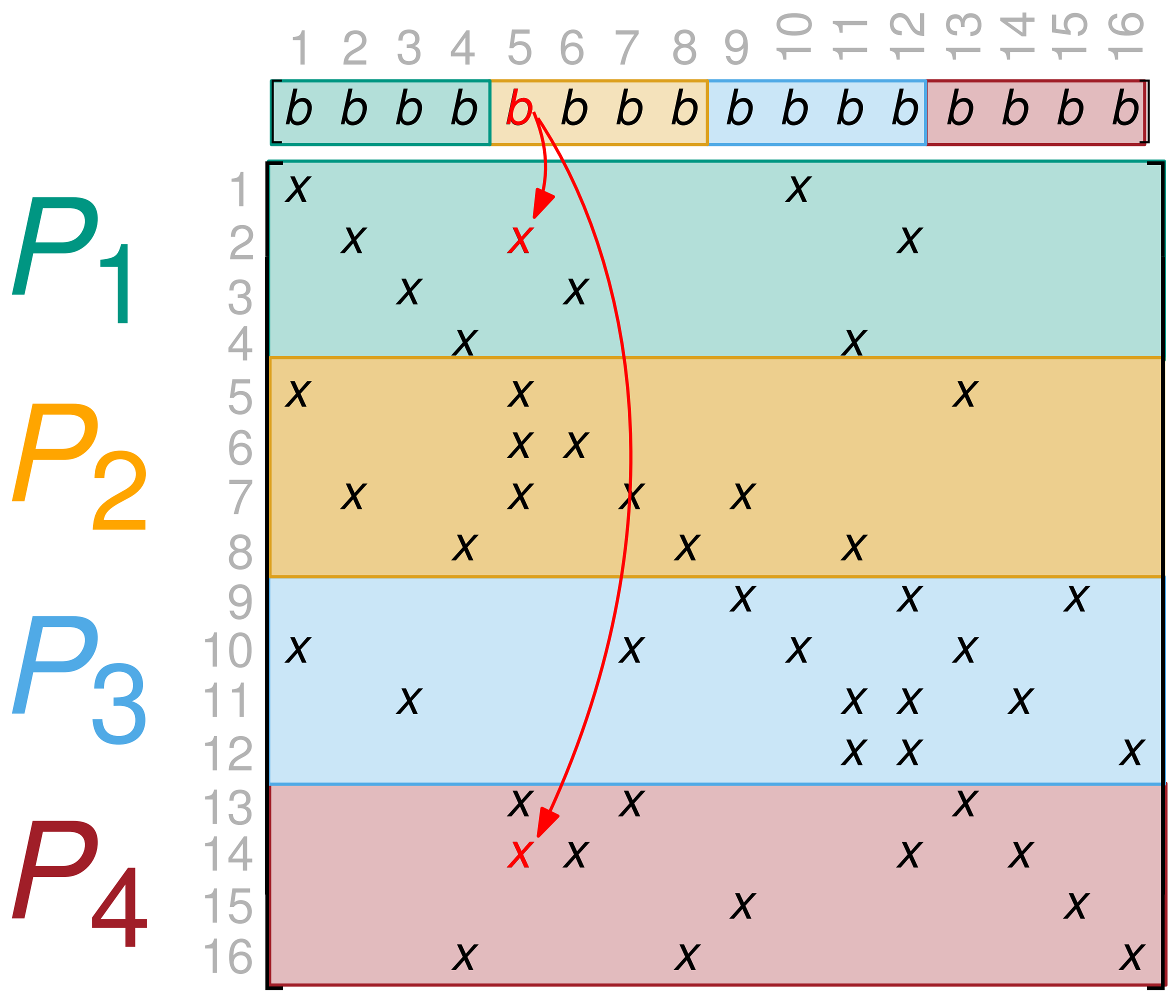
$\Rightarrow$ 14

$\Rightarrow$ 12

Commuication Volume?

Naive Approach: Rowwise Decomposition

$A \in \mathbb{R}^{16\times 16}$



Load Balancing?

$\Rightarrow$ 9

$\Rightarrow$ 12

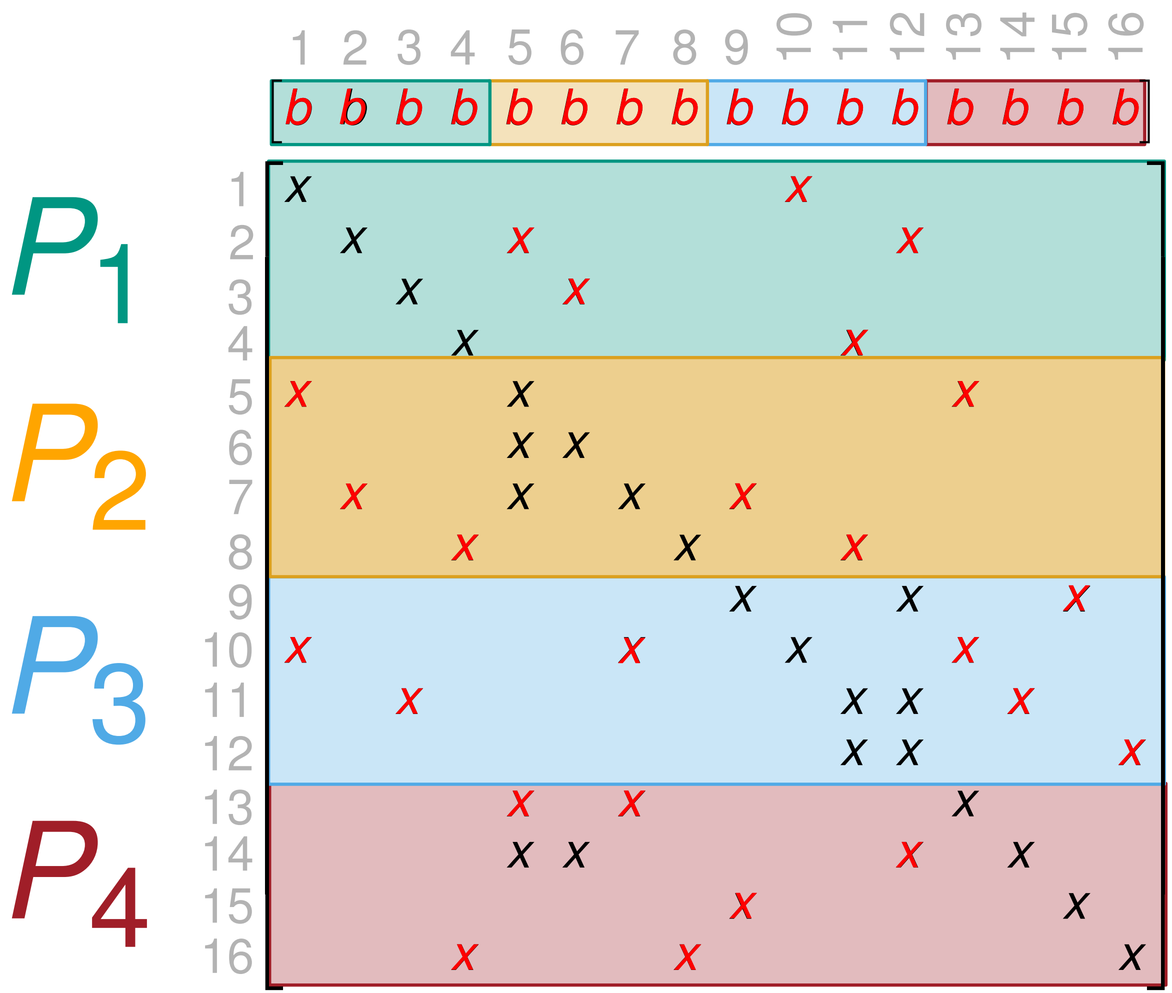
$\Rightarrow$ 14

$\Rightarrow$ 12

Commuication Volume?

Naive Approach: Rowwise Decomposition

$A \in \mathbb{R}^{16 \times 16}$



Load Balancing?

$\Rightarrow$ 9

$\Rightarrow$ 12

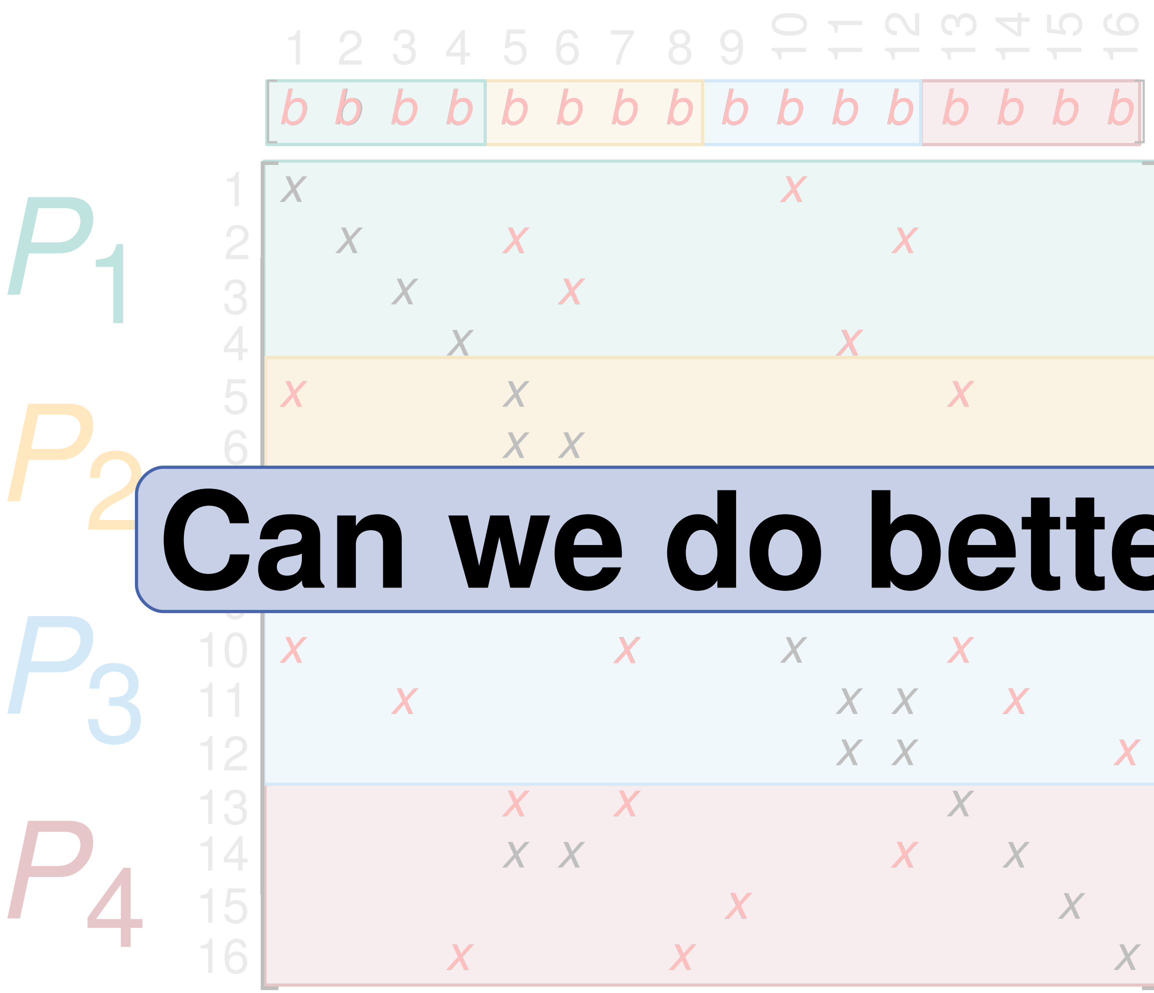
$\Rightarrow$ 14

$\Rightarrow$ 12

Commuication Volume? $\Rightarrow$ 24 entries!

Naive Approach: Rowwise Decomposition

$A \in \mathbb{R}^{16\times 16}$



Load Balancing?

$\Rightarrow$ 9

$\Rightarrow$ 12

$\Rightarrow$ 14

$\Rightarrow$ 12

Can we do better?

Commuication Volume? $\Rightarrow$ 24 entries!

Institute of Theoretical Informatics
Algorithmics Group

From SpM×V to Hypergraph Partitioning

$A \in \mathbb{R}^{16\times16} \Rightarrow H = (V_R, E_C)$

- One vertex per row:
  $\Rightarrow V_R = \{v_1, v_2, \ldots, v_{16}\}$
- One hyperedge per column:
  $\Rightarrow E_C = \{e_1, e_2, \ldots, e_{16}\}$

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
	b	b	b	b	b	b	b	b	b	b	b	b	b	b	b	b
1	x									x						
2		x			x							x				
3			x			x										
4				x							x					
5	x				x								x			
6					x	x										
7		x			x		x		x							
8				x				x			x					
9									x			x			x	
10	x						x			x			x			
11			x								x	x		x		
12											x	x				x
13					x		x						x			
14					x	x						x		x		
15									x						x	
16				x				x								x

Institute of Theoretical Informatics
Algorithmics Group

From SpM×V to Hypergraph Partitioning

$A \in \mathbb{R}^{16\times16} \Rightarrow H = (V_R, E_C)$

- One vertex per row:

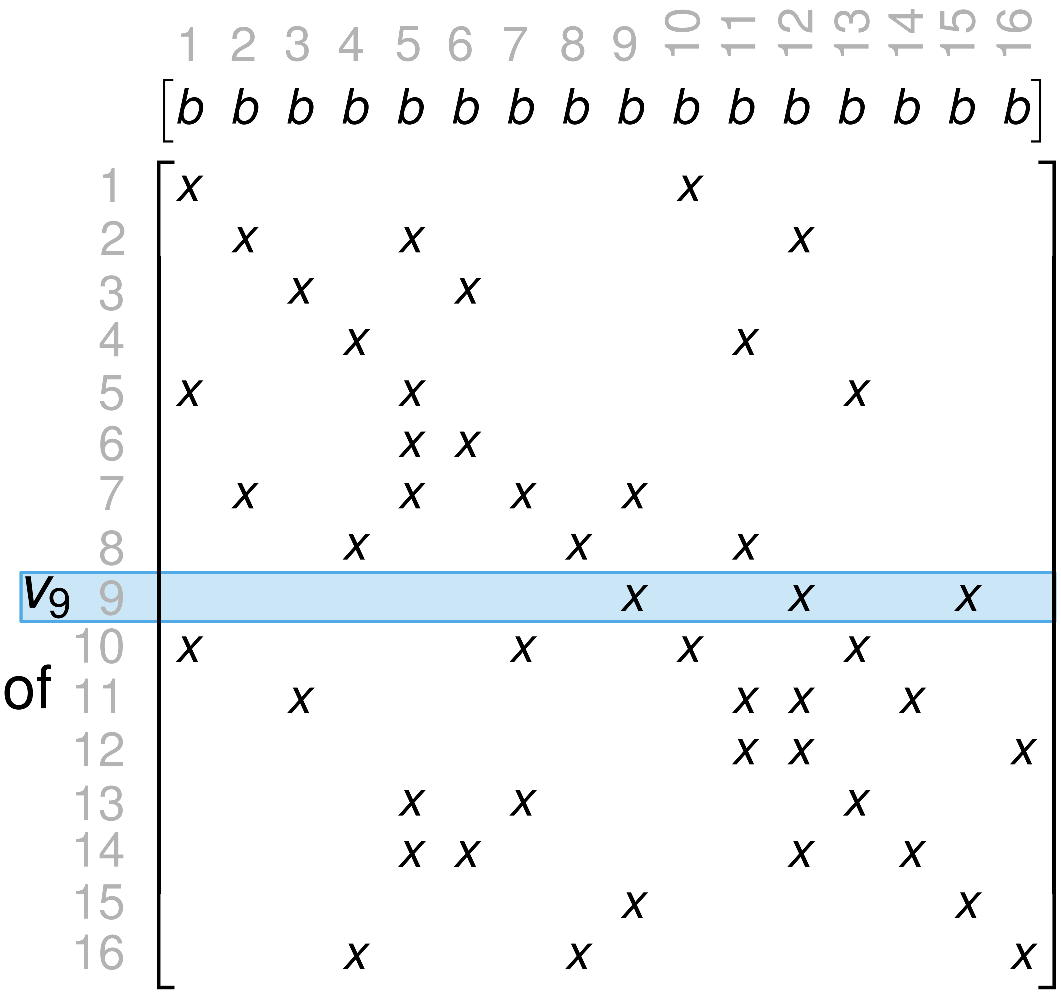
  $\Rightarrow V_R = \{v_1, v_2, \ldots, v_{16}\}$

- One hyperedge per column:

  $\Rightarrow E_C = \{e_1, e_2, \ldots, e_{16}\}$

$v_i \in V_R$:

- task to compute inner product of row i with b
- $\Rightarrow c(v_i) :=$ # nonzeros

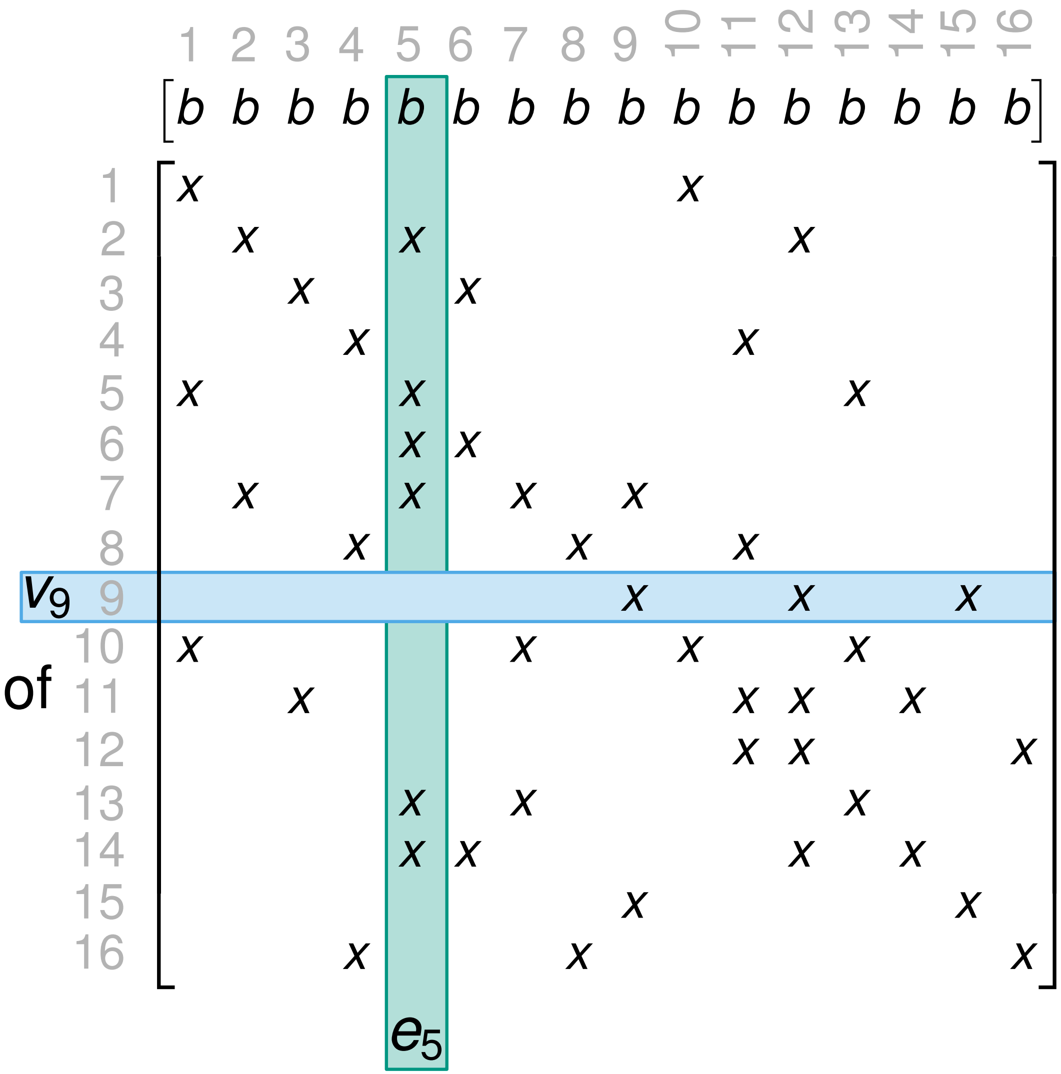


Institute of Theoretical Informatics
Algorithmics Group

From SpM×V to Hypergraph Partitioning

$A \in \mathbb{R}^{16\times 16} \Rightarrow H = (V_R, E_C)$

- One vertex per row:
  $\Rightarrow V_R = \{v_1, v_2, \ldots, v_{16}\}$
- One hyperedge per column:
  $\Rightarrow E_C = \{e_1, e_2, \ldots, e_{16}\}$



$v_i \in V_R$:

- task to compute inner product of row i with b
- $\Rightarrow c(v_i) := \#$ nonzeros

$e_j \in E_C$: set of vertices that need b_j

Institute of Theoretical Informatics
Algorithmics Group

From SpM×V to Hypergraph Partitioning

$A \in \mathbb{R}^{16\times16} \Rightarrow H = (V_R, E_C)$

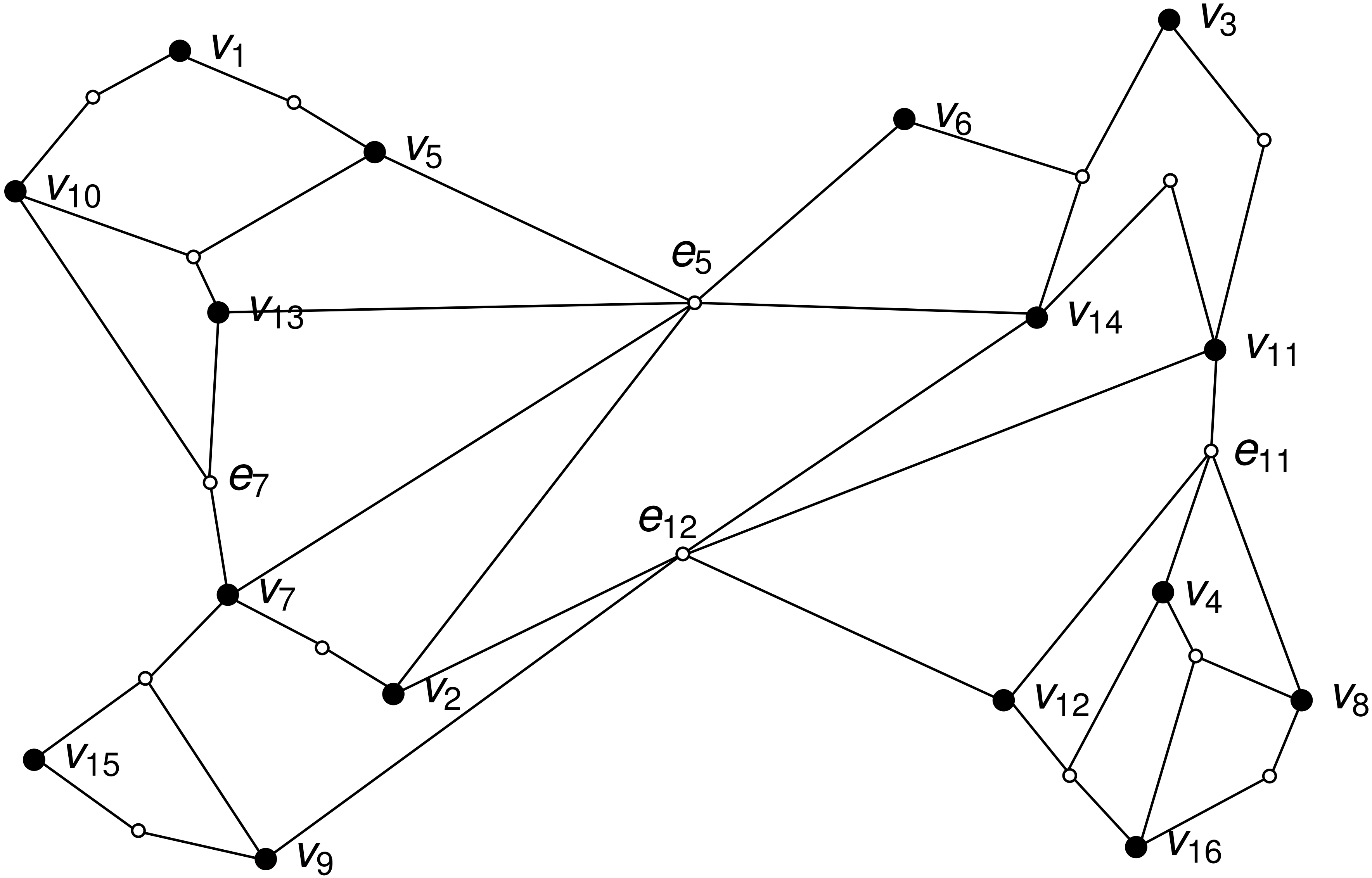
- One vertex per row:
  $\Rightarrow V_R = \{v_1, v_2, \ldots, v_{16}\}$
- One hyperedge per column:
  $\Rightarrow E_C = \{e_1, e_2, \ldots, e_{16}\}$

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
	b	b	b	b	b	b	b	b	b	b	b	b	b	b	b	b
1	x									x						
2		x			x							x				
3			x			x										
4				x							x					
5	x				x								x			
6					x	x										
9															x	
12											x	x				x
13					x		x						x			
14					x	x						x		x		
15									x						x	
16				x				x								x

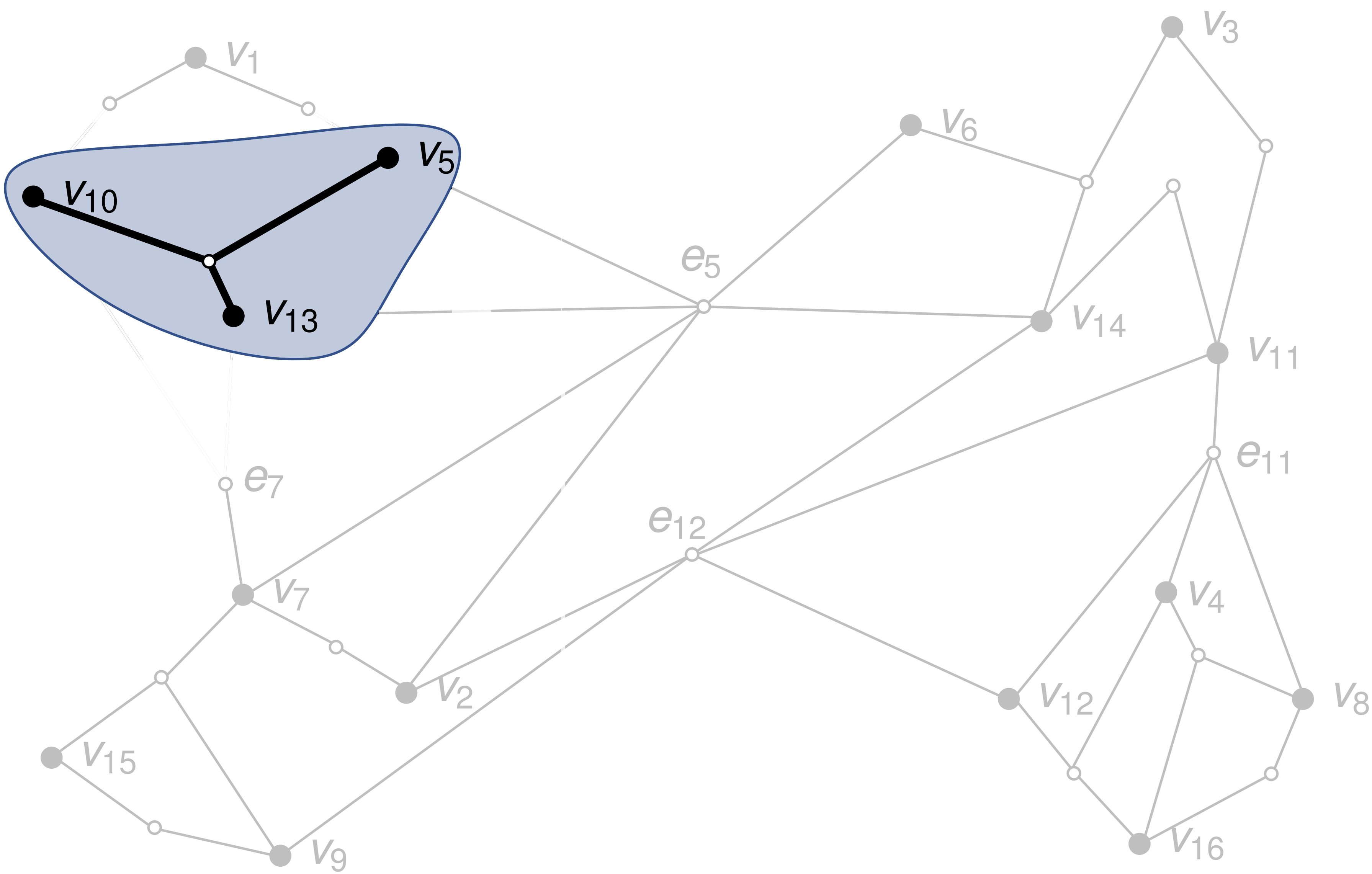
Solution: ε-balanced partition of H

- balanced partition $\rightsquigarrow$ computational load balance
- small $(\lambda - 1)$-cutsize $\rightsquigarrow$ minimizing communication volume

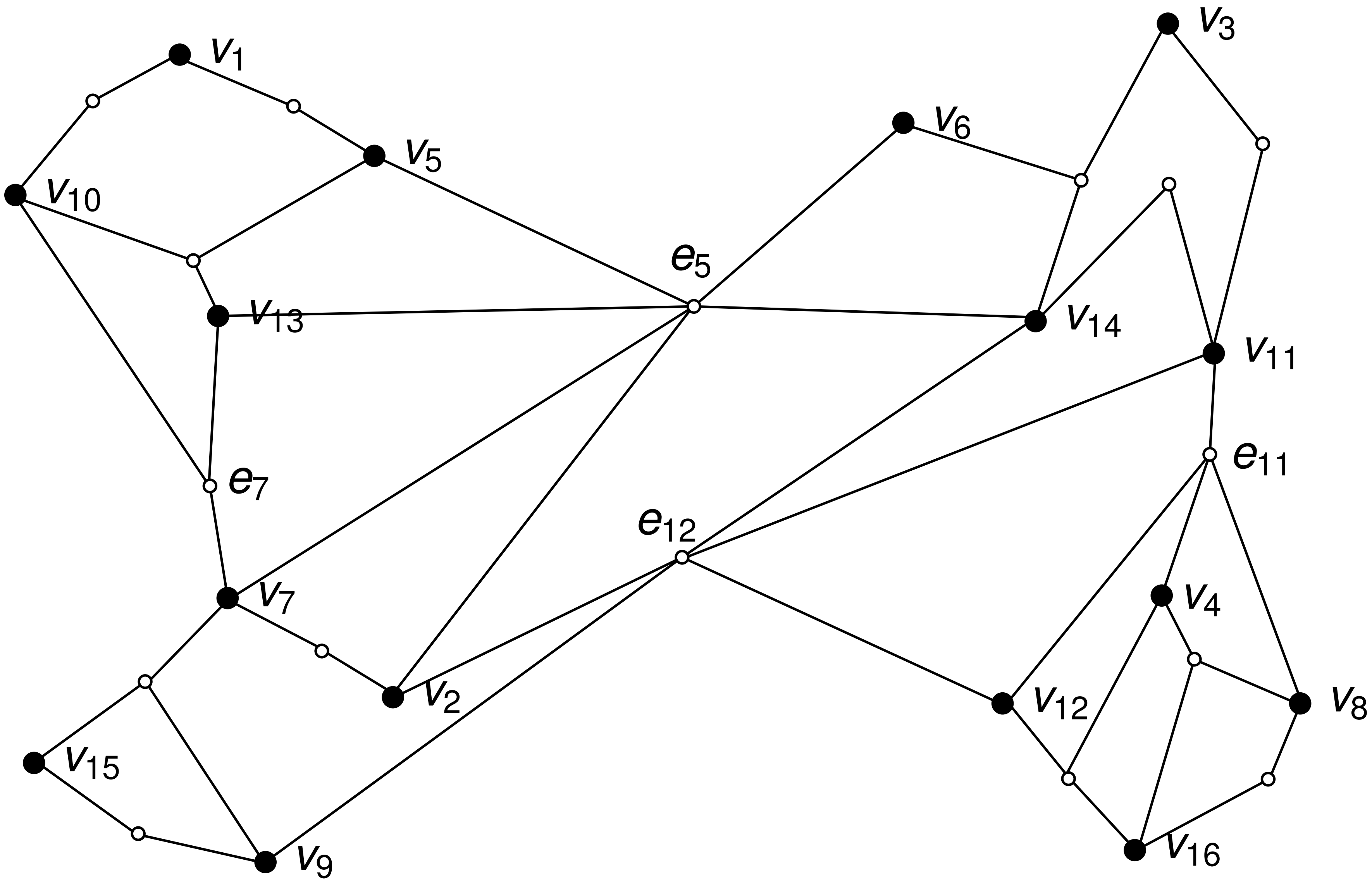
From SpM×V to Hypergraph Partitioning



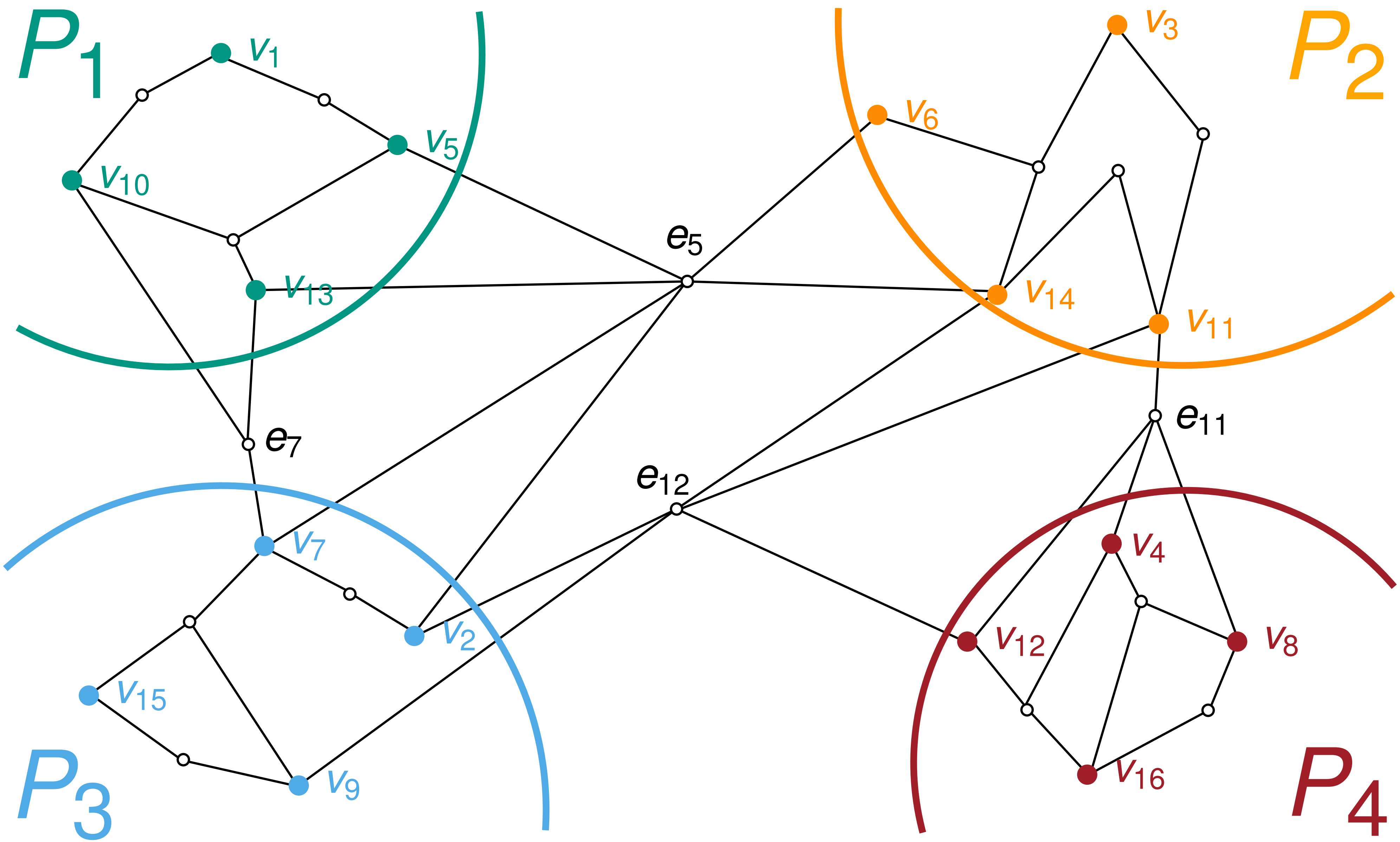
From SpM×V to Hypergraph Partitioning



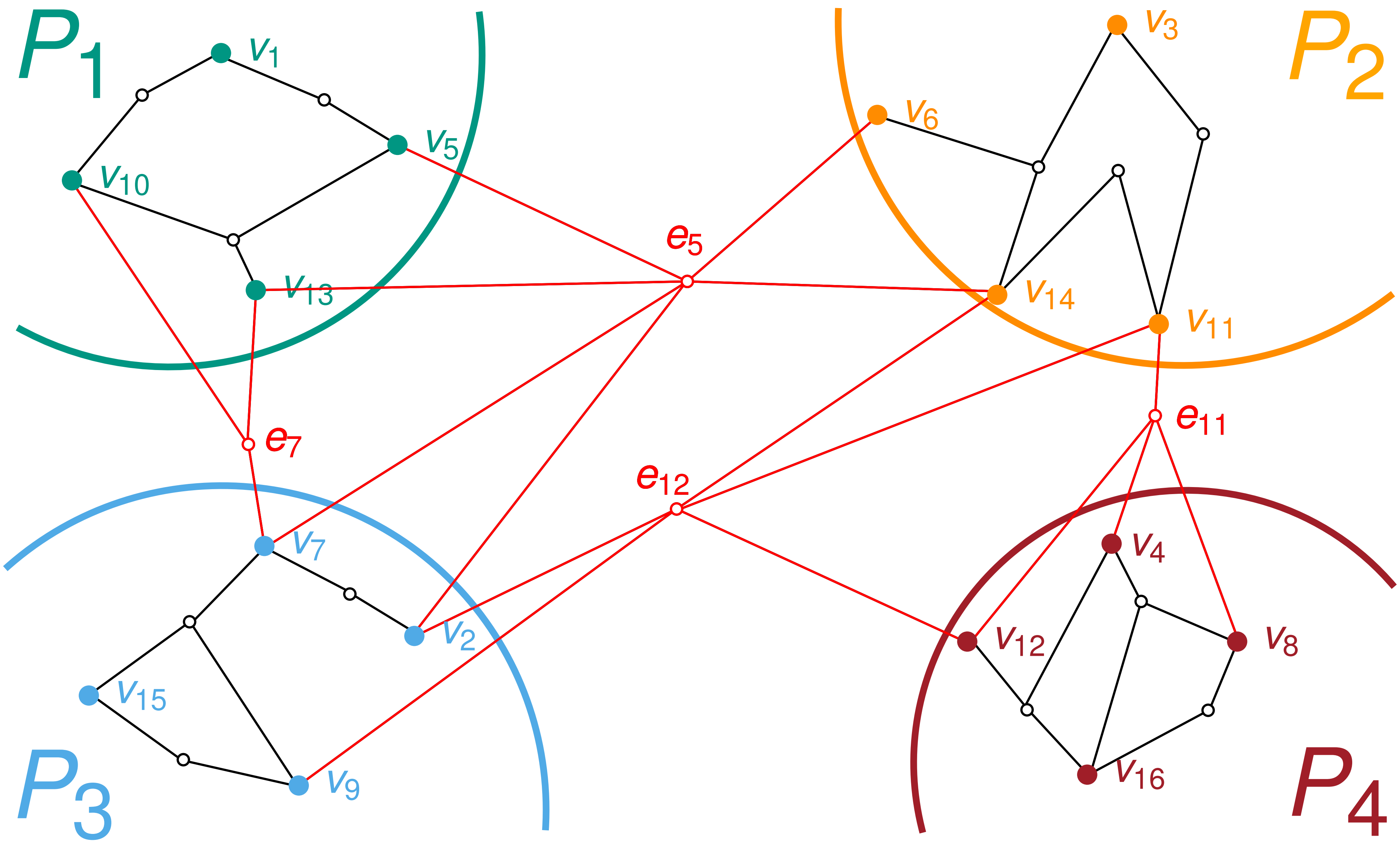
From SpM×V to Hypergraph Partitioning



From SpM×V to Hypergraph Partitioning

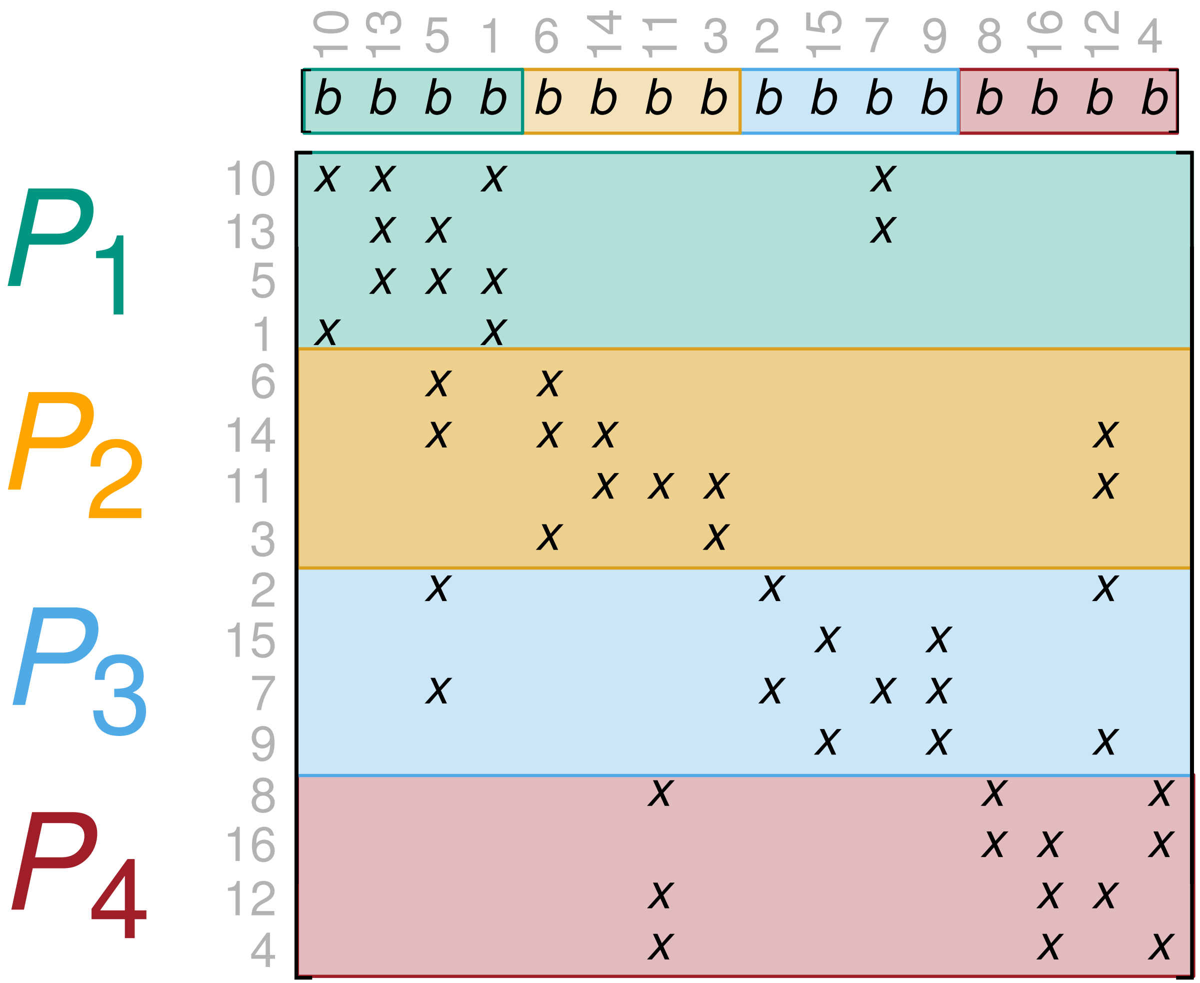


From SpM×V to Hypergraph Partitioning

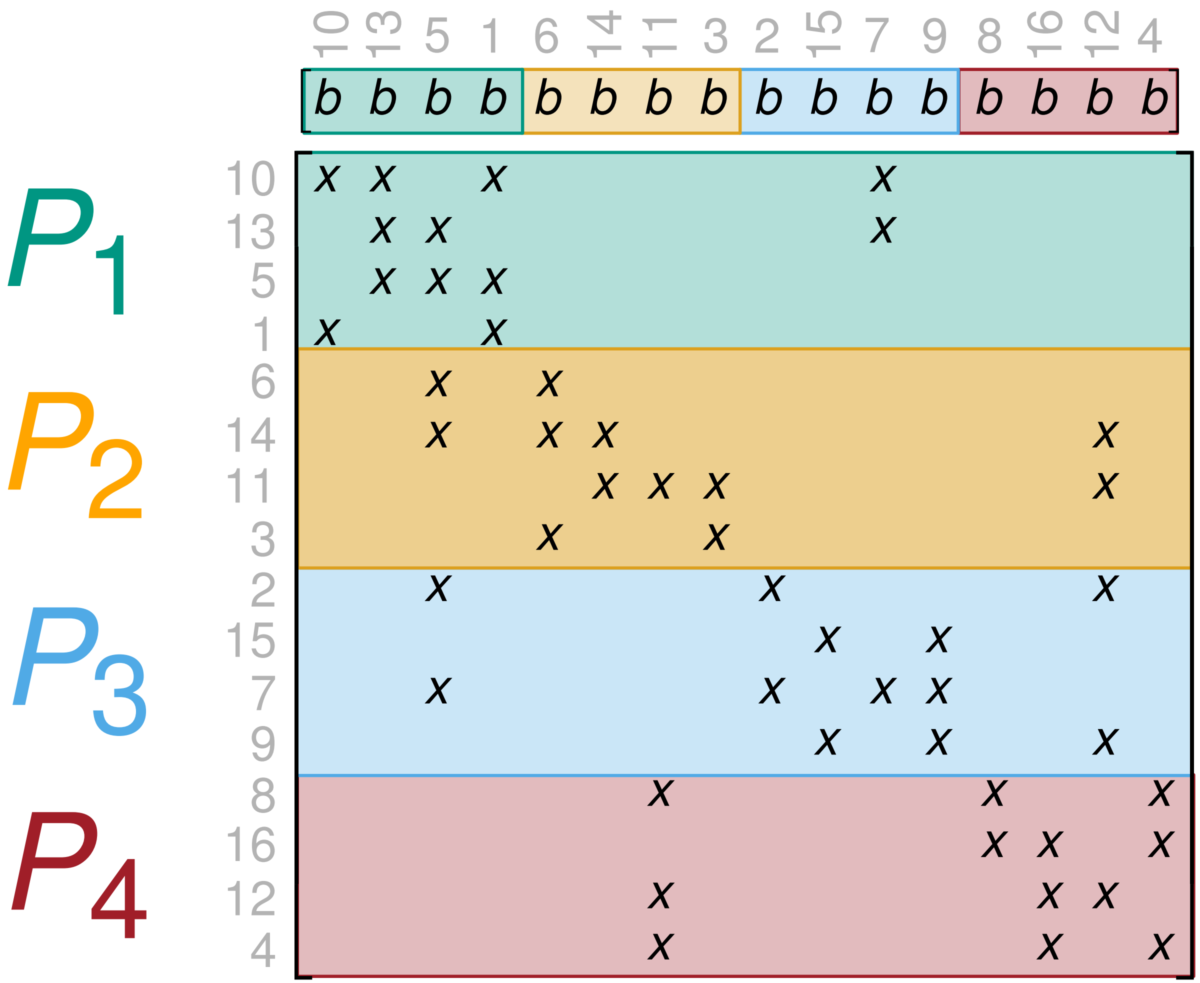


Institute of Theoretical Informatics
Algorithmics Group

From Hypergraph Partitioning to SpM×V



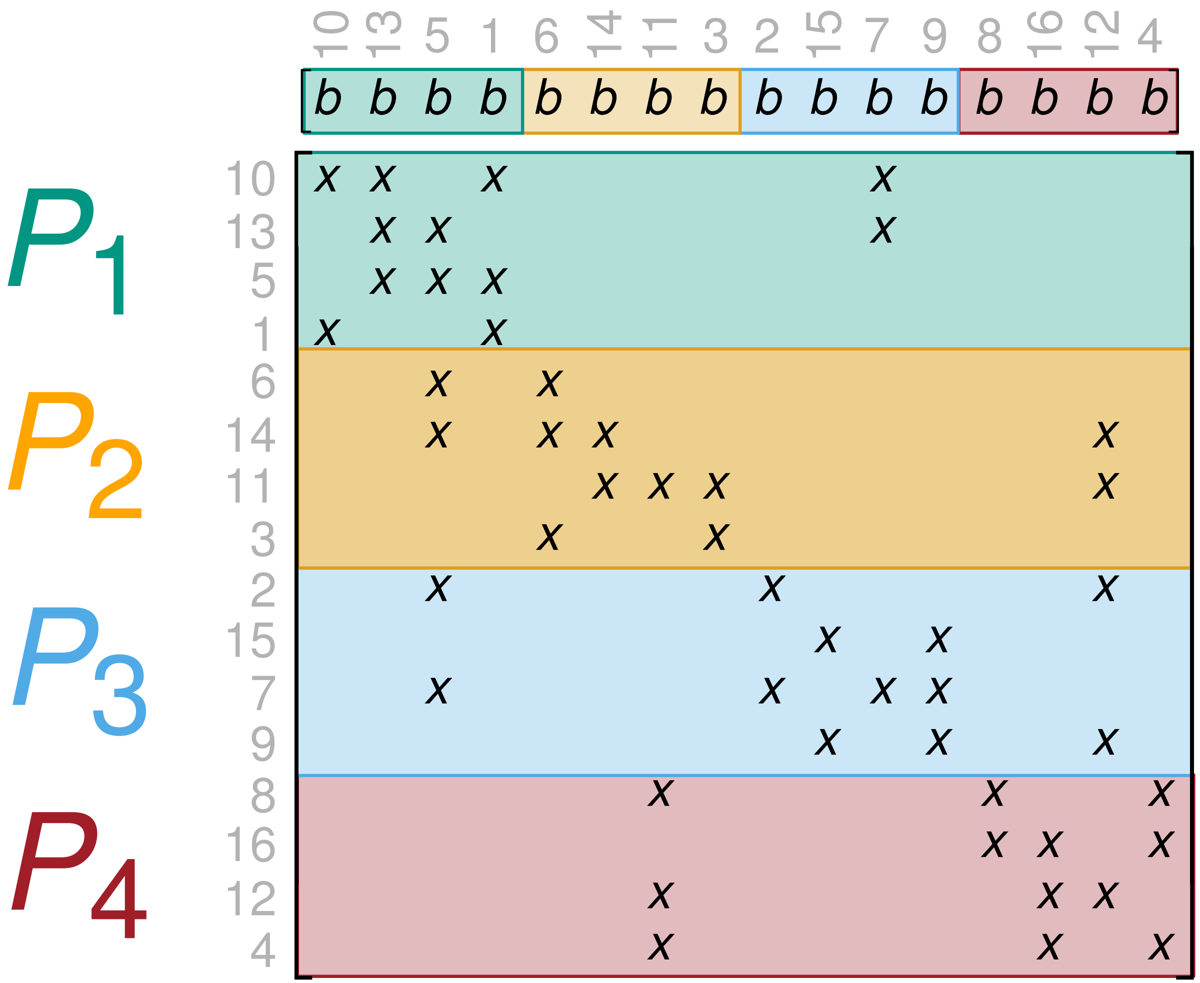
From Hypergraph Partitioning to SpM$\times$V



Load Balancing?

Institute of Theoretical Informatics
Algorithmics Group

From Hypergraph Partitioning to SpM$\times$V



Load Balancing?

$\Rightarrow$ 12

$\Rightarrow$ 12

$\Rightarrow$ 12

$\Rightarrow$ 12

Institute of Theoretical Informatics
Algorithmics Group

From Hypergraph Partitioning to SpM$\times$V

Where are the cut-hyperedges?

		10	13	5	1	6	14	11	3	2	15	7	9	8	16	12	4
		b	*b*	*b*	*b*	*b*	*b*	*b*	*b*	*b*	*b*	*b*	*b*	*b*	*b*	*b*	*b*
P_1	10	*x*	*x*		*x*							*x*					
	13		*x*	*x*								*x*					
	5		*x*	*x*	*x*												
	1	*x*			*x*												
P_2	6			*x*		*x*											
	14			*x*		*x*	*x*									*x*	
	11						*x*	*x*	*x*							*x*	
	3					*x*			*x*								
P_3	2			*x*						*x*						*x*	
	15										*x*		*x*				
	7			*x*						*x*		*x*	*x*				
	9										*x*		*x*			*x*	
P_4	8							*x*						*x*			*x*
	16													*x*	*x*		*x*
	12							*x*							*x*	*x*	
	4							*x*							*x*		*x*

Load Balancing?

$P_1 \Rightarrow 12$

$P_2 \Rightarrow 12$

$P_3 \Rightarrow 12$

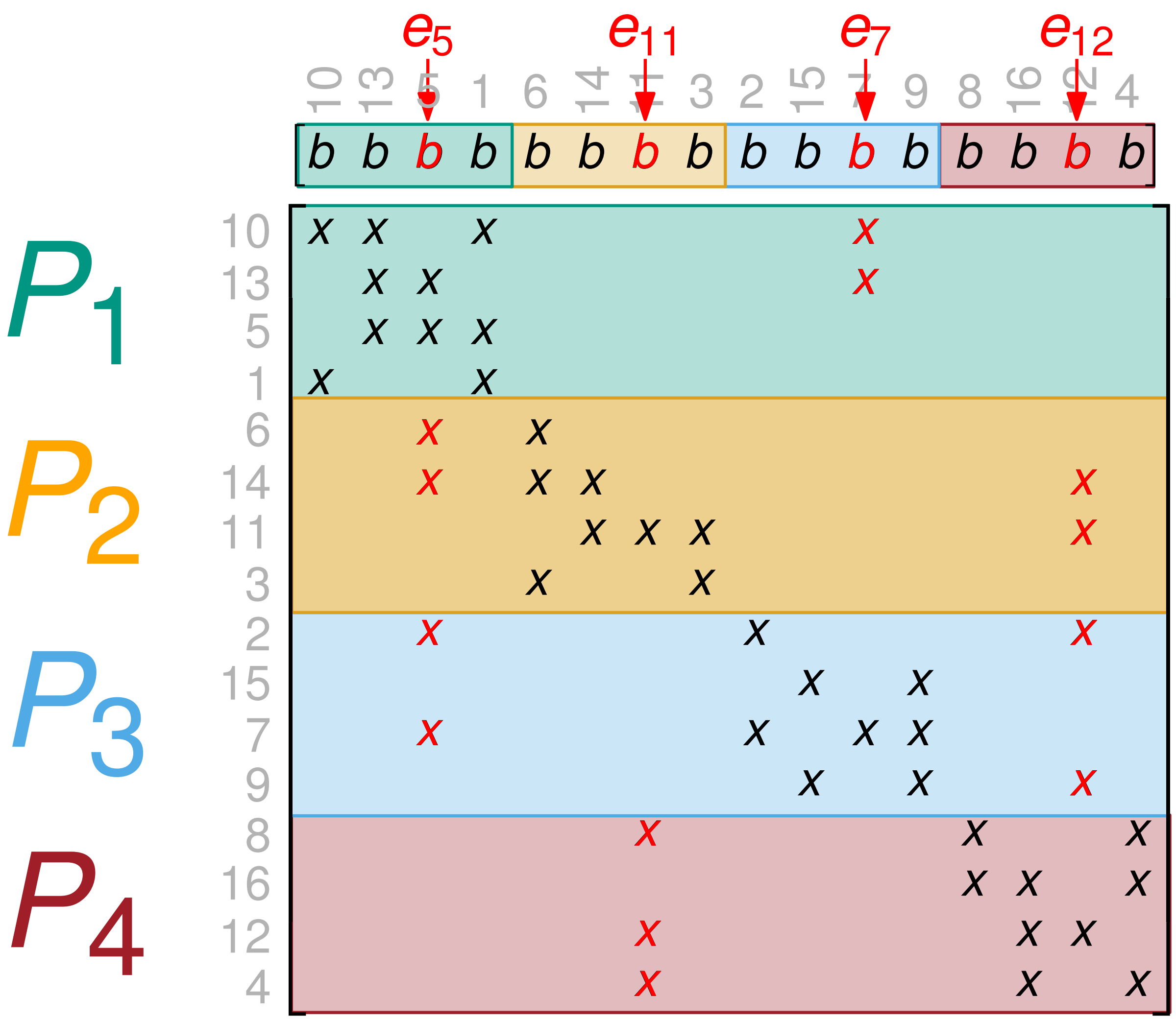
$P_4 \Rightarrow 12$

Commuication Volume?

Institute of Theoretical Informatics
Algorithmics Group

From Hypergraph Partitioning to SpM×V

Where are the cut-hyperedges?



Load Balancing?

⇒ 12

⇒ 12

⇒ 12

⇒ 12

Commuication Volume? ⇒ 6 entries!

Institute of Theoretical Informatics
Algorithmics Group

How does (Hyper)Graph Partitioning work?

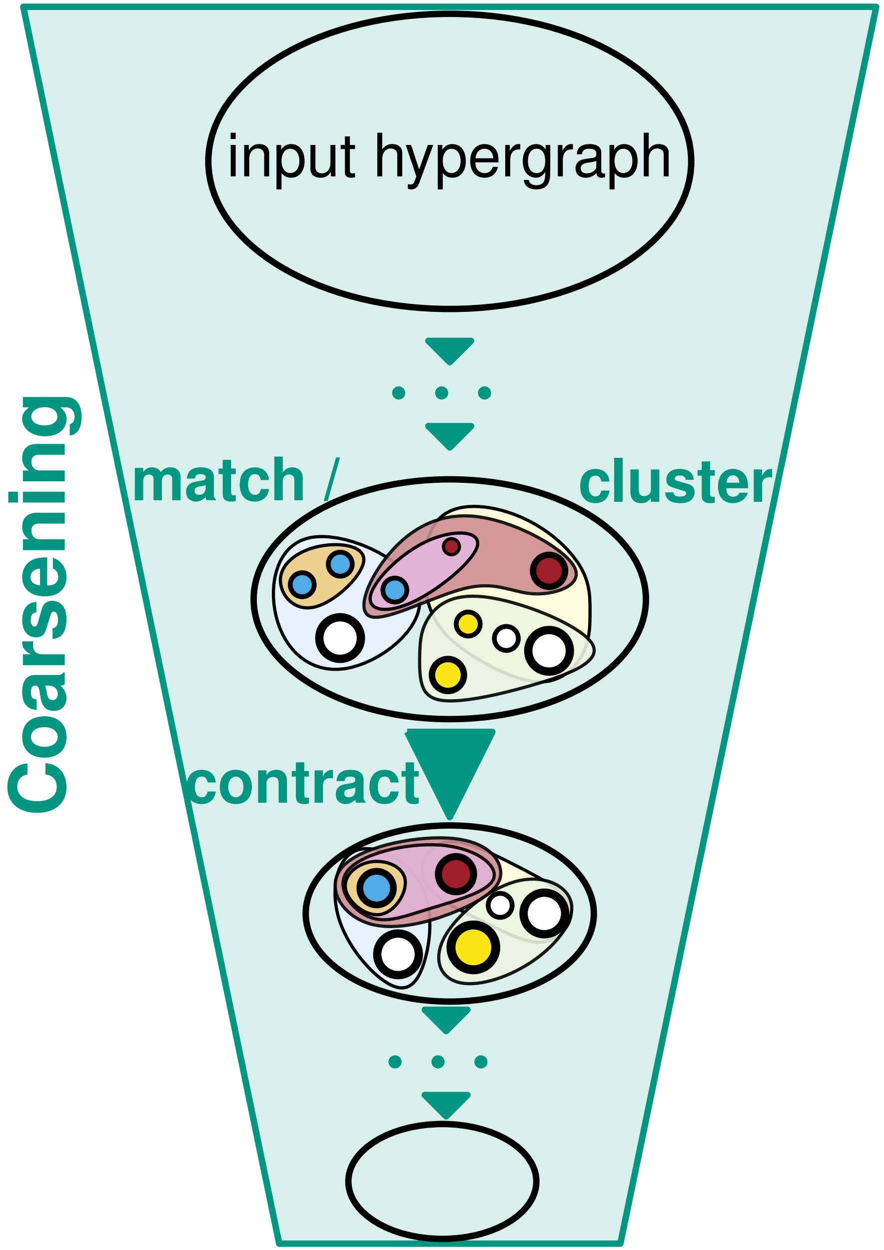
How does

Bad News:

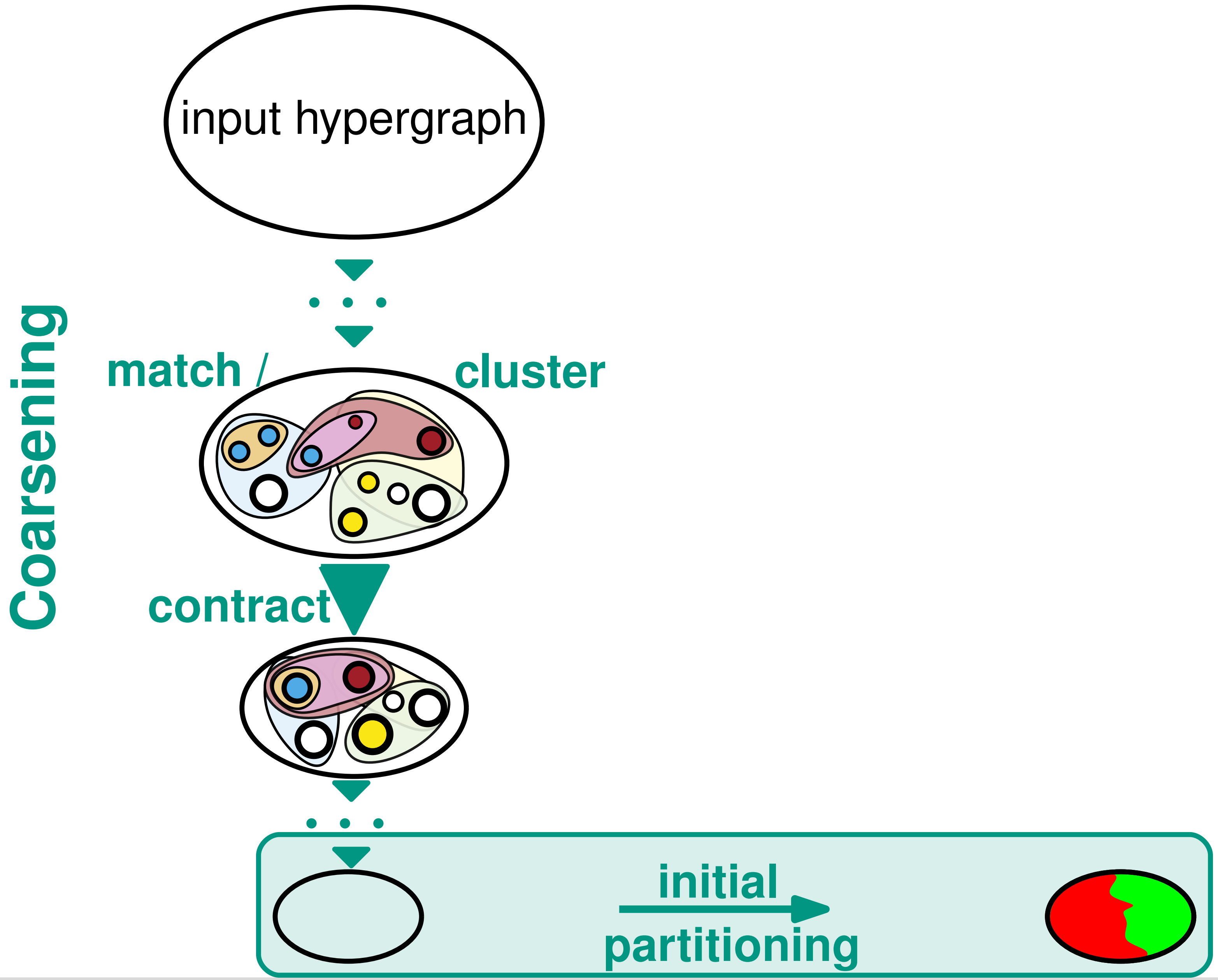
- Hypergraph Partitioning is $\mathbf{NP}$-hard
- even finding good approximate solutions for graphs is $\mathbf{NP}$-hard

work?

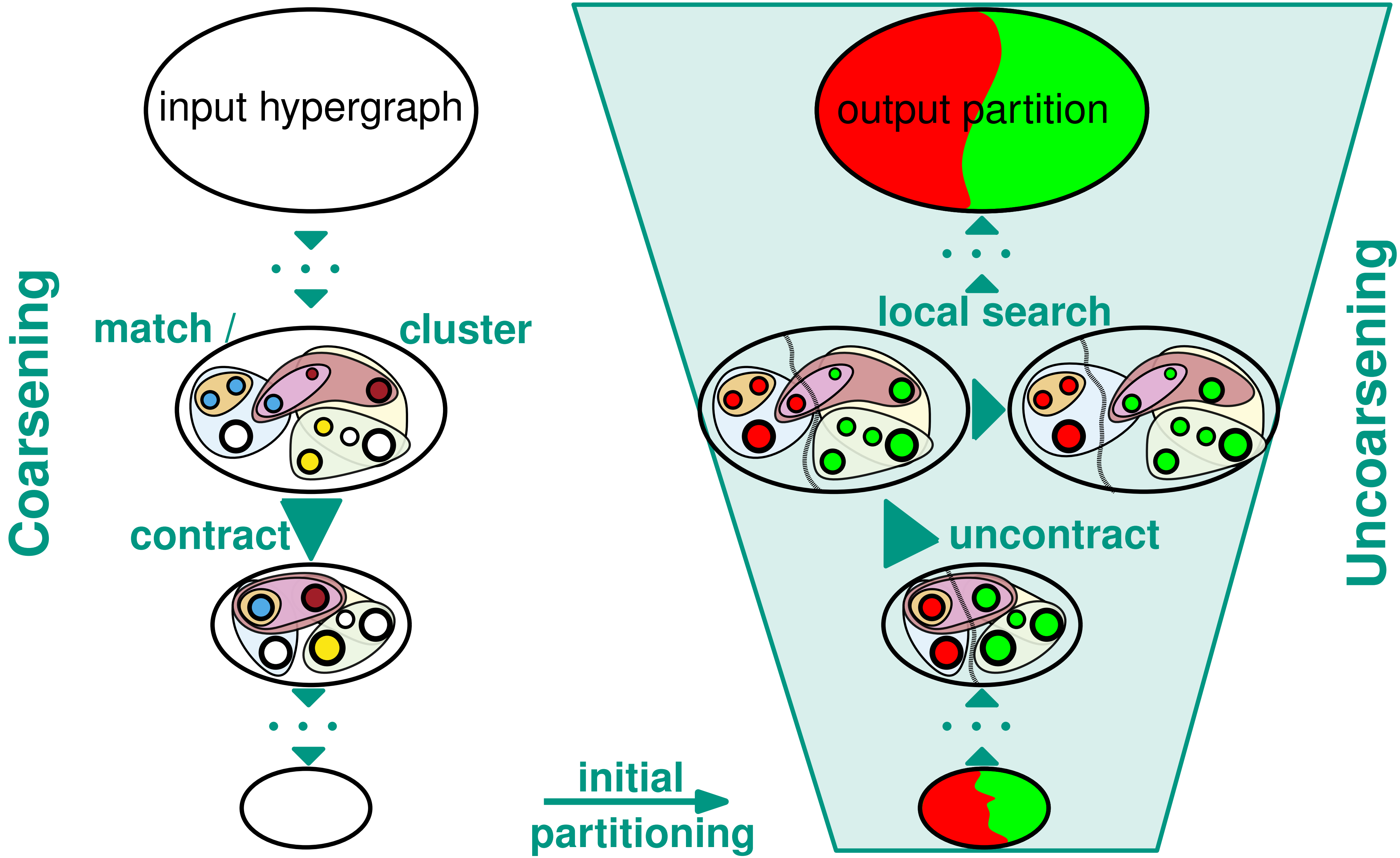
Successful Heuristic: Multilevel Paradigm



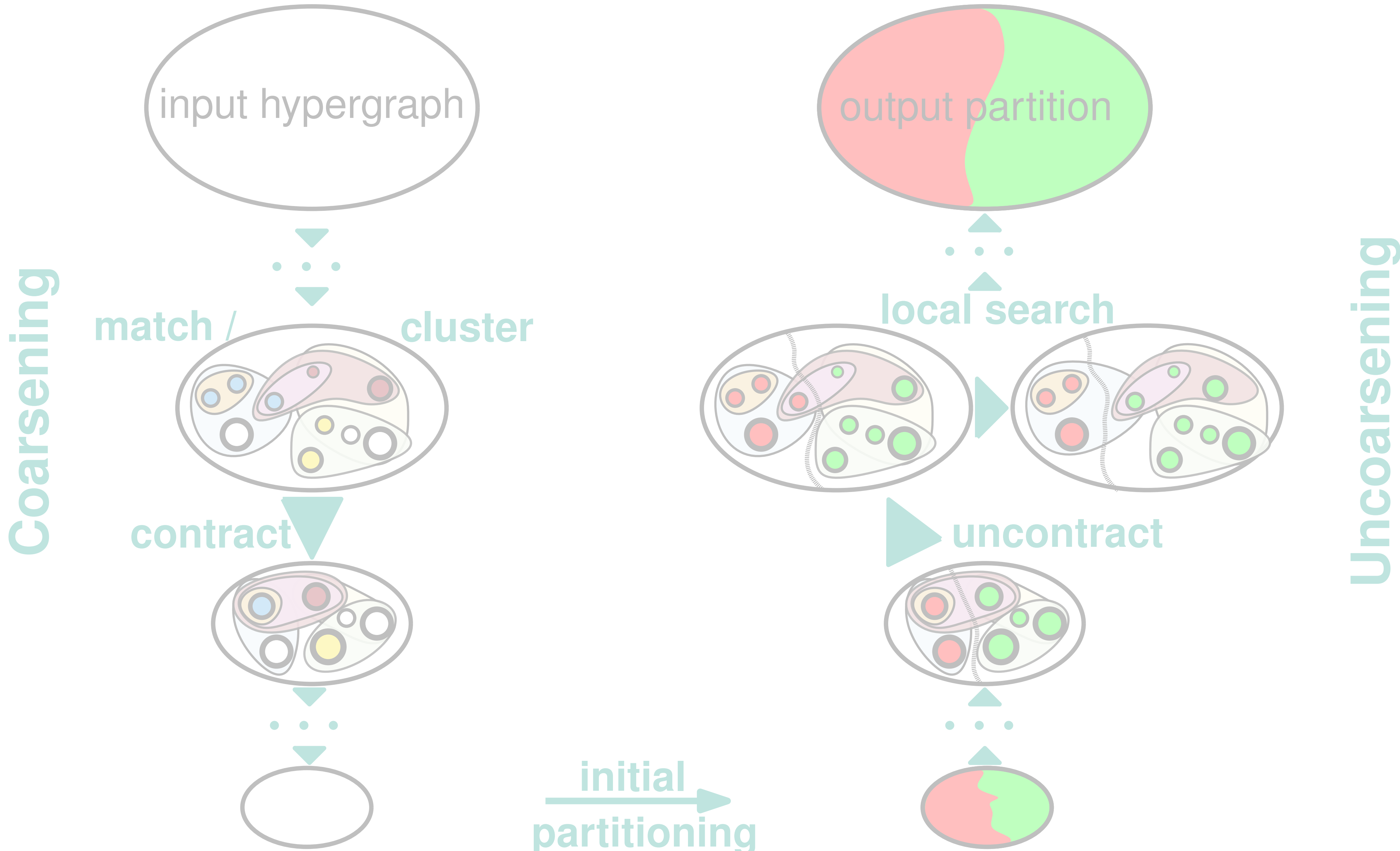
Successful Heuristic: Multilevel Paradigm



Successful Heuristic: Multilevel Paradigm



Multilevel Paradigm - Algorithmic Ingredients

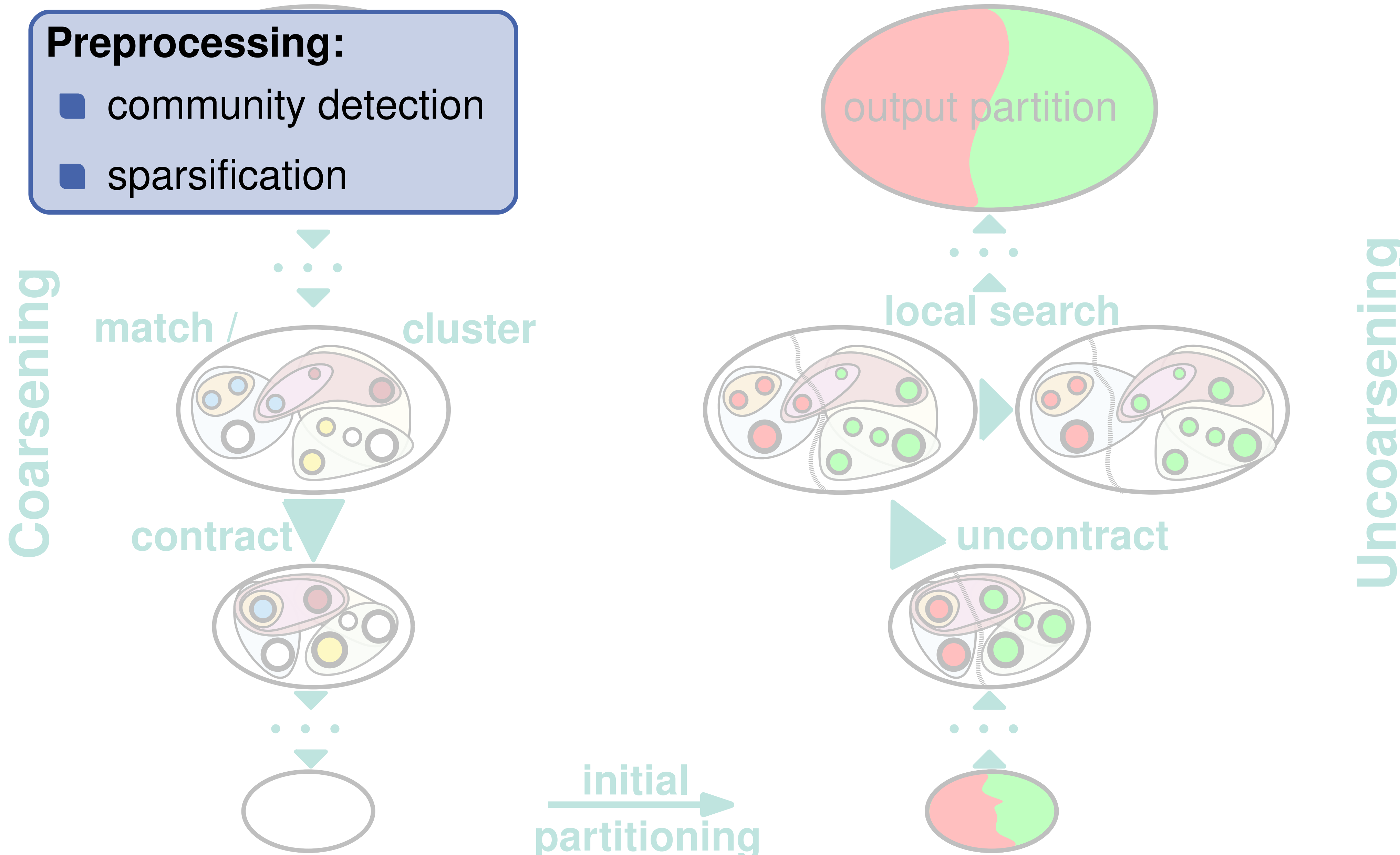


Institute of Theoretical Informatics
Algorithmics Group

Multilevel Paradigm - Algorithmic Ingredients

Preprocessing:

- community detection
- sparsification



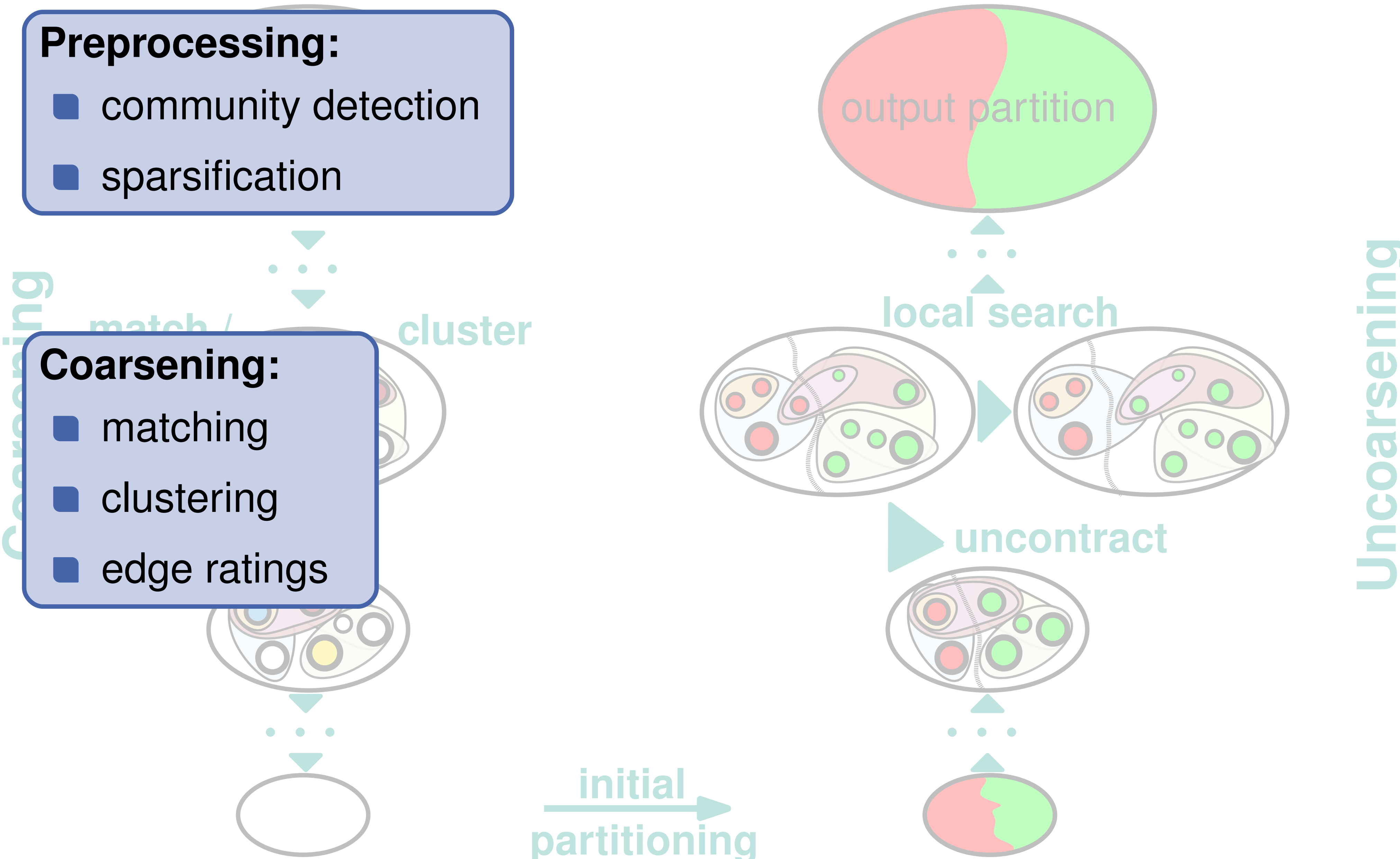
Multilevel Paradigm - Algorithmic Ingredients

Preprocessing:

- community detection
- sparsification

Coarsening:

- matching
- clustering
- edge ratings



Institute of Theoretical Informatics
Algorithmics Group

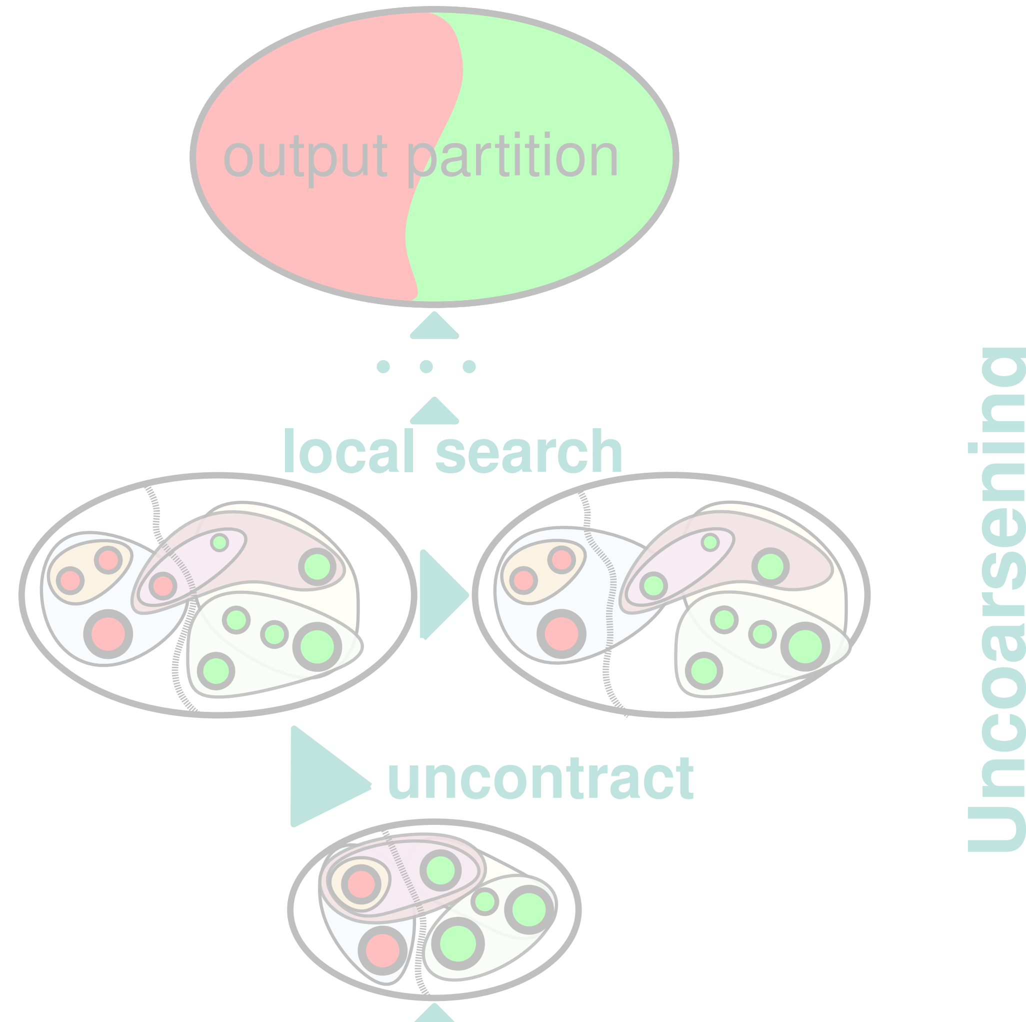
Multilevel Paradigm - Algorithmic Ingredients

Preprocessing:

- community detection
- sparsification

Coarsening:

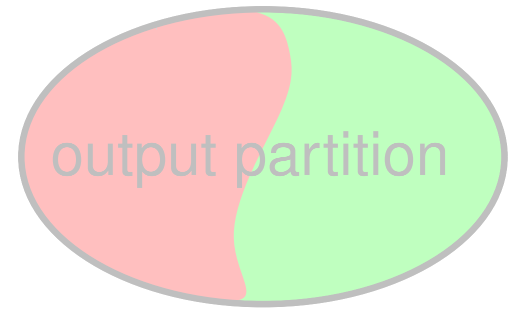
- matching
- clustering
- edge ratings



Initial Partitioning:

- portfolio of various algorithms ⇝ diversification

Multilevel Paradigm - Algorithmic Ingredients



Preprocessing:

- community detection
- sparsification

Coarsening:

- matching
- clustering
- edge ratings

Local Search:

- Kernighan-Lin
- Fiduccia-Mattheyses
- Max-Flow Min-Cut

Initial Partitioning:

- portfolio of various algorithms $\rightsquigarrow$ diversification

Multilevel Paradigm - Algorithmic Ingredients

Preprocessing:
- community detection
- sparsification

Metaheuristics:
- Global Search
- Evolutionary Algorithms

Coarsening:
- matching
- clustering
- edge ratings

Local Search:
- Kernighan-Lin
- Fiduccia-Mattheyses
- Max-Flow Min-Cut

Initial Partitioning:
- portfolio of various algorithms $\rightsquigarrow$ diversification

Multilevel Paradigm - Algorithmic Ingredients

Preprocessing:
- community detection
- sparsification

Metaheuristics:
- Global Search
- Evolutionary Algorithms

Coarsening:
- matching
- clustering
- edge ratings

Parallelization:
- shared memory
- distributed memory

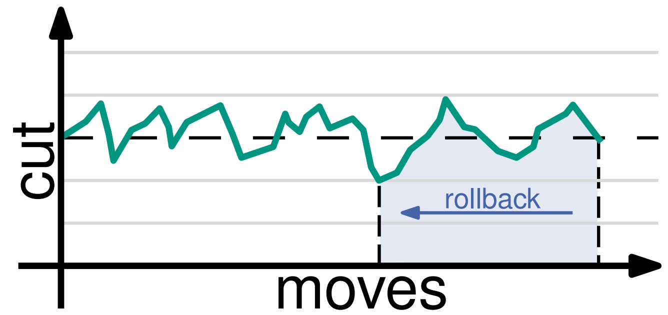
Local Search:
- Kernighan-Lin
- Fiduccia-Mattheyses
- Max-Flow Min-Cut

Initial Partitioning:
- portfolio of various algorithms $\rightsquigarrow$ diversification

Fiduccia-Mattheyses Algorithm

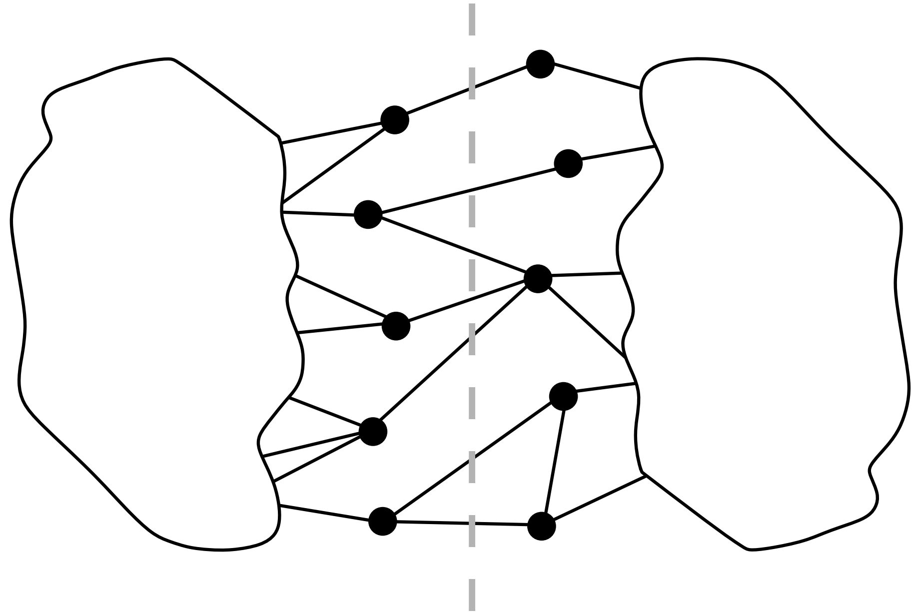
Algorithm 1: FM Local Search

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



can worsen solution

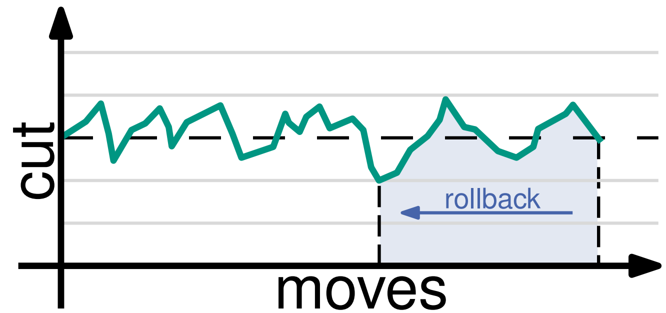
- compute gain $g(v) = d_{ext}(v) - d_{int}(v)$
- alternate between blocks
- edge-cut: **7**



Fiduccia-Mattheyses Algorithm

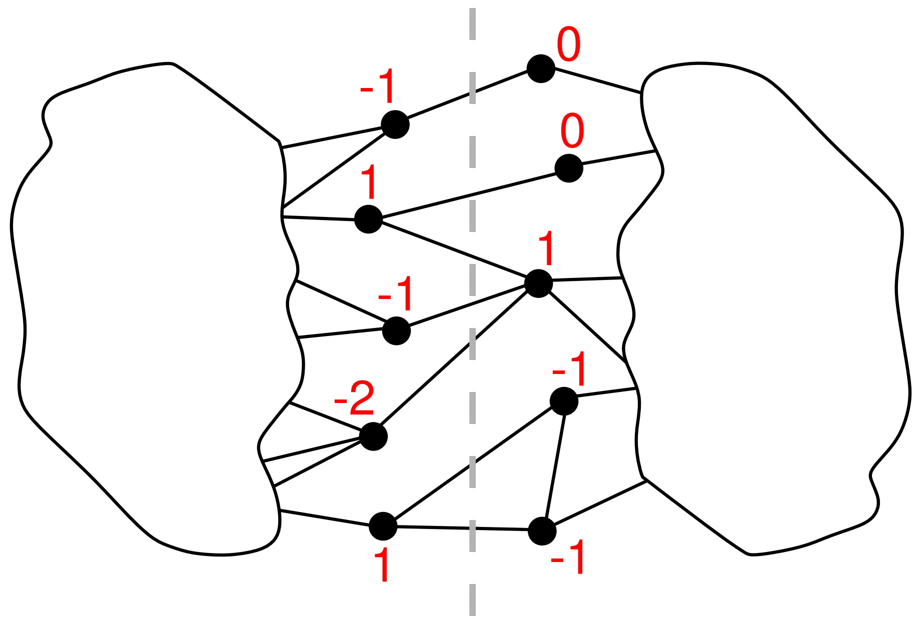
Algorithm 1: FM Local Search

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



can worsen solution

- compute gain $g(v) = d_{ext}(v) - d_{int}(v)$
- alternate between blocks
- edge-cut: **7**



Institute of Theoretical Informatics
Algorithmics Group

Fiduccia-Mattheyses Algorithm

Karlsruhe Institute of Technology

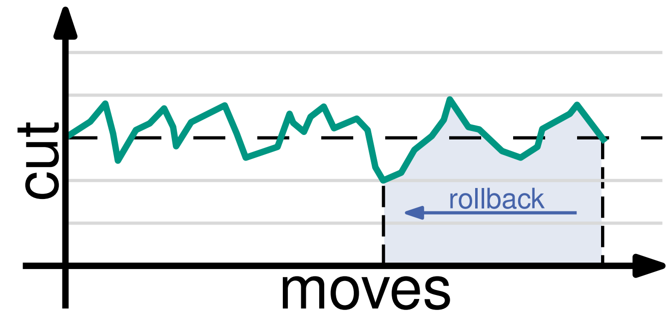
Algorithm 1: FM Local Search

while $\neg$ *done* **do**

  `find` best move

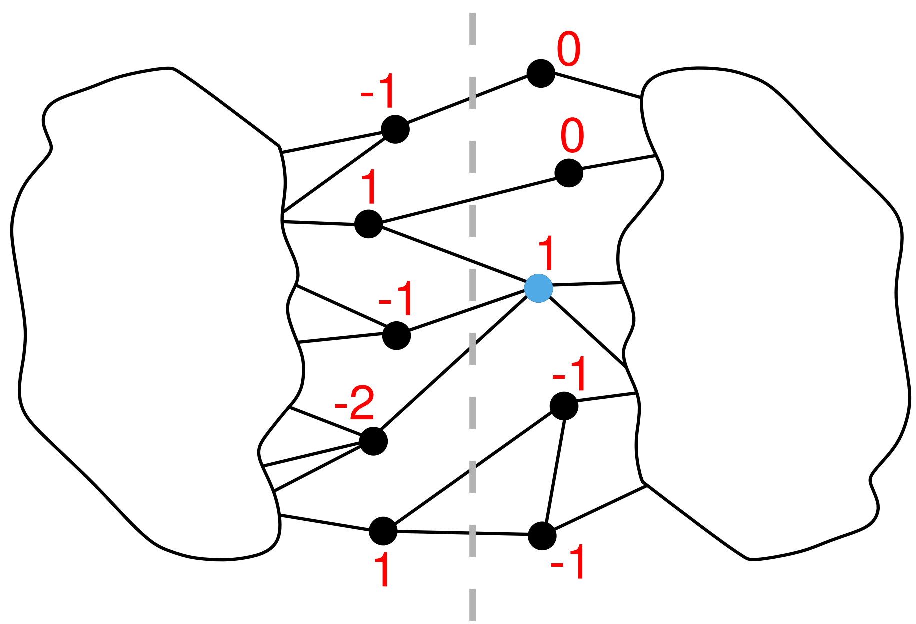
  `perform` best move

`rollback` to best solution



can worsen solution

- compute gain $g(v) = d_{\text{ext}}(v) - d_{\text{int}}(v)$
- alternate between blocks
- edge-cut: **7**

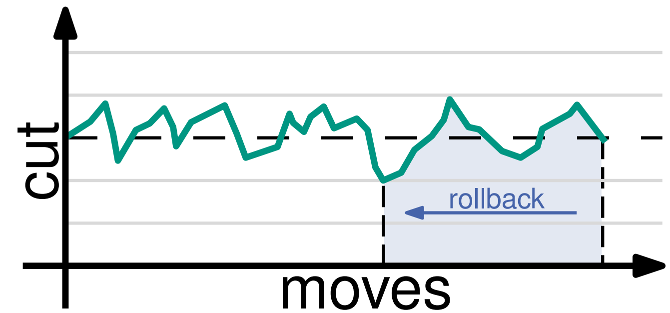


Institute of Theoretical Informatics
Algorithmics Group

Fiduccia-Mattheyses Algorithm

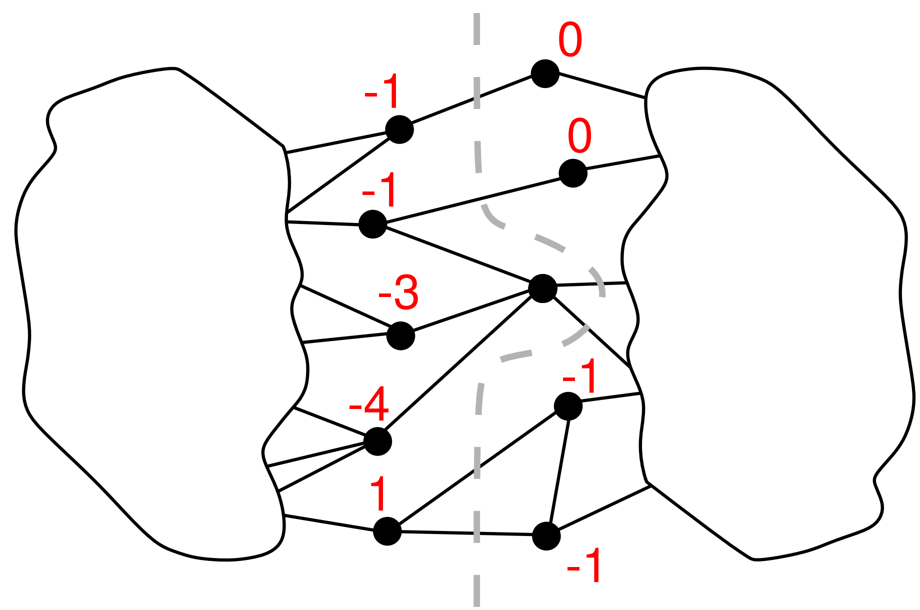
Algorithm 1: FM Local Search

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



can worsen solution

- recalculate gain $g(v)$ of neighbors
- move each node at most once
- edge-cut: **7**, **6**



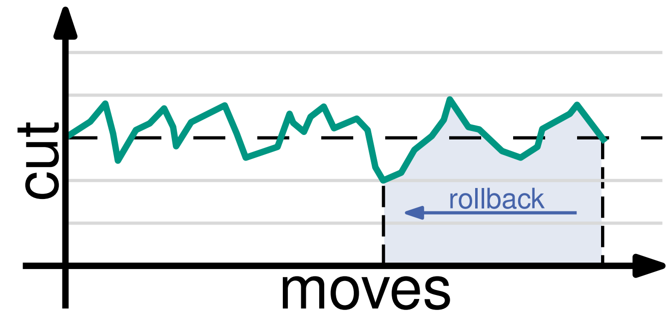
Institute of Theoretical Informatics
Algorithmics Group

Fiduccia-Mattheyses Algorithm

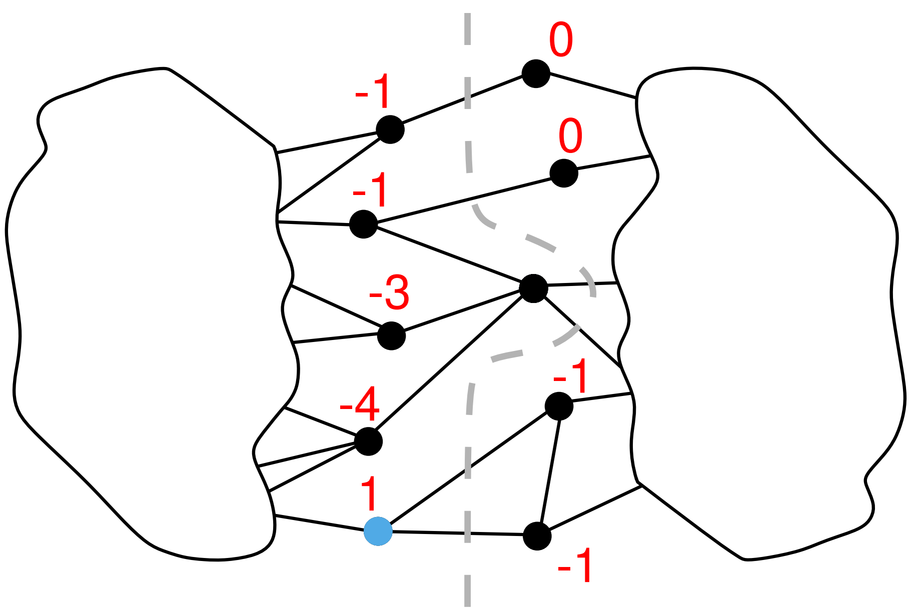
Karlsruhe Institute of Technology

Algorithm 1: FM Local Search

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



can worsen solution

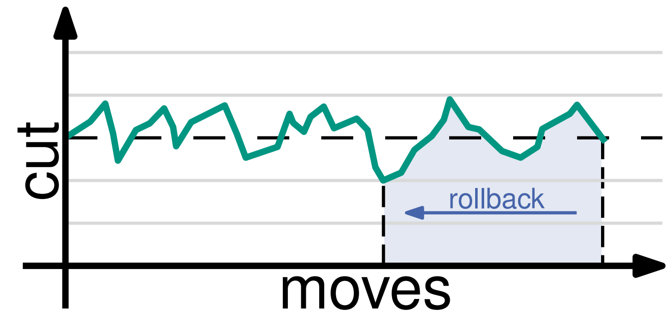


- recalculate gain $g(v)$ of neighbors
- move each node at most once
- edge-cut: **7**, **6**

Fiduccia-Mattheyses Algorithm

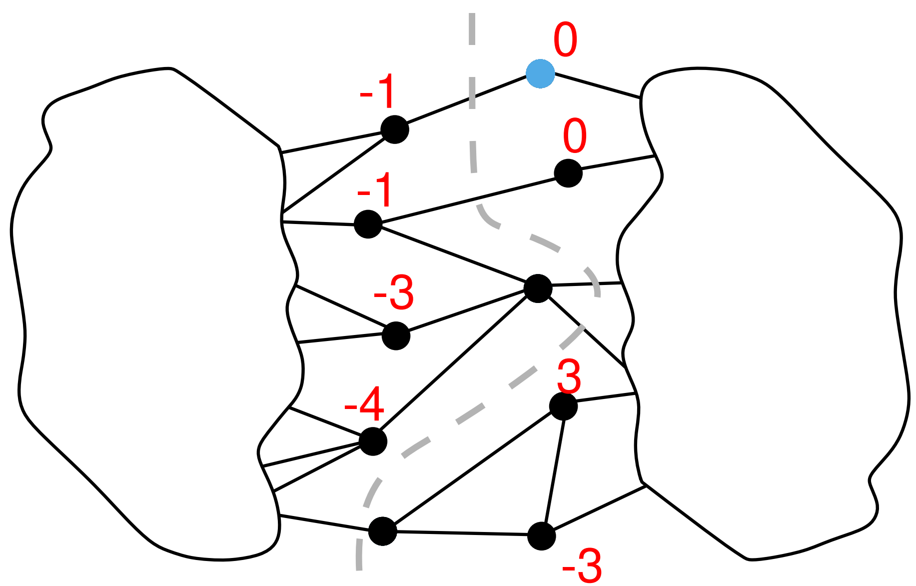
Algorithm 1: FM Local Search

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



can worsen solution

- recalculate gain $g(v)$ of neighbors
- move each node at most once
- edge-cut: **7**, **6**,**5**

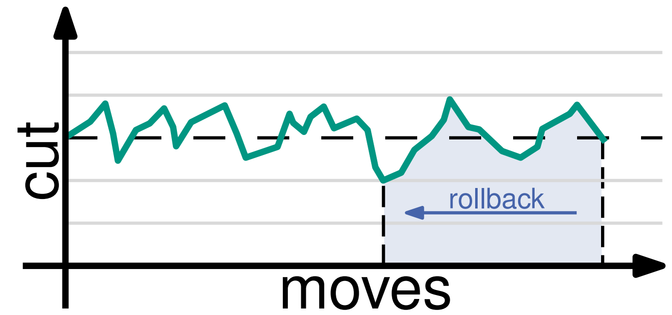


Institute of Theoretical Informatics
Algorithmics Group

Fiduccia-Mattheyses Algorithm

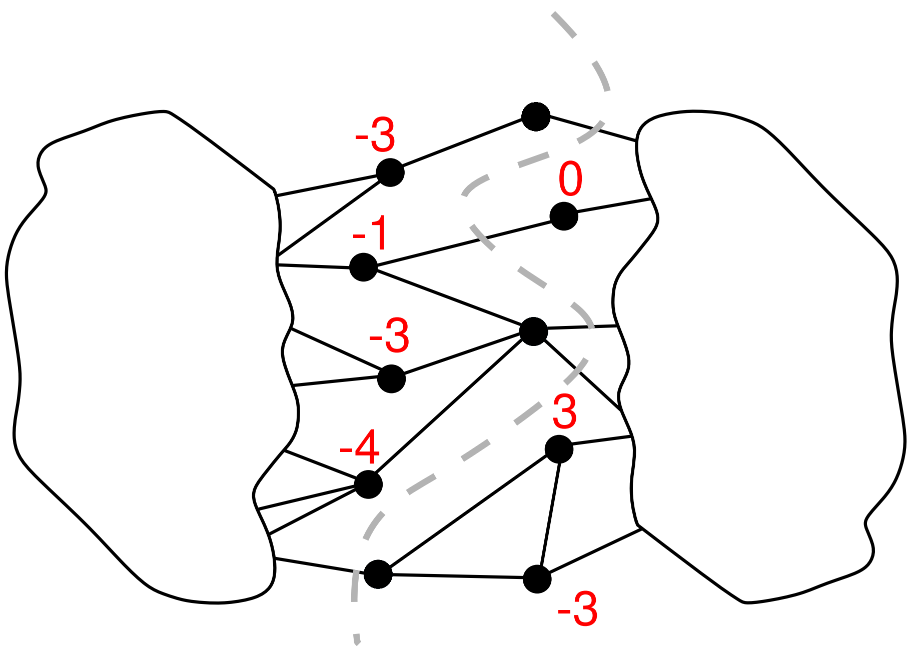
Algorithm 1: FM Local Search

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



can worsen solution

- recalculate gain $g(v)$ of neighbors
- move each node at most once
- edge-cut: **7**, **6**,**5**,**5**

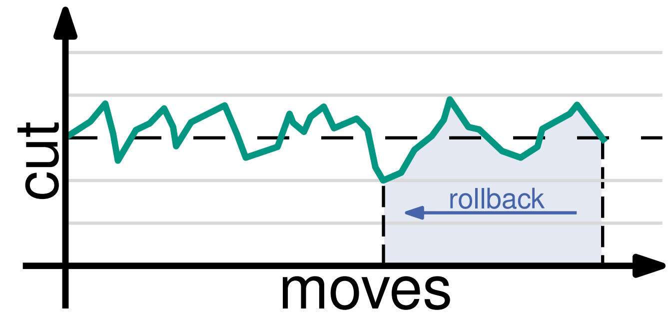


Institute of Theoretical Informatics
Algorithmics Group

Fiduccia-Mattheyses Algorithm

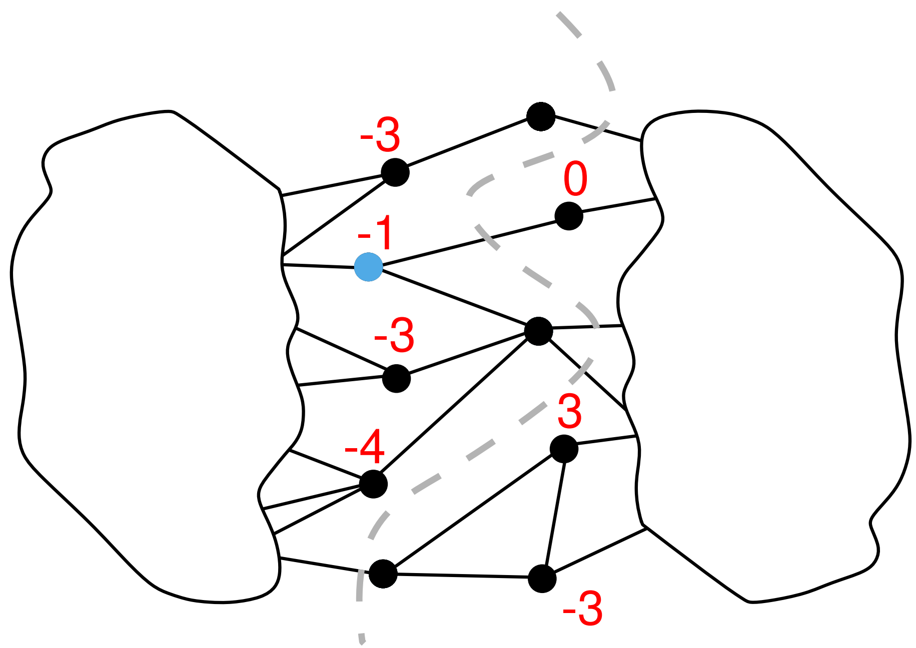
Algorithm 1: FM Local Search

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



can worsen solution

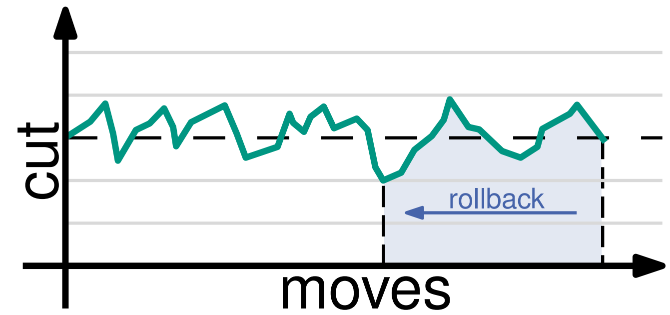
- recalculate gain $g(v)$ of neighbors
- move each node at most once
- edge-cut: **7**, **6**,**5**,**5**



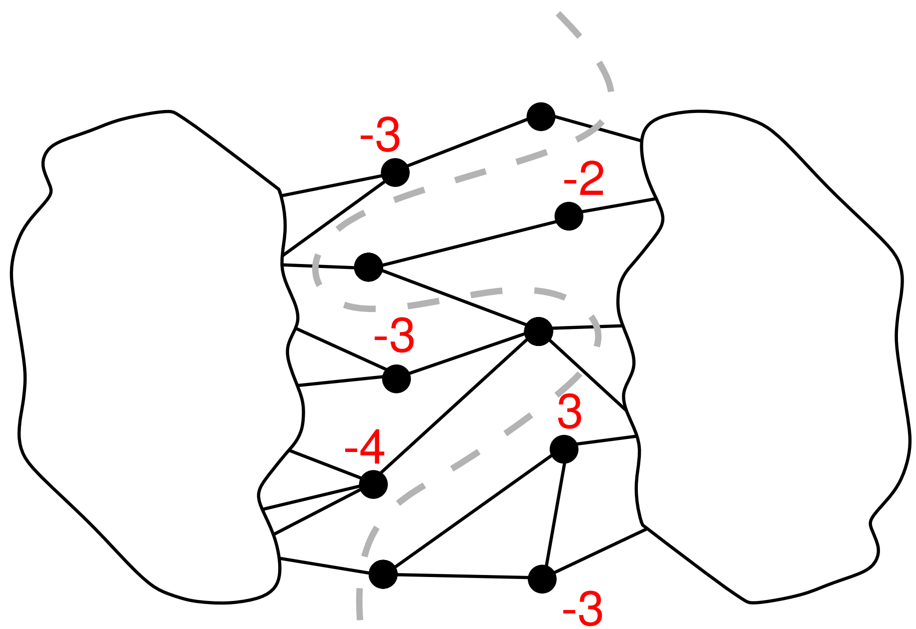
Fiduccia-Mattheyses Algorithm

Algorithm 1: FM Local Search

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



can worsen solution

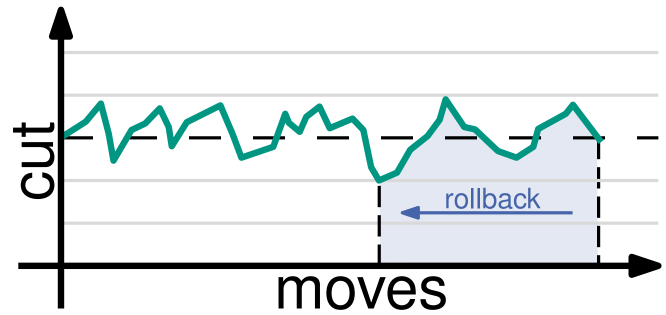


- recalculate gain $g(v)$ of neighbors
- move each node at most once
- edge-cut: **7**, **6**,**5**,**5**,**6**

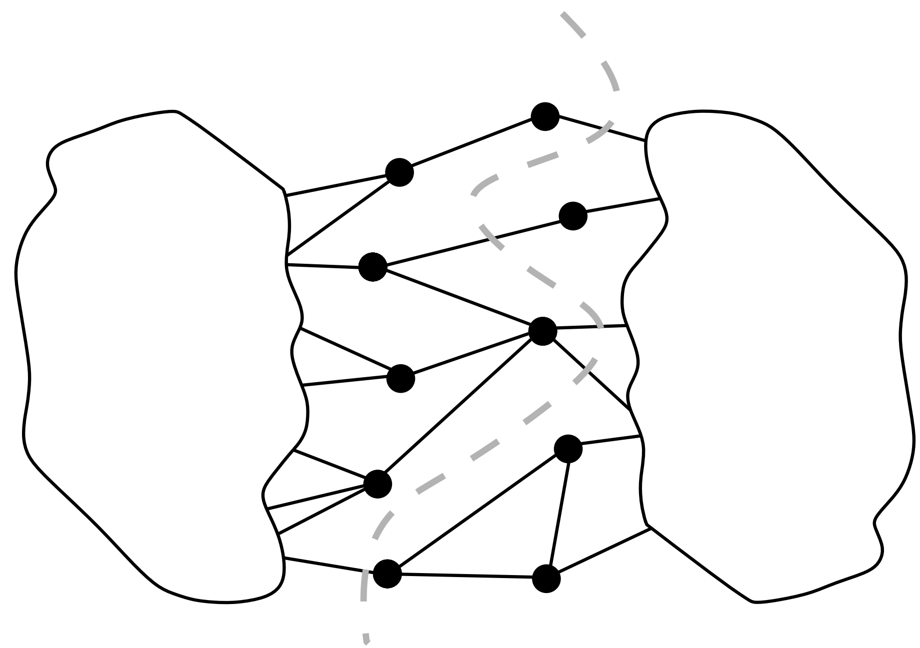
Fiduccia-Mattheyses Algorithm

Algorithm 1: FM Local Search

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



can worsen solution

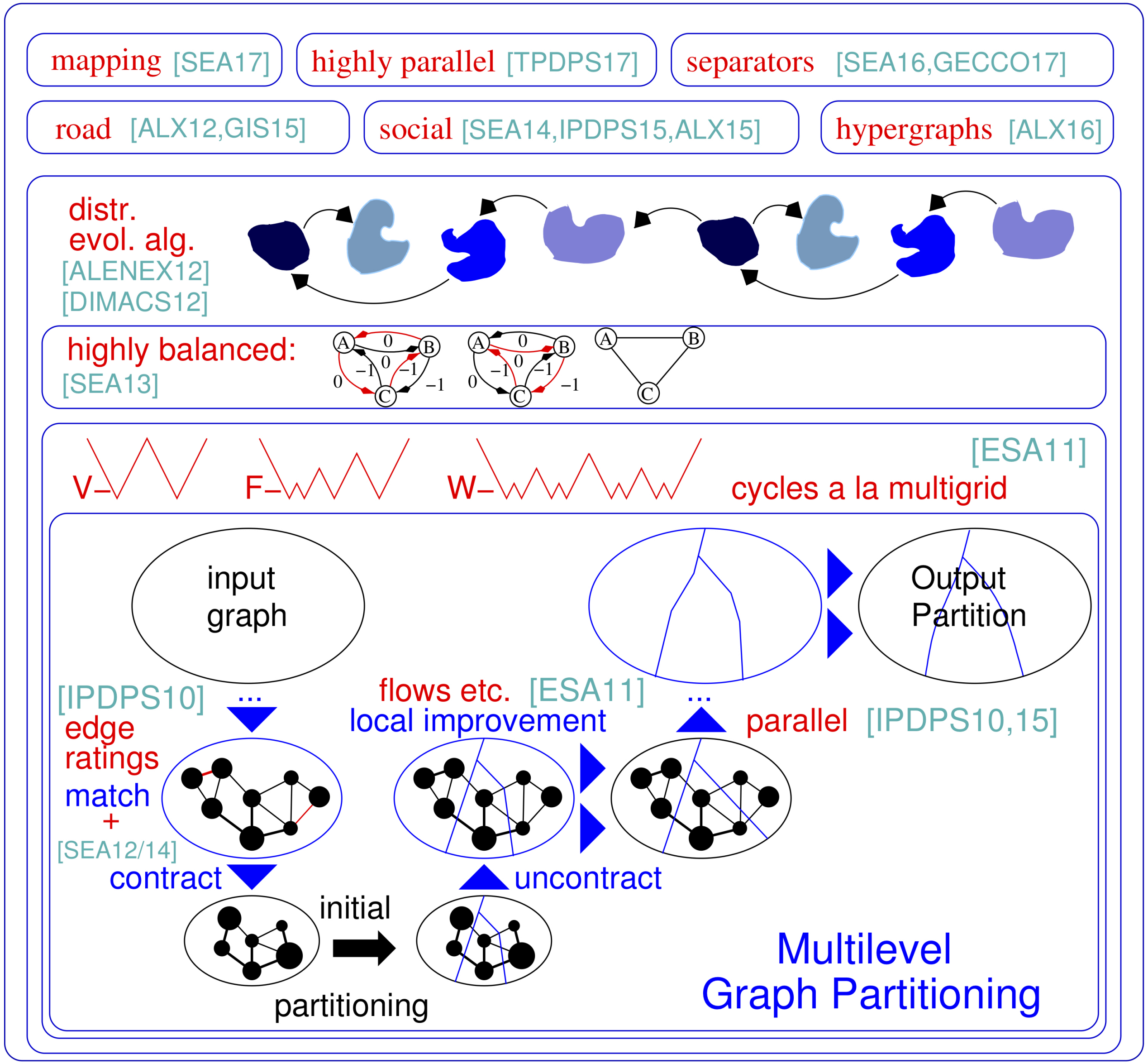


- recalculate gain $g(v)$ of neighbors
- move each node at most once
- edge-cut: **7**, **6**,**5**,**5**,**6**

rollback

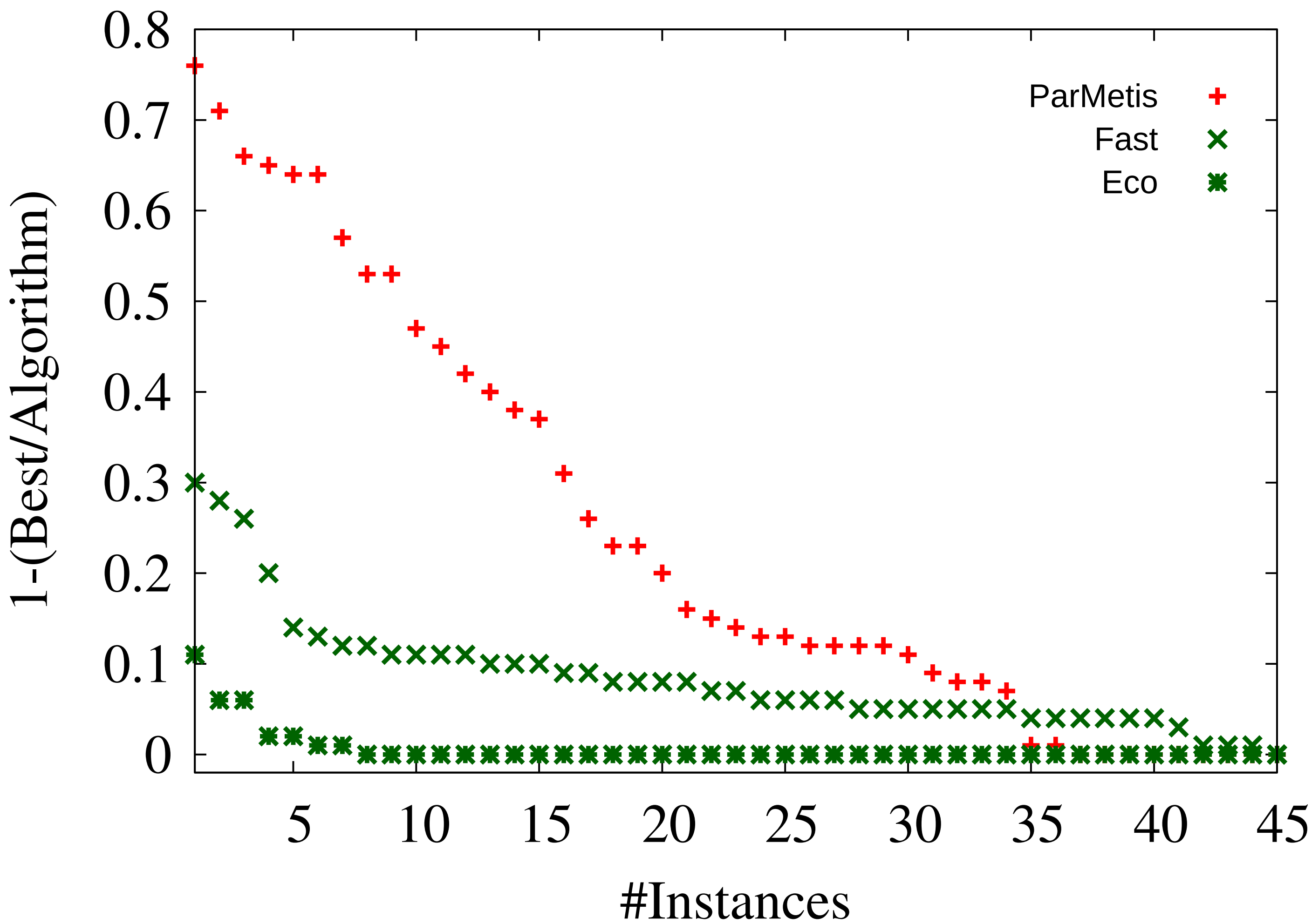
Institute of Theoretical Informatics
Algorithmics Group

KaHIP - Karlsruhe High Quality Partitioning



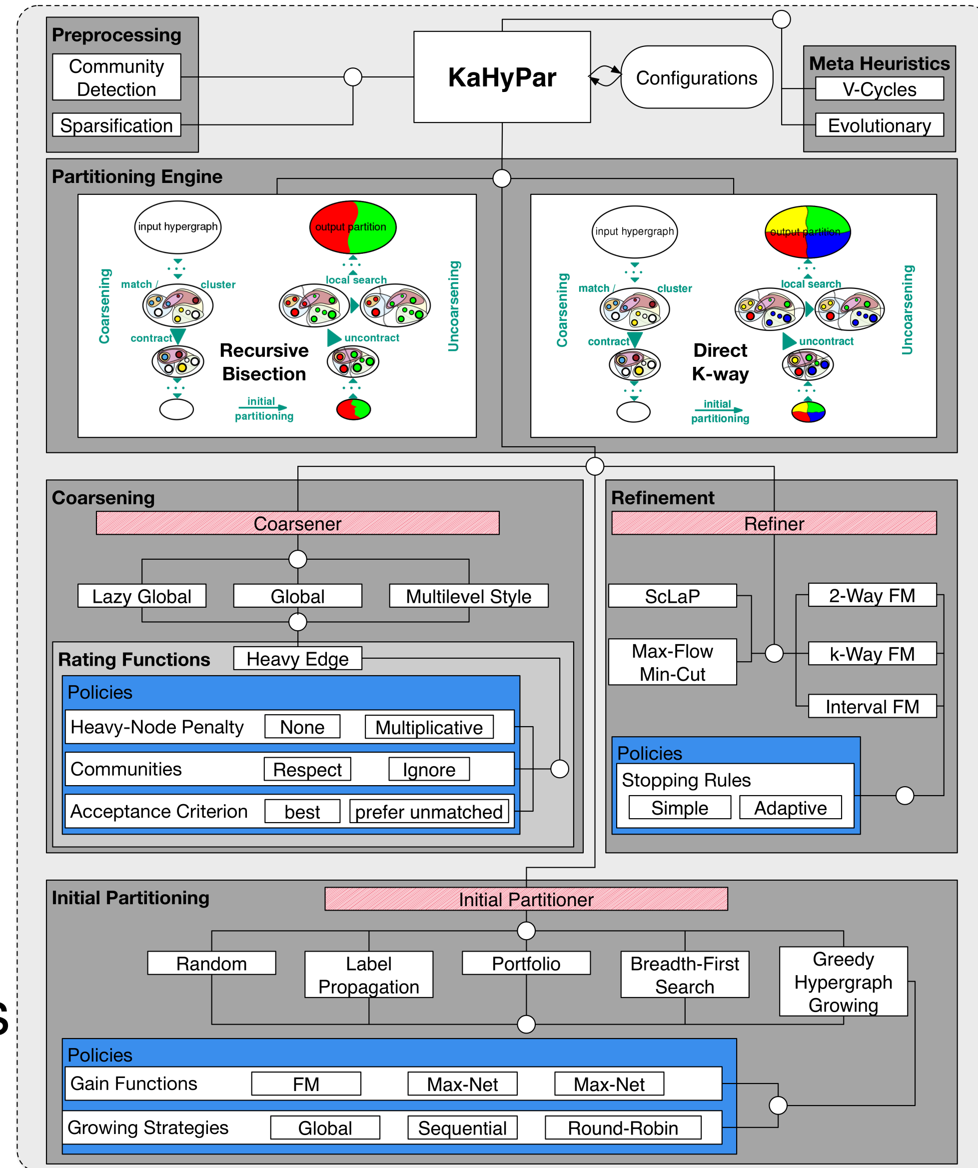
http://algo2.iti.kit.edu/kahip/

Experimental Results – KaHIP (ParHIP)



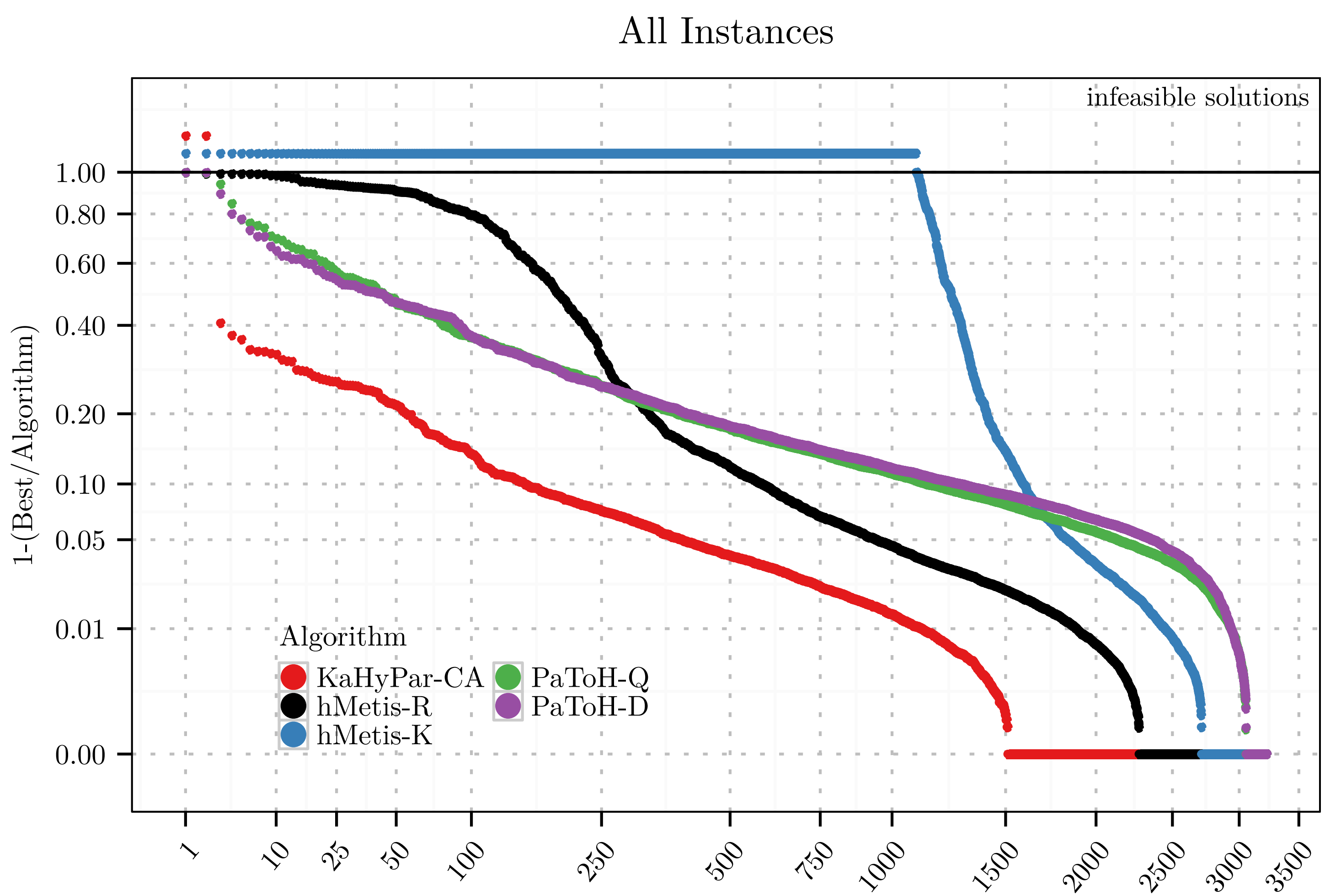
KaHyPar - Karlsruhe Hypergraph Partitioning

- *n*-Level Partitioning Framework
- Objectives:
  - hyperedge cut
  - connectivity ($\lambda - 1$)
- Partitioning Modes:
  - recursive bisection
  - direct *k*-way
- Upcoming Features:
  - evolutionary algorithm
  - flow-based refinement
  - advanced local search algorithms
- `http://www.kahypar.org`



Institute of Theoretical Informatics
Algorithmics Group

Experimental Results – KaHyPar



Conclusion

(Hyper)Graph Partitioning:

- fundamental graph problem with many application areas
- successful heuristic: multilevel approach + local search
- Graphs: KaHIP – `http://algo2.iti.kit.edu/kahip/`
- Hypergraphs: KaHyPar – `http://www.kahypar.org`