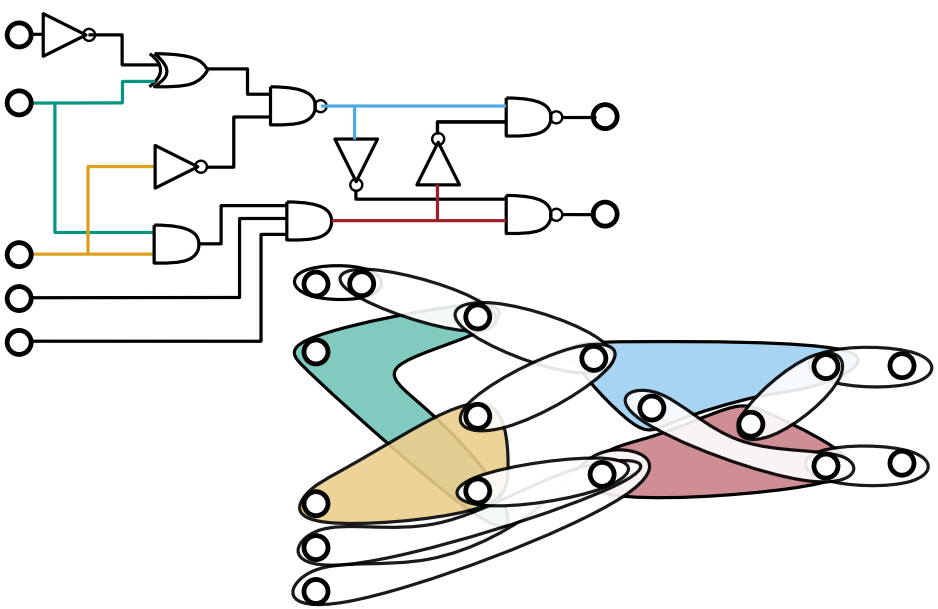
Karlsruhe Institute of Technology

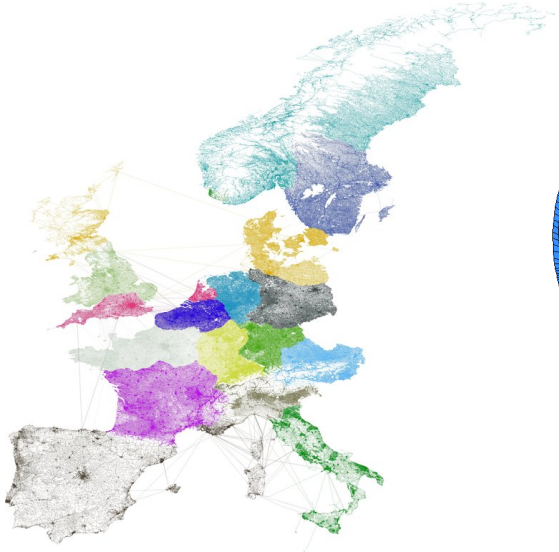
Brief Introduction to Hypergraph Partitioning

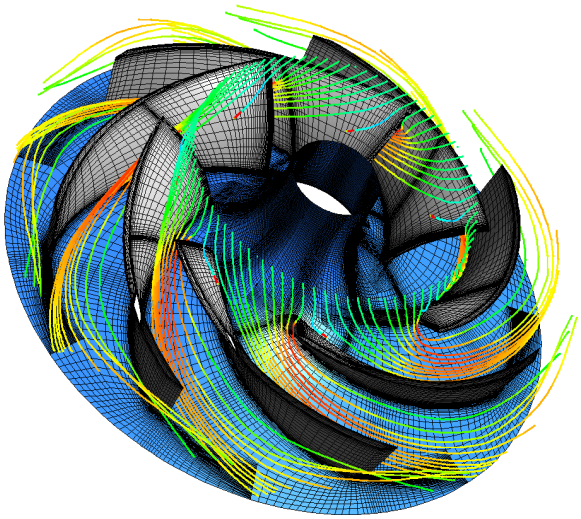
Bioinformatics Programming Practical Kickoff Meeting · April 19, 2018

Sebastian Schlag

INSTITUTE OF THEORETICAL INFORMATICS · ALGORITHMICS GROUP





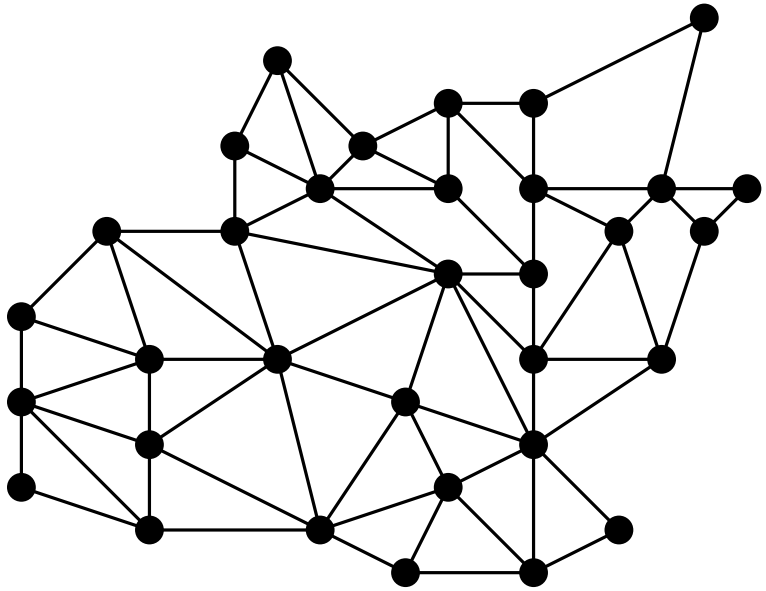


Graphs and Hypergraphs

Graph $G = (V, E)$

vertices — edges

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

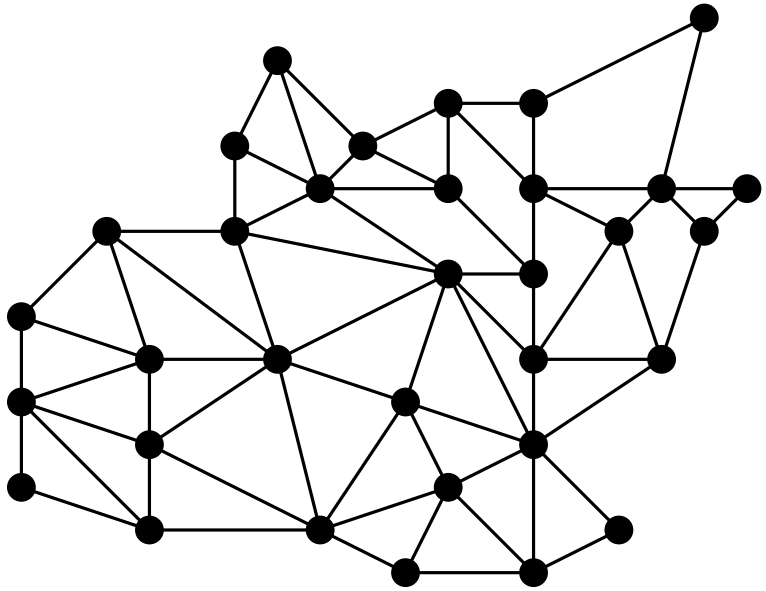


Institute of Theoretical Informatics
Algorithmics Group

Graphs and Hypergraphs

Graph $G = (V, E)$

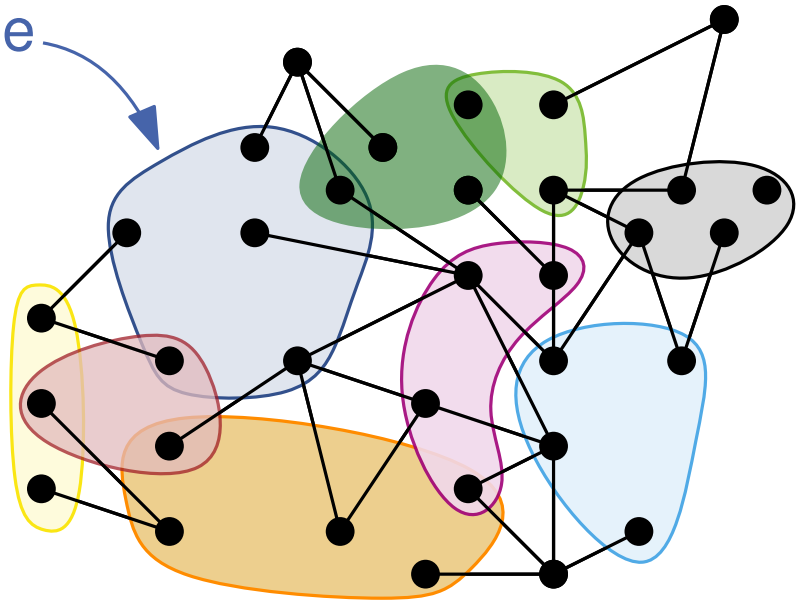
vertices — edges



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

Hypergraph $H = (V, E)$

hyperedge



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

Institute of Theoretical Informatics
Algorithmics Group

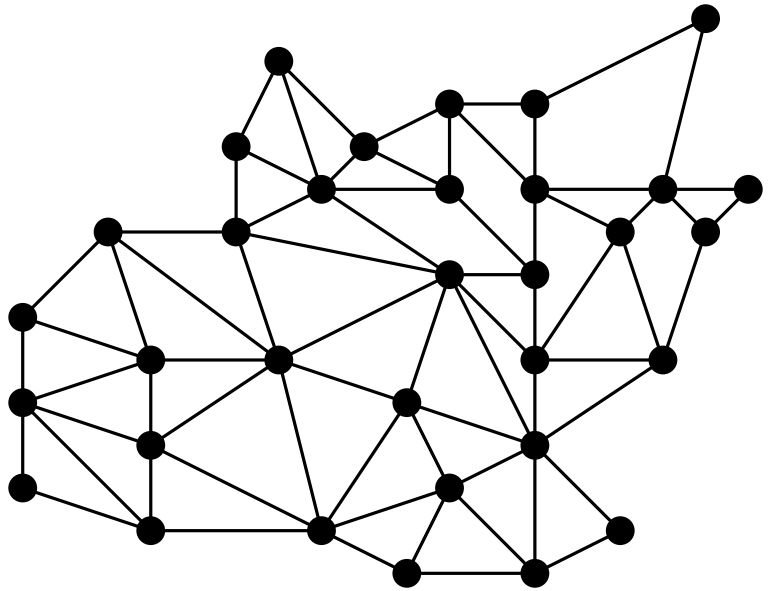
ε-Balanced Graph and Hypergraph Partitioning

Partition (hyper)graph $G = (V, E, c : V \to \mathrm{R}_{>0}, \omega : E \to \mathrm{R}_{>0})$ 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

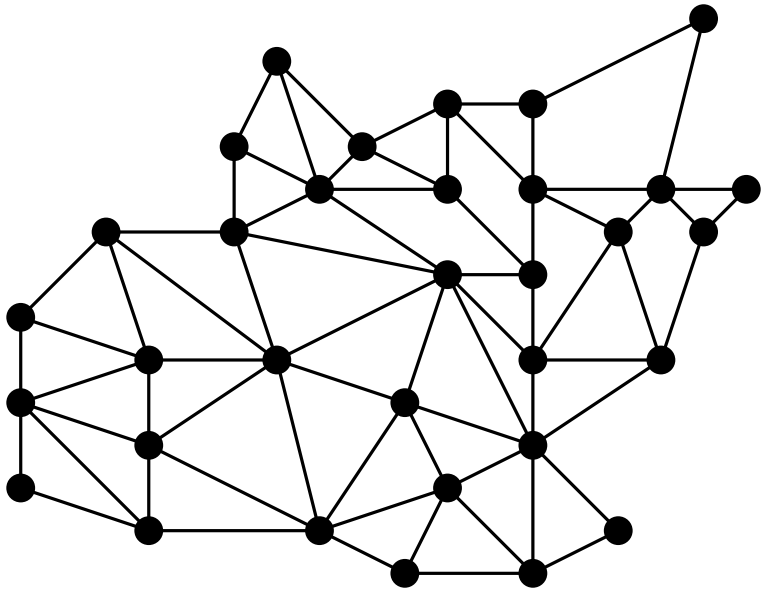
Partition (hyper)graph $G = (V, E, c : V \to \mathrm{R}_{>0}, \omega : E \to \mathrm{R}_{>0})$ into **k** disjoint blocks $V_1, \ldots, V_k$ s.t.

- 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

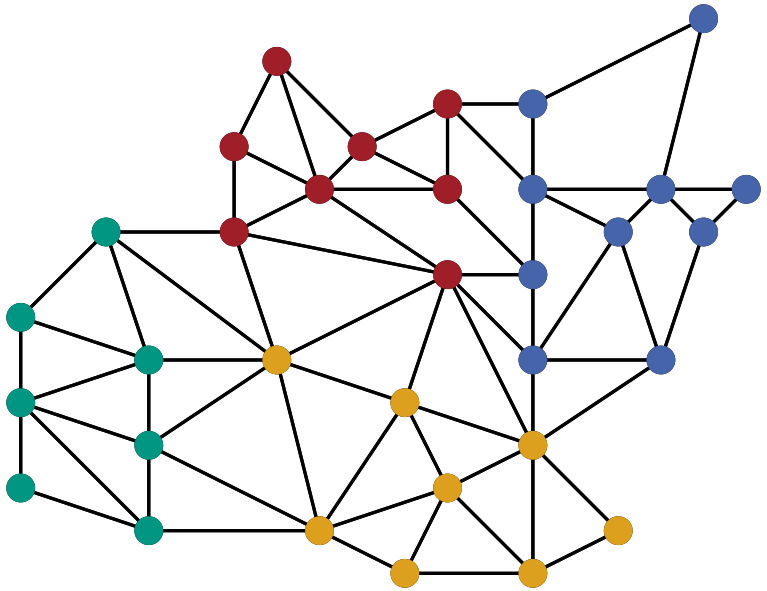
Partition (hyper)graph $G = (V, E, c : V \to \mathrm{R}_{>0}, \omega : E \to \mathrm{R}_{>0})$ into **k** disjoint blocks $V_1, \ldots, V_k$ s.t.

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

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

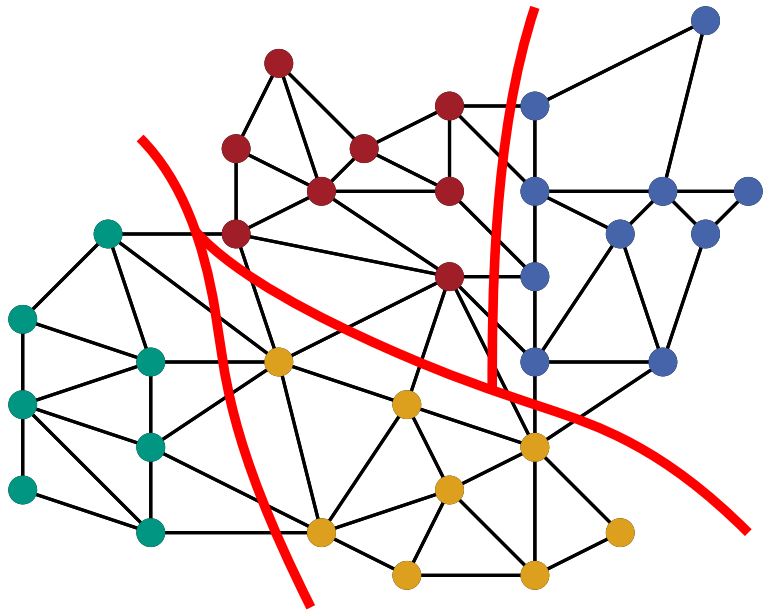
imbalance parameter

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

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

Common Objectives:

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



Institute of Theoretical Informatics
Algorithmics Group

ε-Balanced Graph and Hypergraph Partitioning

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

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

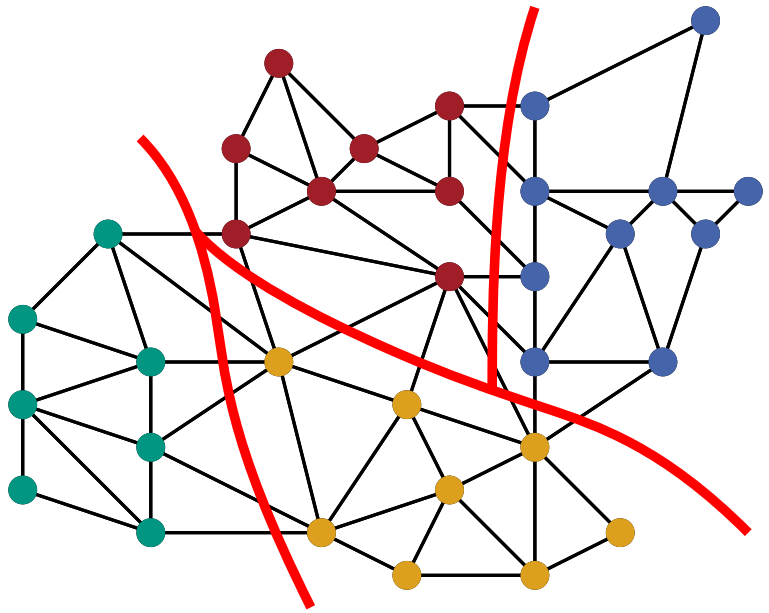
imbalance parameter

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

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

Common Objectives:

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



Institute of Theoretical Informatics
Algorithmics Group

ε-Balanced Graph and Hypergraph Partitioning

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

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

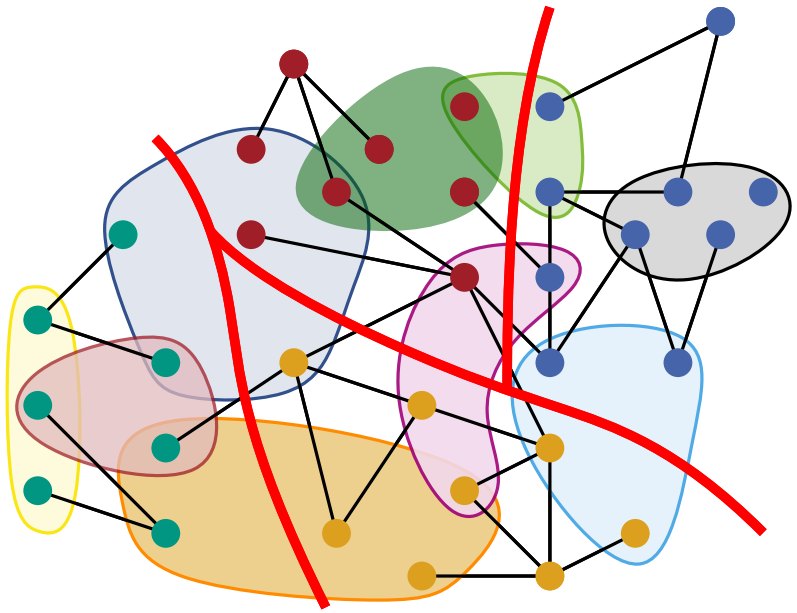
imbalance parameter

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

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

Common Objectives:

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



Institute of Theoretical Informatics
Algorithmics Group

ε-Balanced Graph and Hypergraph Partitioning

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

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

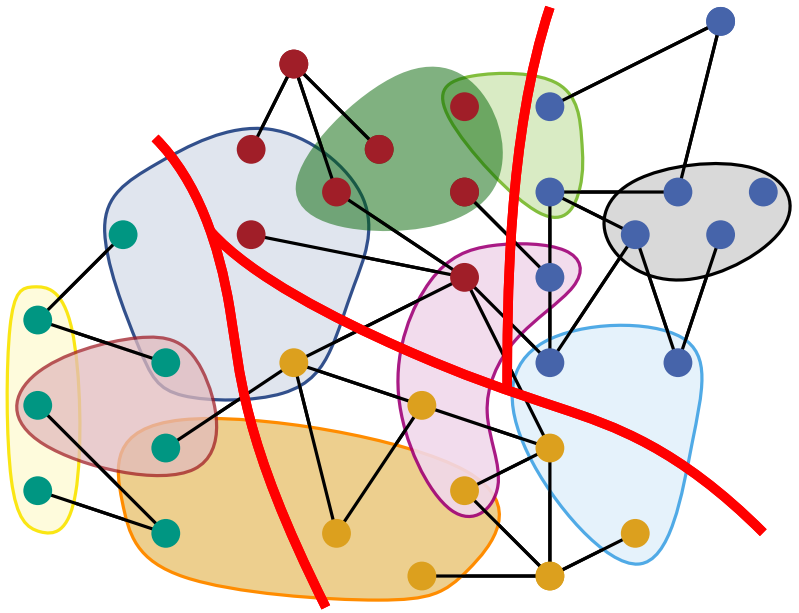
imbalance parameter

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

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

Common Objectives:

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



ε-Balanced Graph and Hypergraph Partitioning

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

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

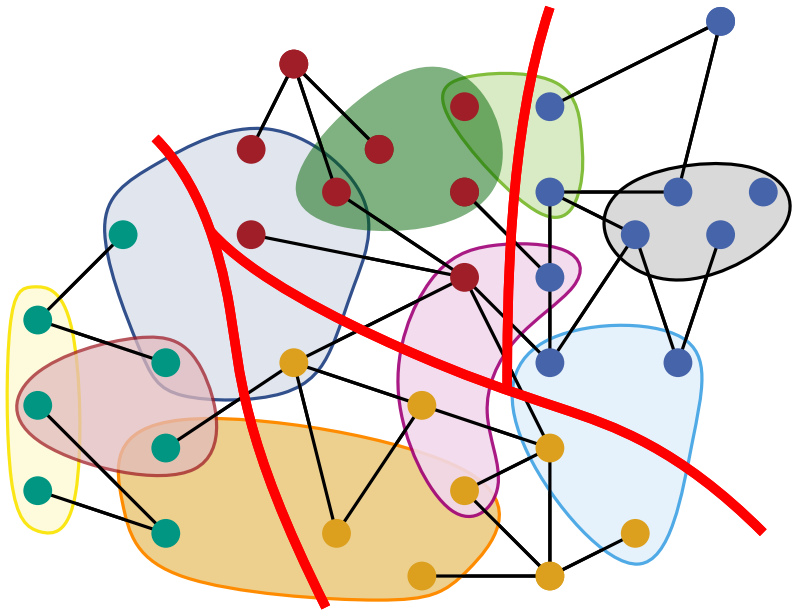
imbalance parameter

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

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

Common Objectives:

- Graphs:
  - cut: $\sum_{e \in \text{cut}} \omega(e) = 17$
- Hypergraphs:
  - cut: $\sum_{e \in \text{cut}} \omega(e) = 5$



ε-Balanced Graph and Hypergraph Partitioning

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

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

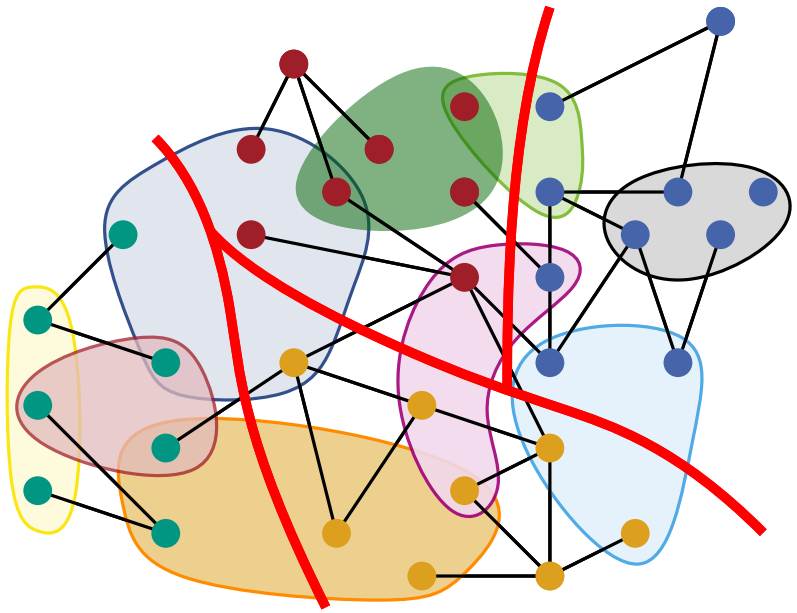
imbalance parameter

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

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

Common Objectives:

- Graphs:
  - cut: $\sum_{e \in \text{cut}} \omega(e) = 17$
- Hypergraphs:
  - cut: $\sum_{e \in \text{cut}} \omega(e) = 5$
  - connectivity: $\sum_{e \in \text{cut}} (\lambda - 1)\, \omega(e)$



ε-Balanced Graph and Hypergraph Partitioning

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

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

imbalance parameter

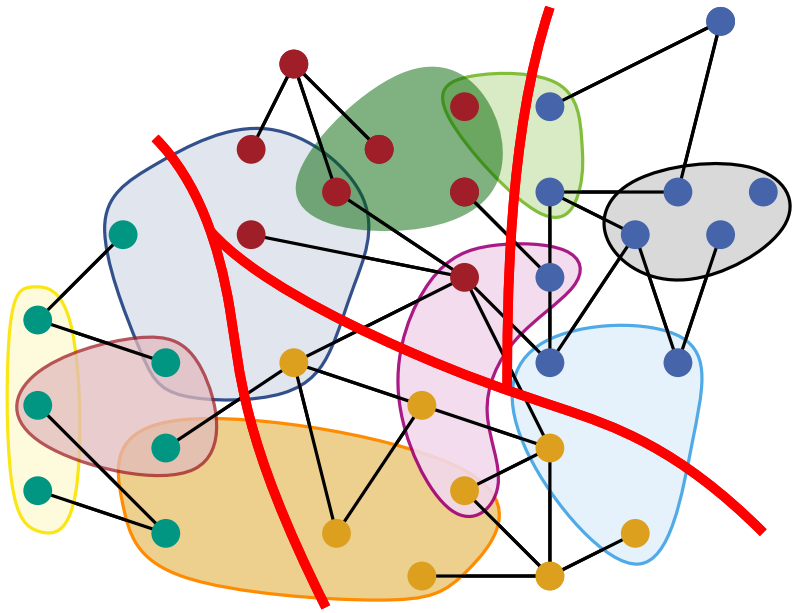
$$c(V_i) \leq (1 + \varepsilon) \left\lceil \frac{c(V)}{k} \right\rceil$$

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

Common Objectives:

- Graphs:
  - cut: $\sum_{e \in \text{cut}} \omega(e) = 17$
- Hypergraphs:
  - cut: $\sum_{e \in \text{cut}} \omega(e) = 5$
  - connectivity: $\sum_{e \in \text{cut}} (\lambda - 1)\, \omega(e)$

 # blocks connected by e



ε-Balanced Graph and Hypergraph Partitioning

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

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

imbalance parameter

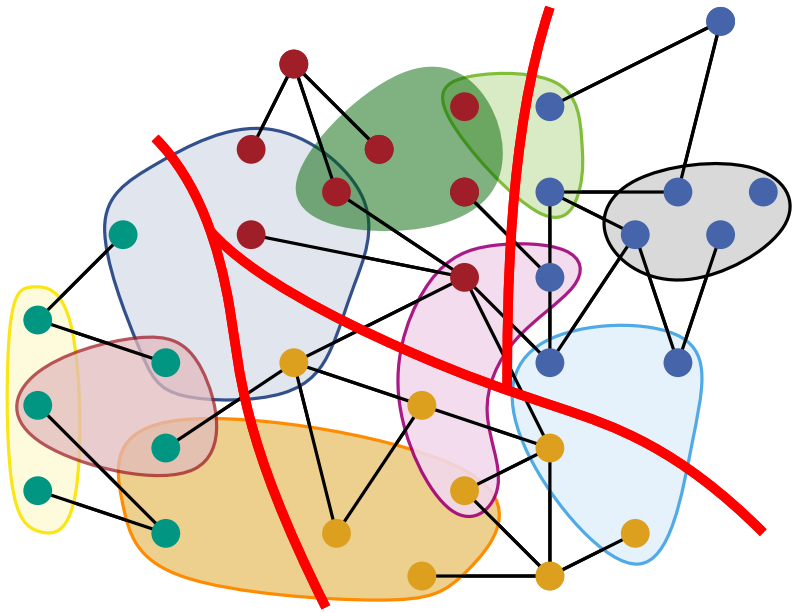
$$c(V_i) \leq (1 + \varepsilon) \left\lceil \frac{c(V)}{k} \right\rceil$$

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

Common Objectives:

- Graphs:
  - cut: $\sum_{e \in \text{cut}} \omega(e) = 17$
- Hypergraphs:
  - cut: $\sum_{e \in \text{cut}} \omega(e) = 5$
  - connectivity: $\sum_{e \in \text{cut}} (\lambda - 1)\, \omega(e) = 7$

 # blocks connected by e



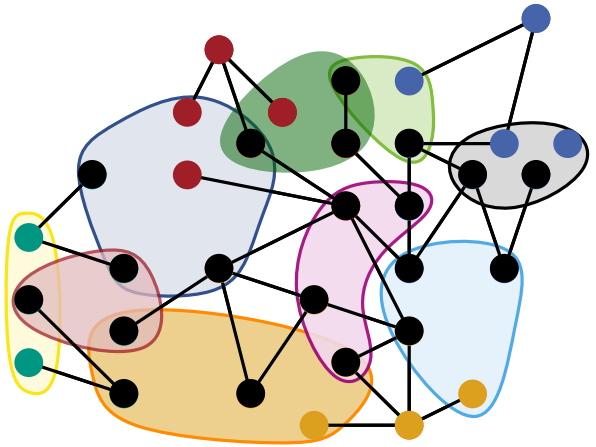
Institute of Theoretical Informatics
Algorithmics Group

Variants of Standard Hypergraph Partitioning

Possibly relevant/interesting variants:

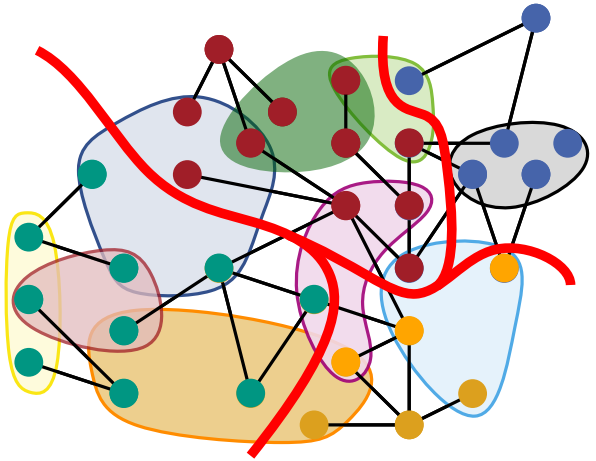
Partitioning with Fixed Vertices:

- some vertices are preassigned to blocks
- fixed vertices must remain in their block



Partitioning with Variable Block Weights:

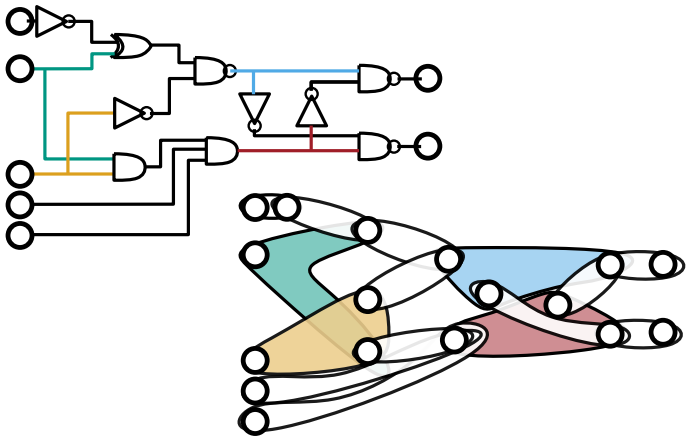
- individual block weights $U := \{U_1, \ldots, U_k\}$
- $\forall V_i : c(V_i) \leq U_i$



$U := \{12, 8, 11, 6\}$

Institute of Theoretical Informatics
Algorithmics Group

Applications



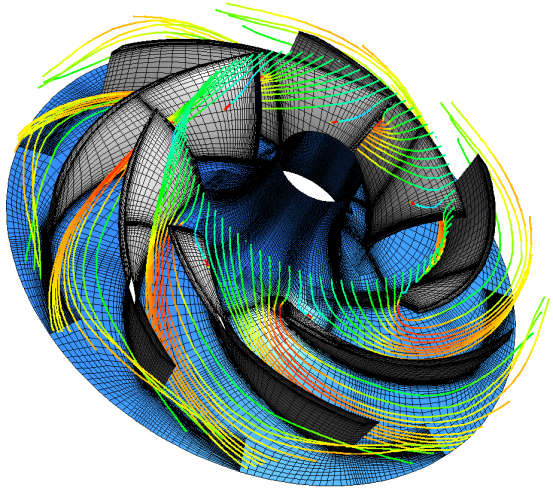
VLSI Design

Warehouse Optimization

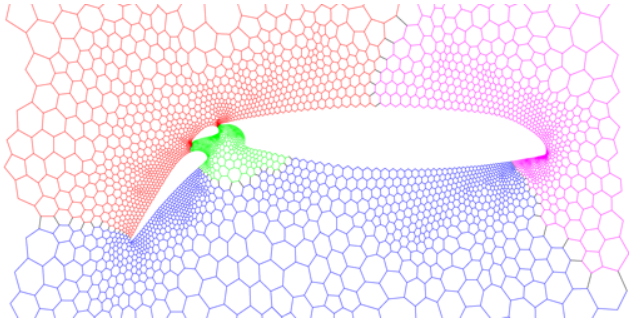
[Martin Grandjean, via Wikimedia Commons]

Complex Networks

Route Planning



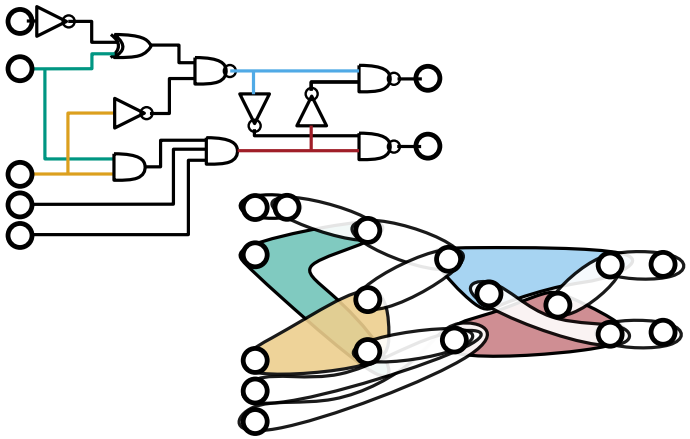
Simulation



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

Scientific Computing

Applications



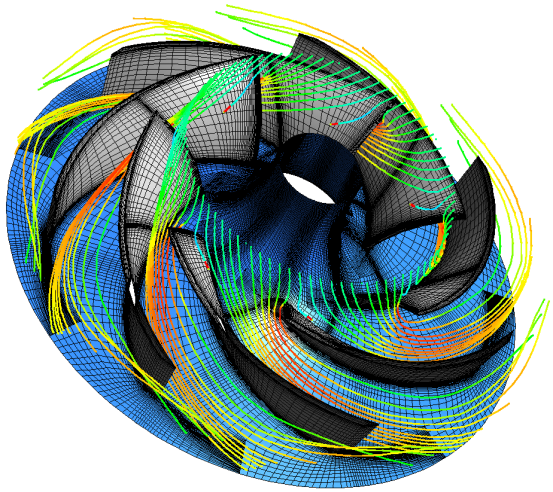
VLSI Design

Warehouse Optimization

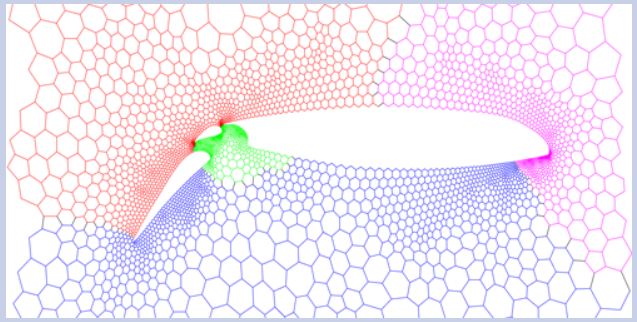
[Martin Grandjean, via Wikimedia Commons]

Complex Networks

Route Planning



Simulation

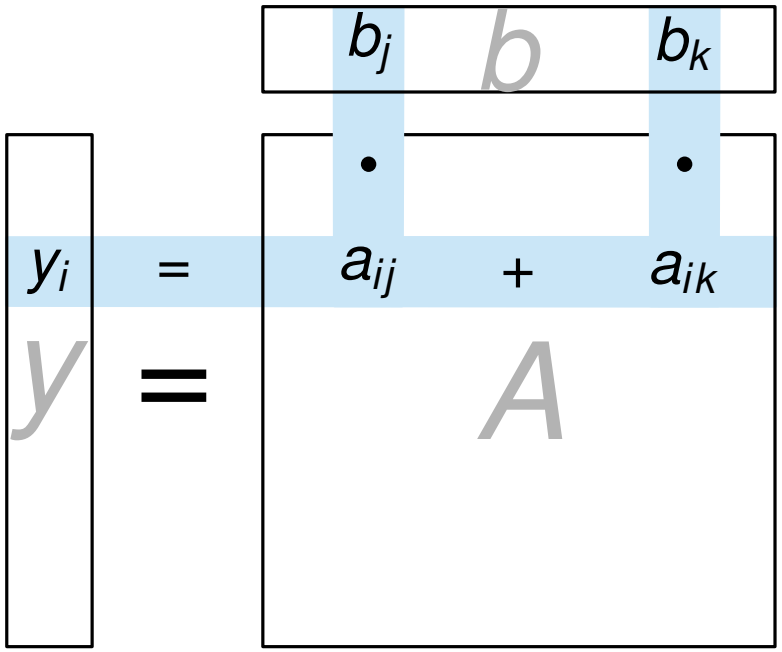


$$\mathbf{R}^{n \times n} \ni Ax = b \in \mathbf{R}^n$$

Scientific Computing

Parallel Sparse-Matrix Vector Product (SpM×V)

$$y = A\,b$$

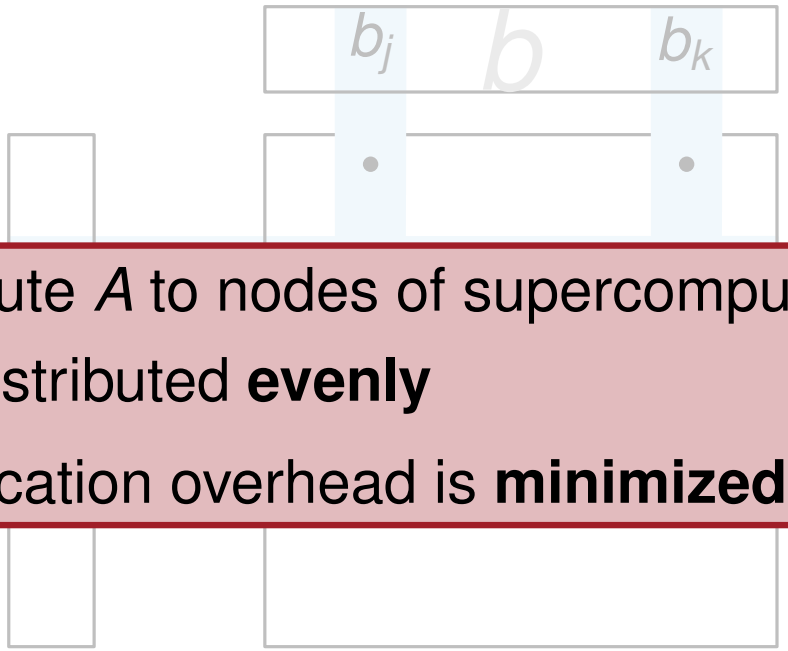


Setting:

- 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

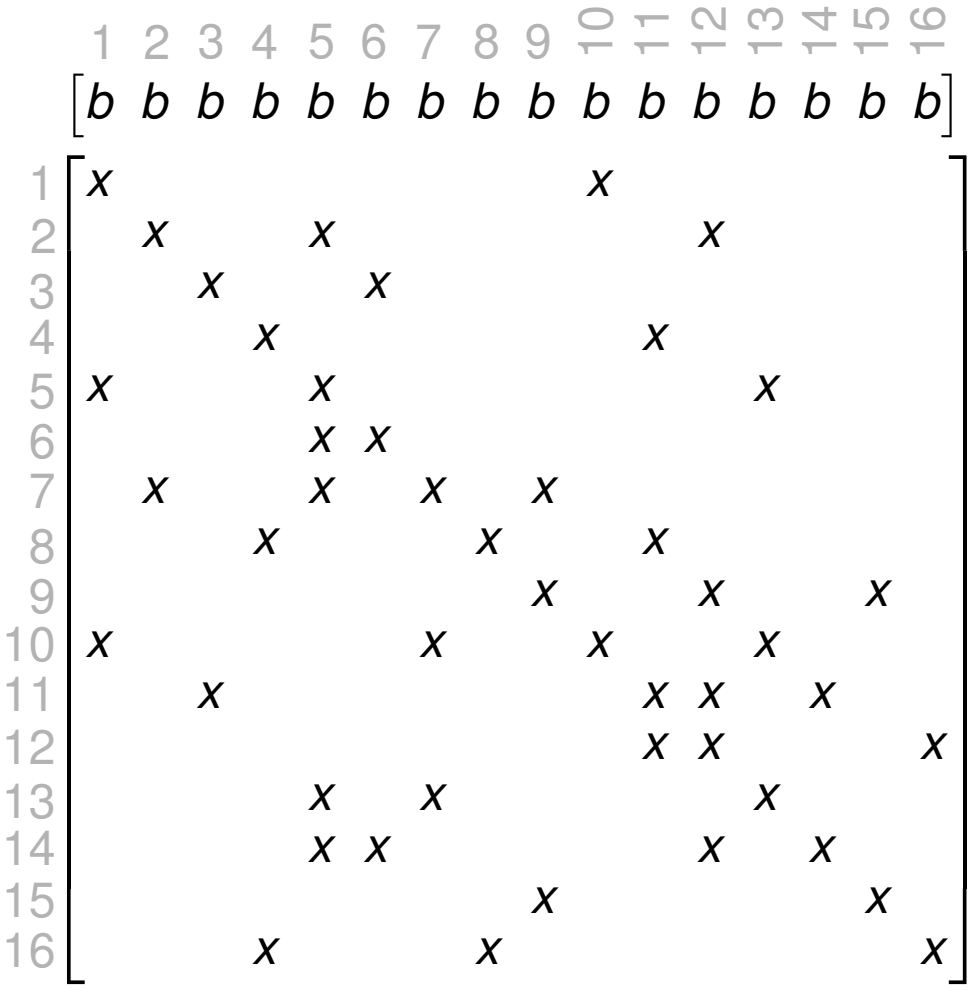
- work is distributed **evenly**
- communication overhead is **minimized**

Setting:

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

Naive Approach: Rowwise Decomposition

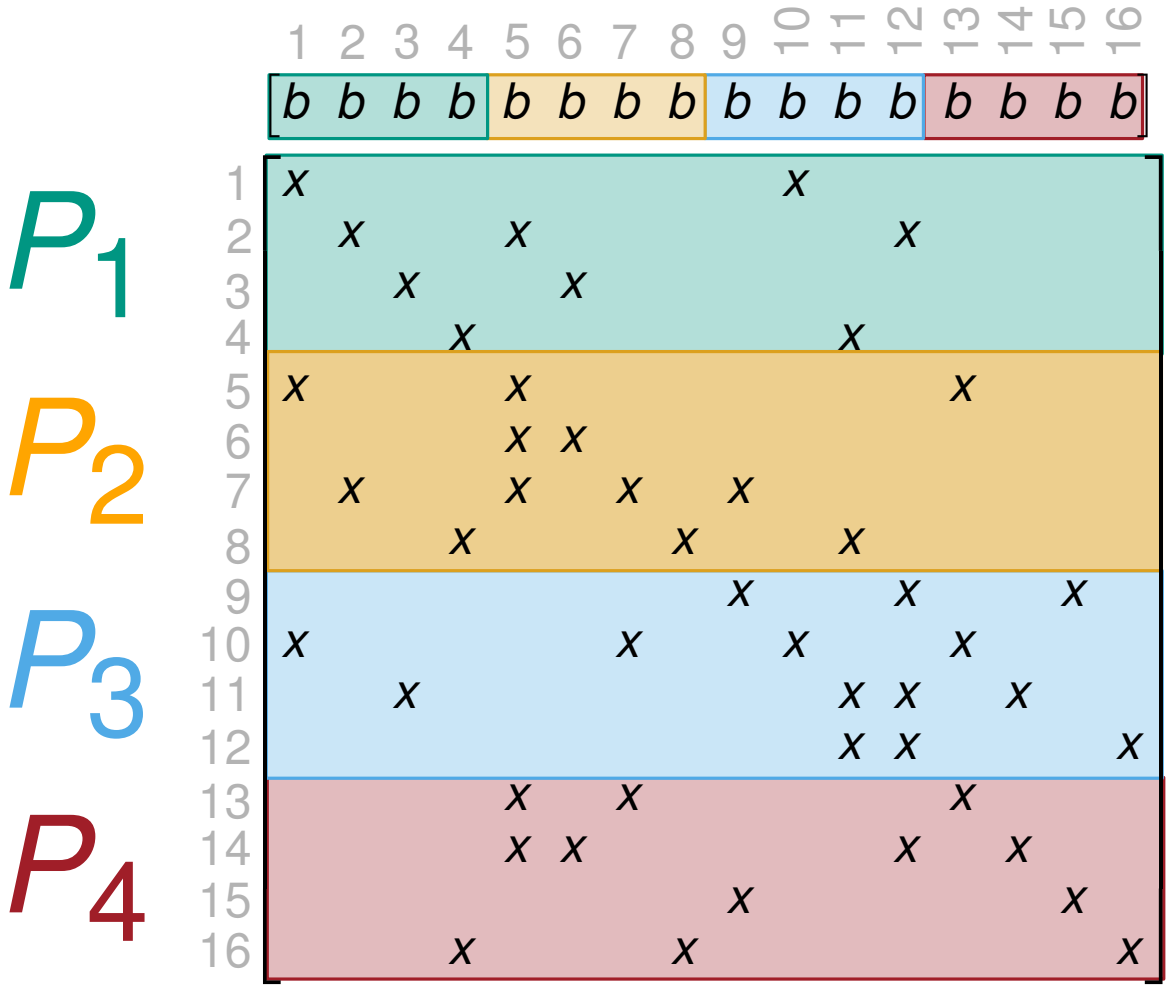
$A \in \mathbf{R}^{16 \times 16}$



Institute of Theoretical Informatics
Algorithmics Group

Naive Approach: Rowwise Decomposition

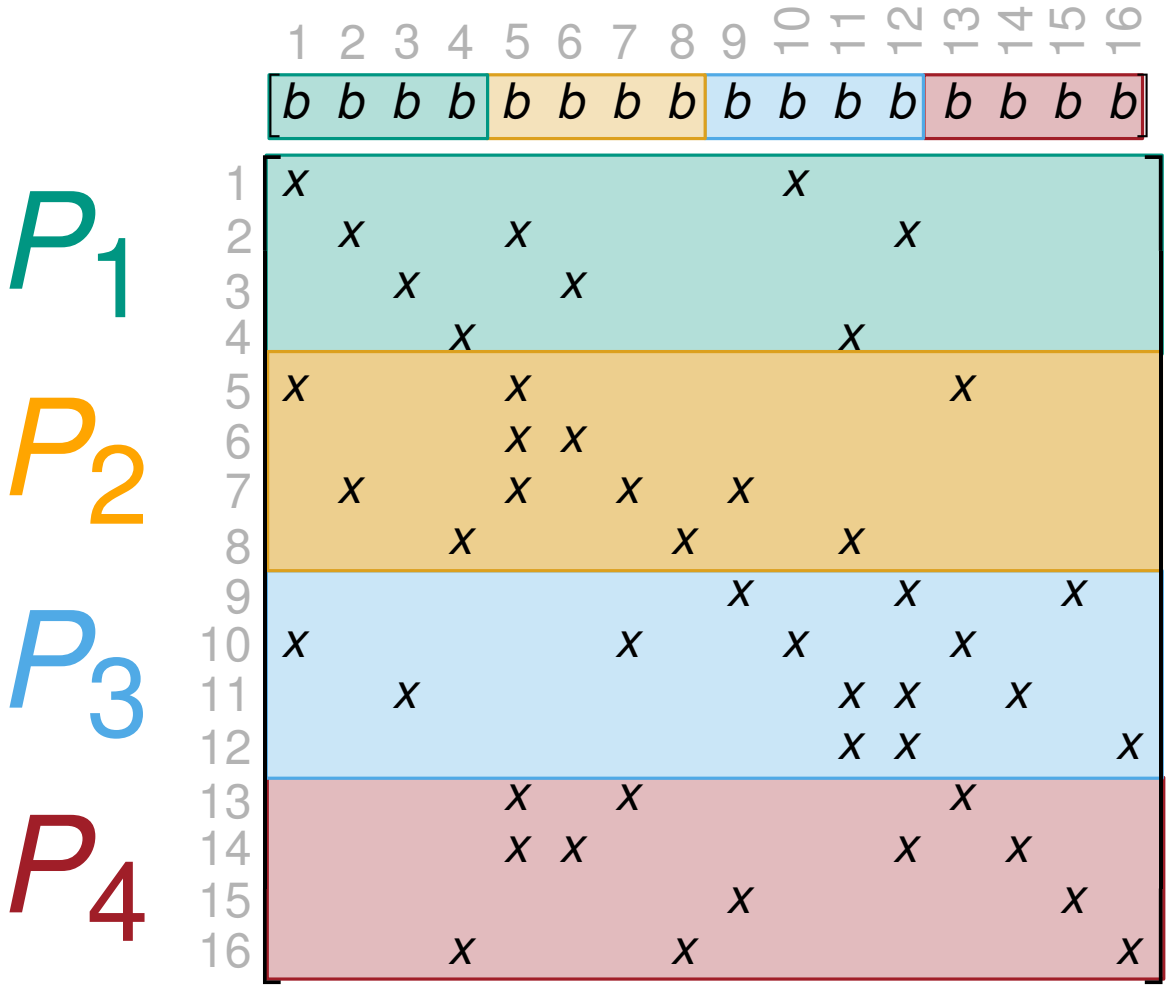
$A \in \mathbf{R}^{16 \times 16}$



Institute of Theoretical Informatics
Algorithmics Group

Naive Approach: Rowwise Decomposition

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



Load Balancing?

$\Rightarrow 9$

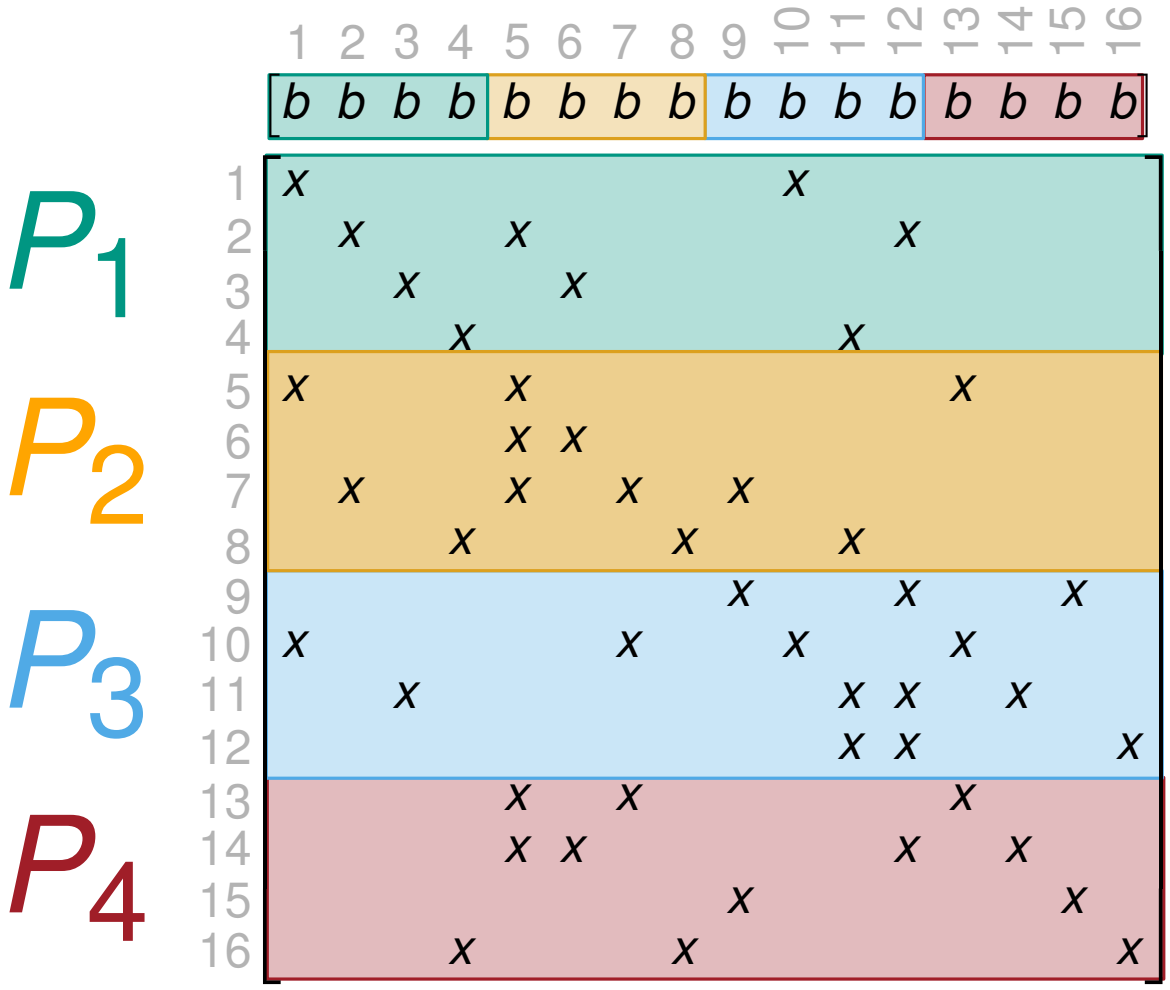
$\Rightarrow 12$

$\Rightarrow 14$

$\Rightarrow 12$

Naive Approach: Rowwise Decomposition

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



Load Balancing?

$\Rightarrow 9$

$\Rightarrow 12$

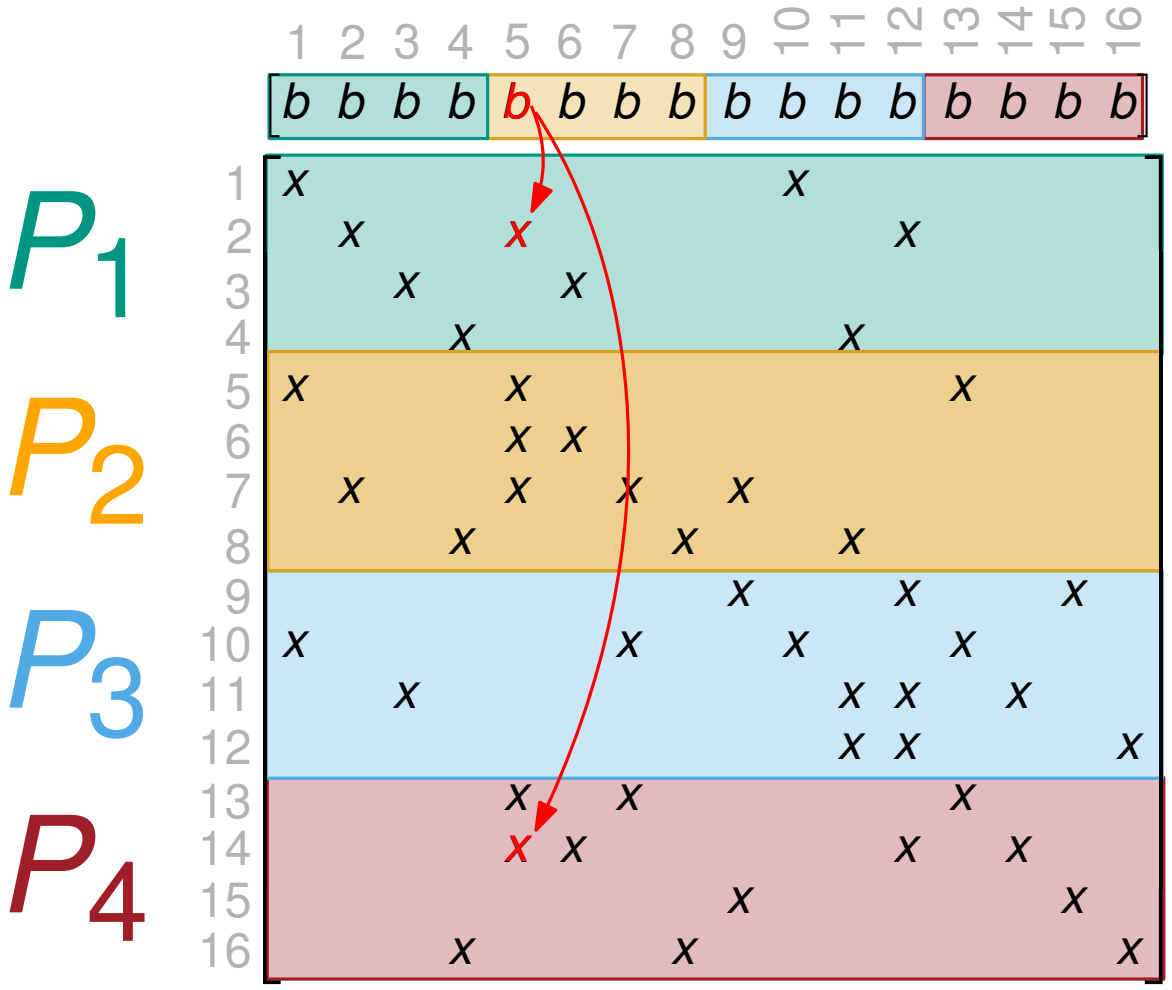
$\Rightarrow 14$

$\Rightarrow 12$

Commuication Volume?

Naive Approach: Rowwise Decomposition

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



Load Balancing?

$\Rightarrow 9$

$\Rightarrow 12$

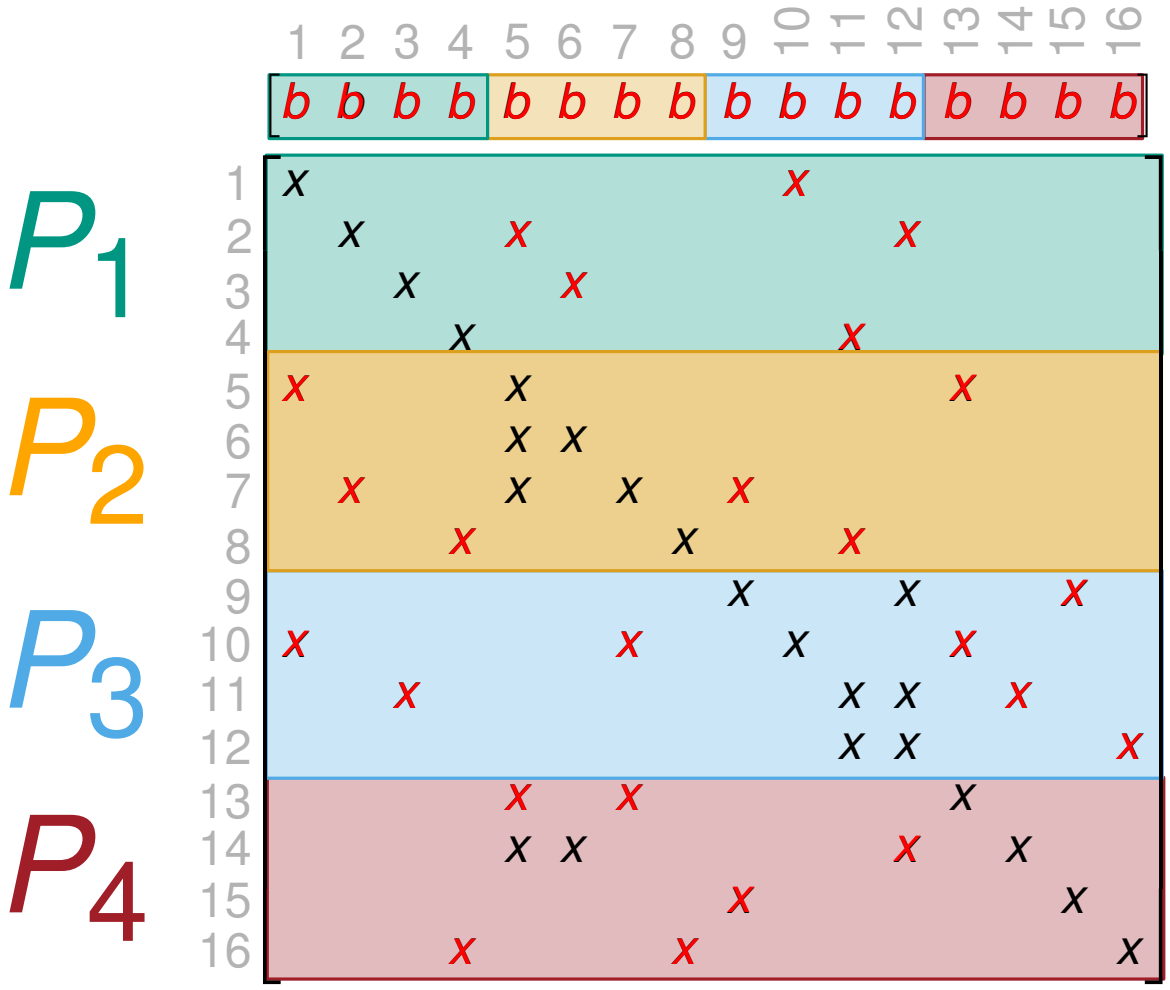
$\Rightarrow 14$

$\Rightarrow 12$

Commuication Volume?

Naive Approach: Rowwise Decomposition

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



Load Balancing?

$\Rightarrow 9$

$\Rightarrow 12$

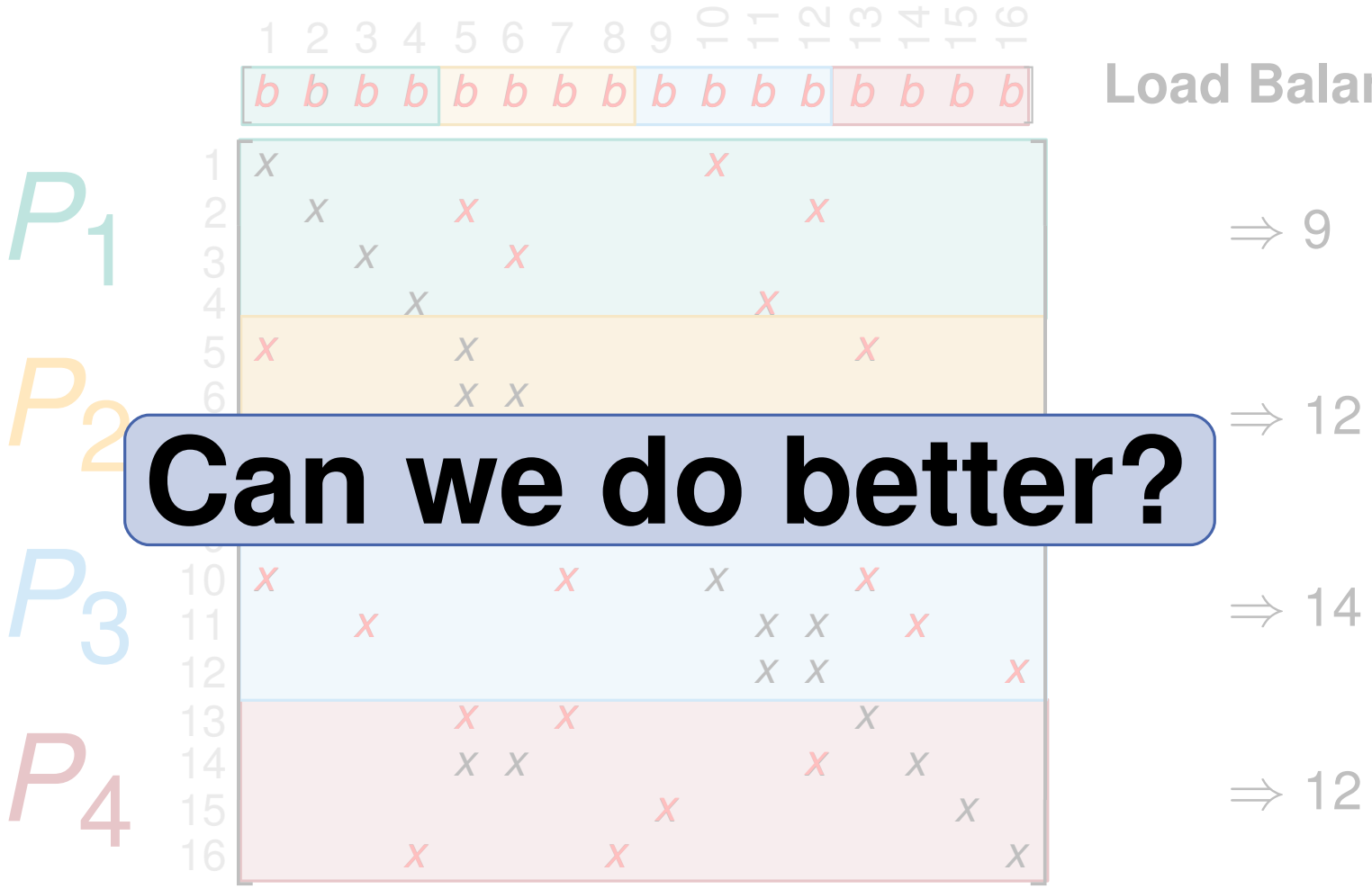
$\Rightarrow 14$

$\Rightarrow 12$

Commuication Volume? $\Rightarrow$ 24 entries!

Naive Approach: Rowwise Decomposition

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



Can we do better?

Commuication Volume? ⇒ 24 entries!

Institute of Theoretical Informatics
Algorithmics Group

From SpM×V to Hypergraph Partitioning

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

- one vertex per row:

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

- one hyperedge per column:

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

	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

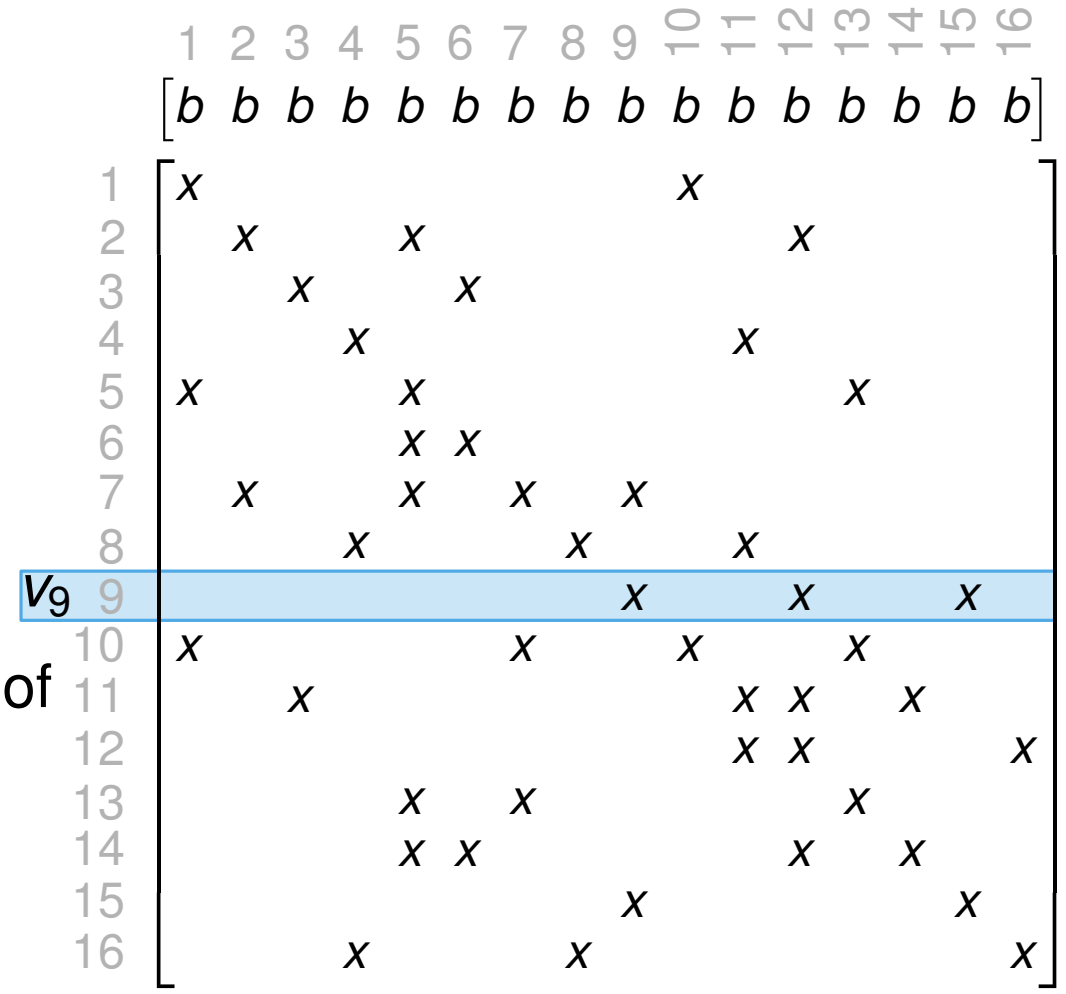
From SpM×V to Hypergraph Partitioning

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

- 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



Institute of Theoretical Informatics
Algorithmics Group

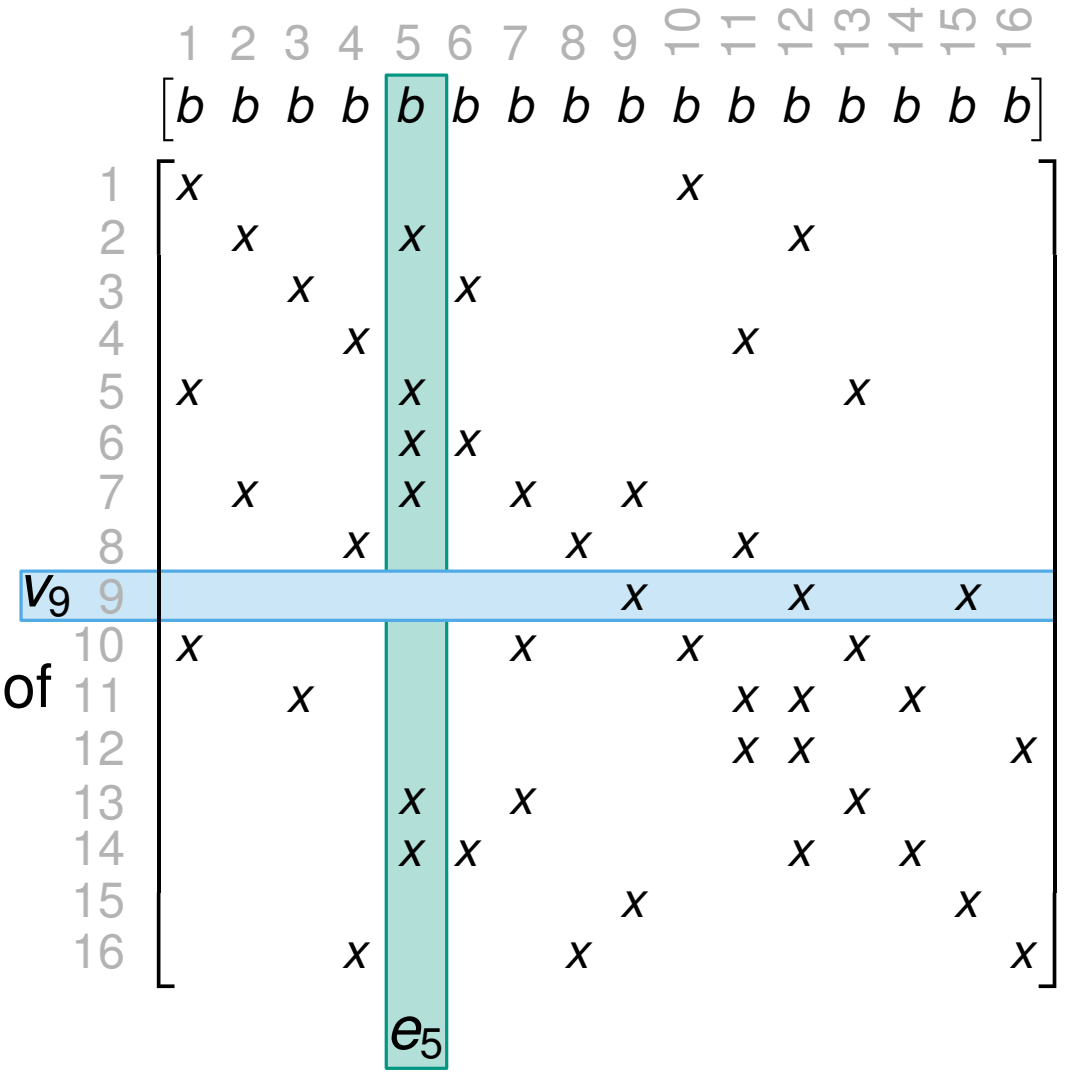
From SpM×V to Hypergraph Partitioning

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

- 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



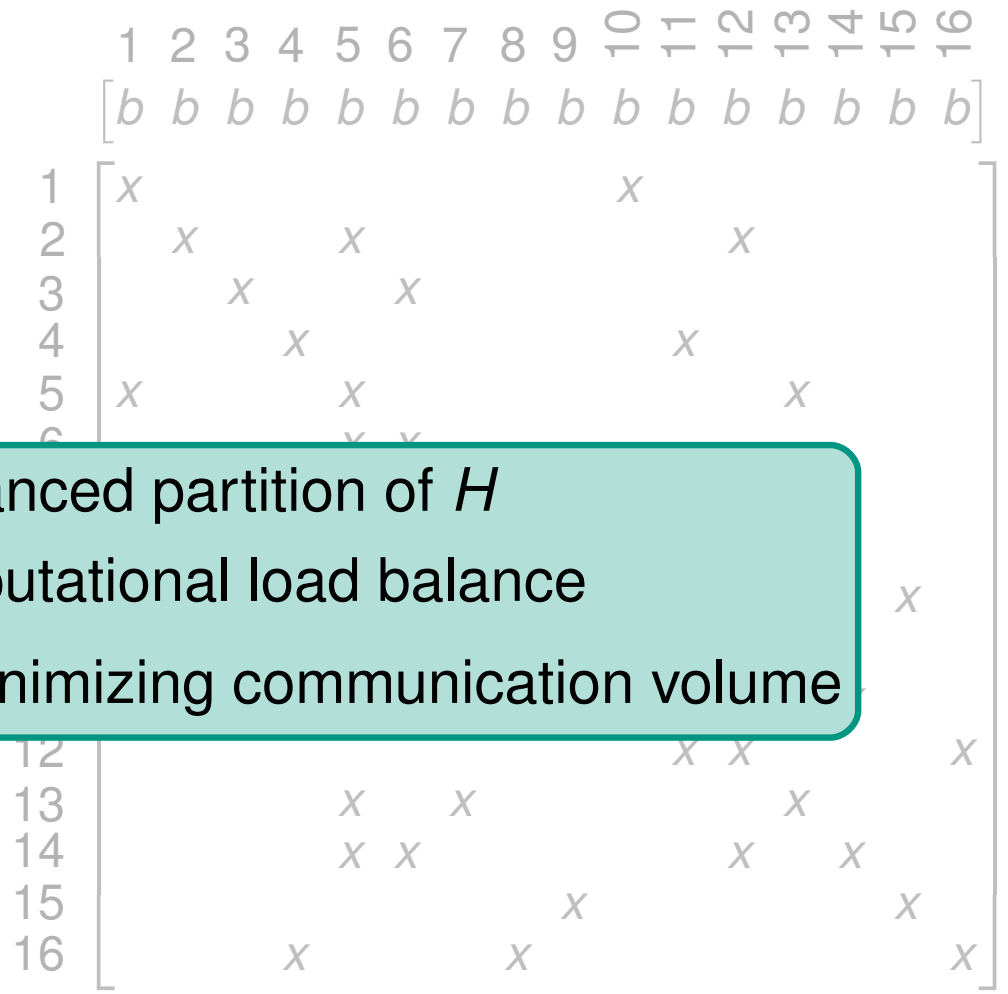
$e_j \in E_C$: set of vertices that need b_j

Institute of Theoretical Informatics
Algorithmics Group

From SpM×V to Hypergraph Partitioning

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

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

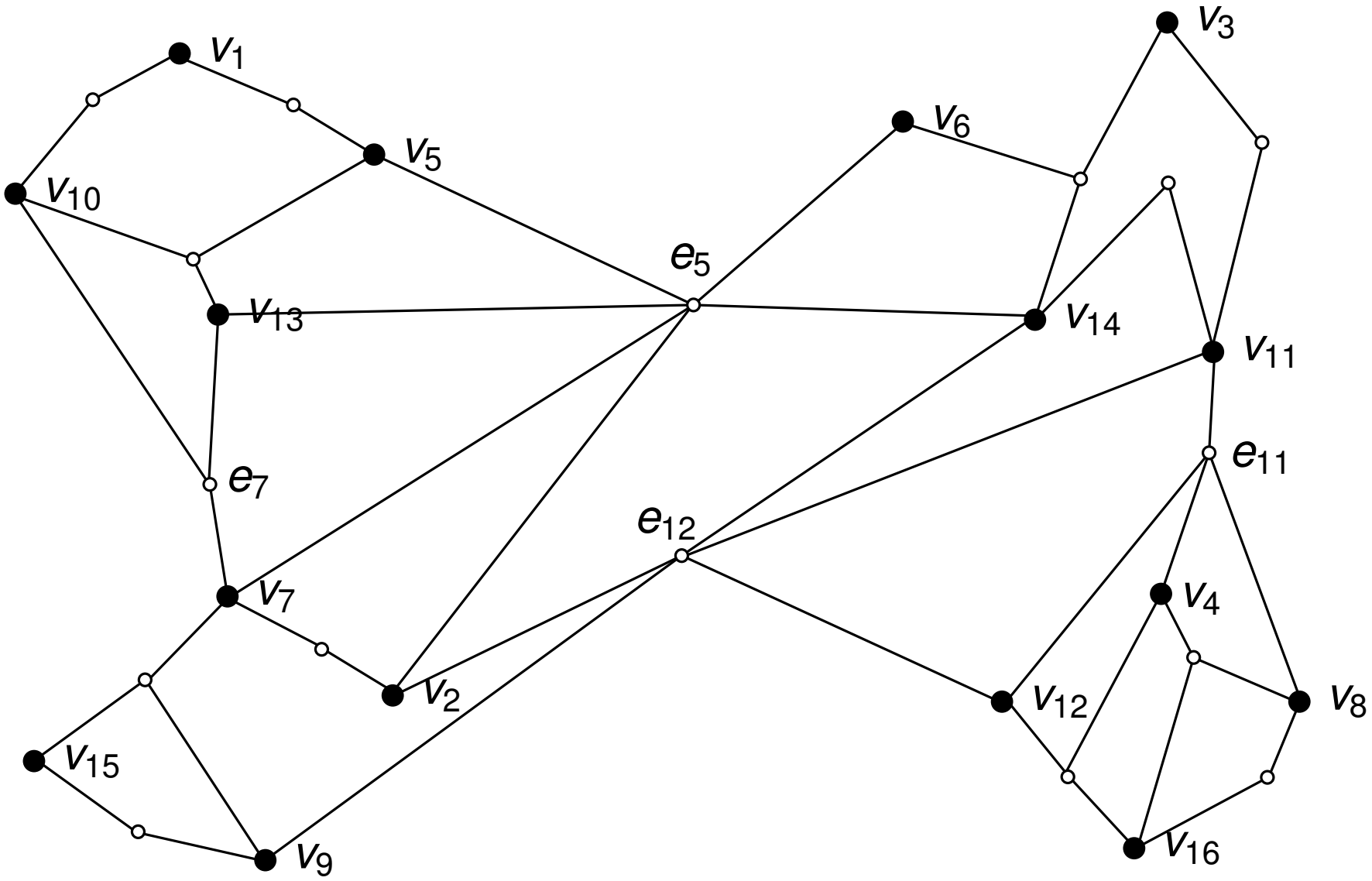


Solution: ε-balanced partition of H

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

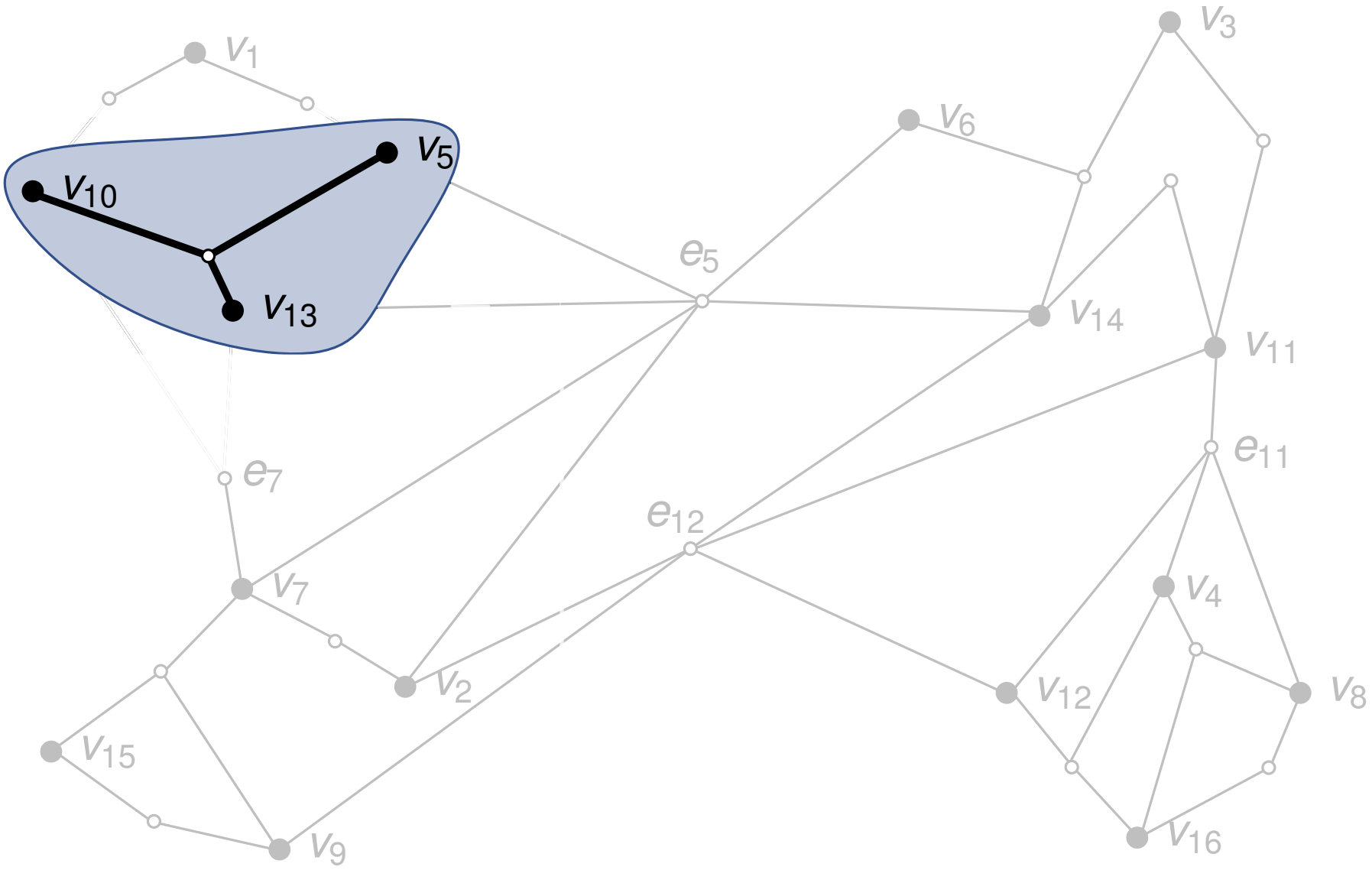
Institute of Theoretical Informatics
Algorithmics Group

From SpM$\times$V to Hypergraph Partitioning



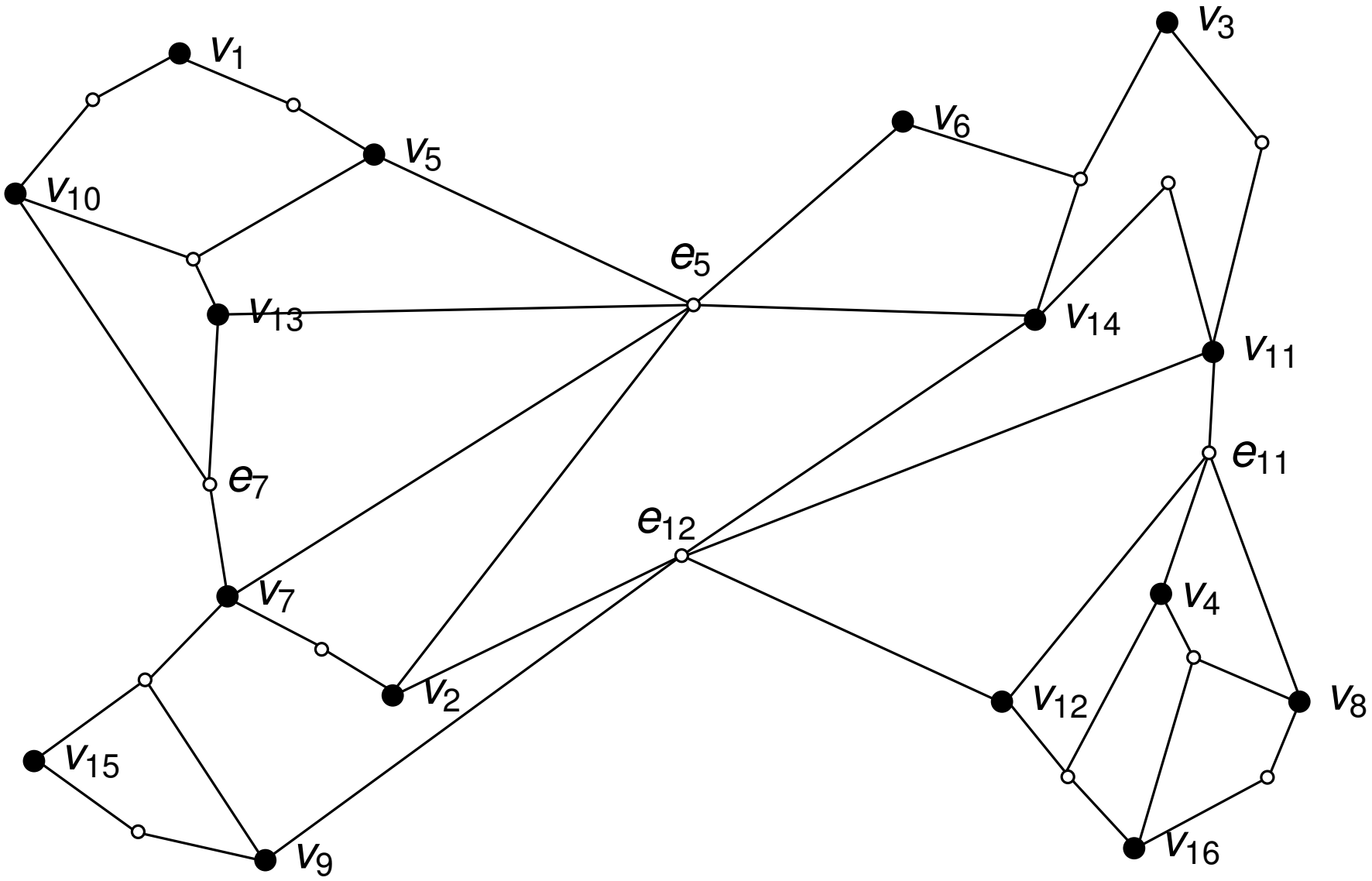
Institute of Theoretical Informatics
Algorithmics Group

From SpM×V to Hypergraph Partitioning



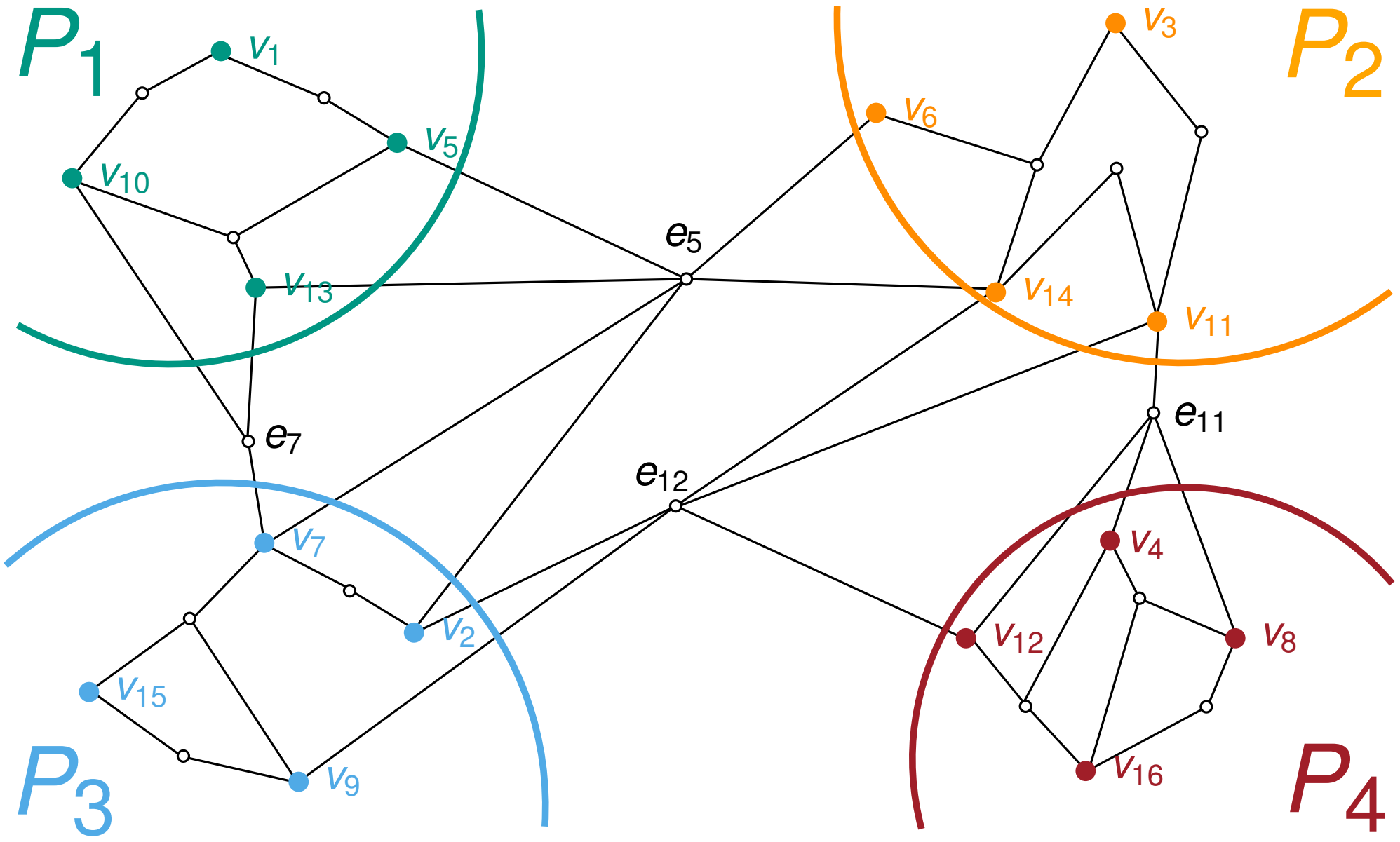
Institute of Theoretical Informatics
Algorithmics Group

From SpM$\times$V to Hypergraph Partitioning



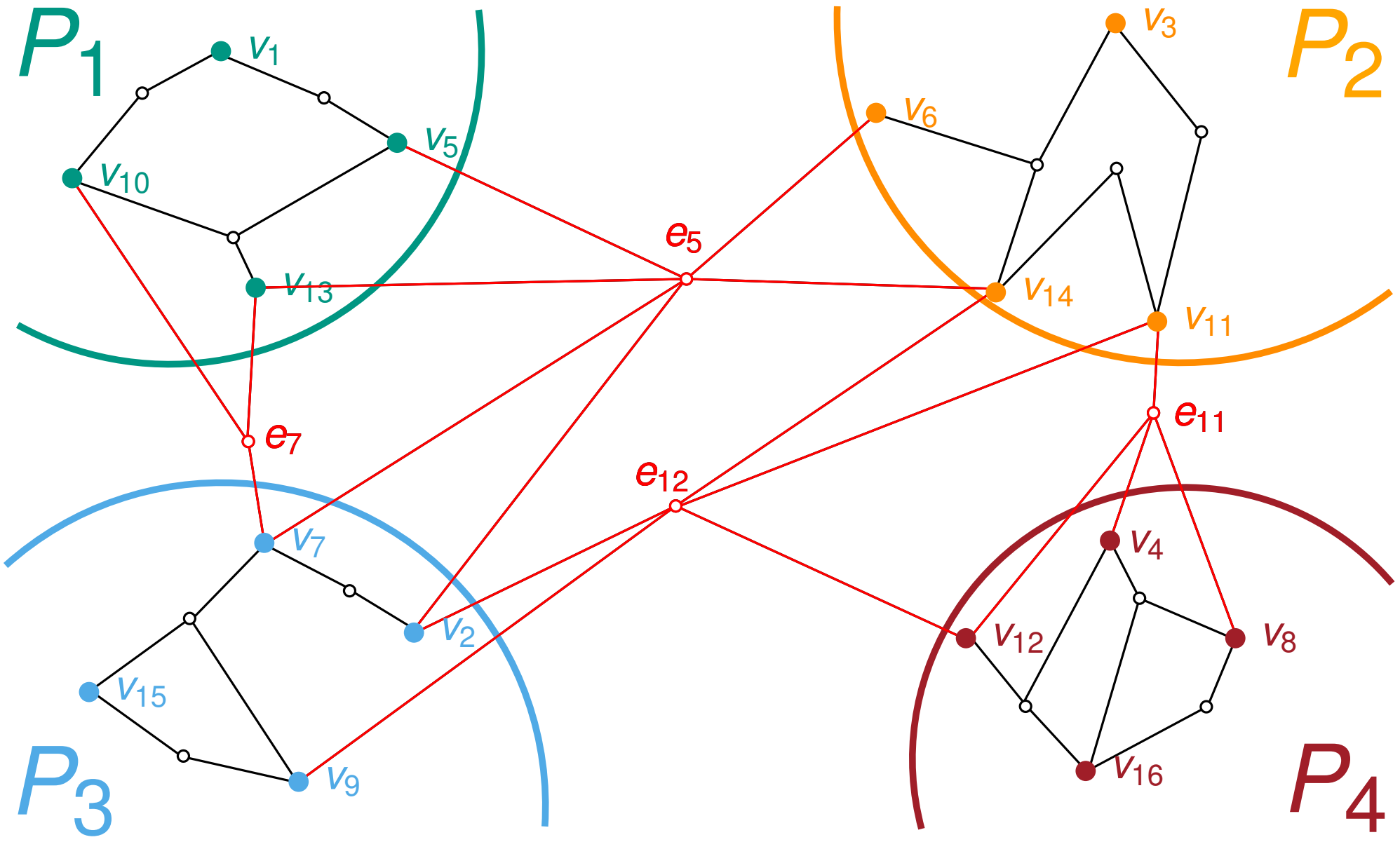
Institute of Theoretical Informatics
Algorithmics Group

From SpM×V to Hypergraph Partitioning



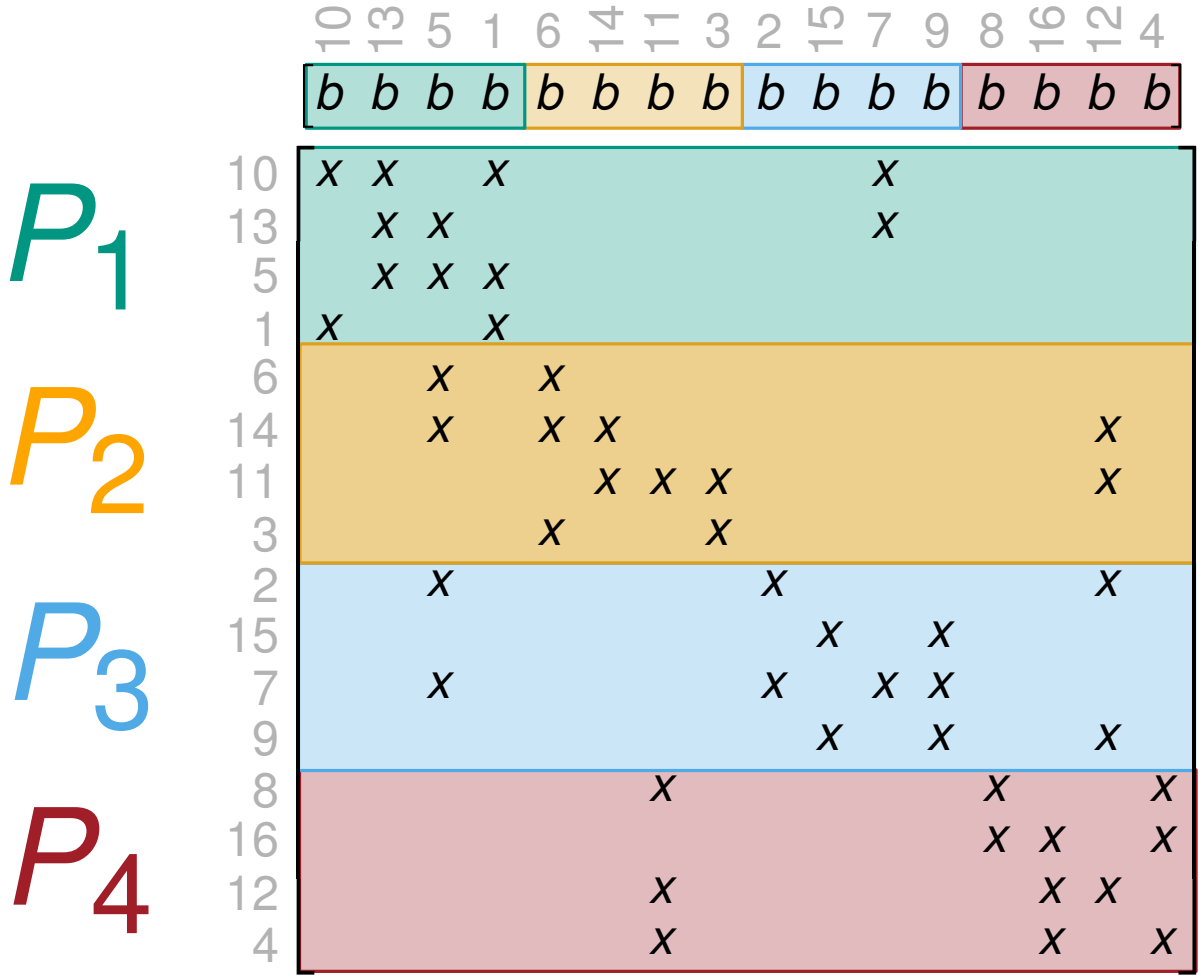
Institute of Theoretical Informatics
Algorithmics Group

From SpM×V to Hypergraph Partitioning



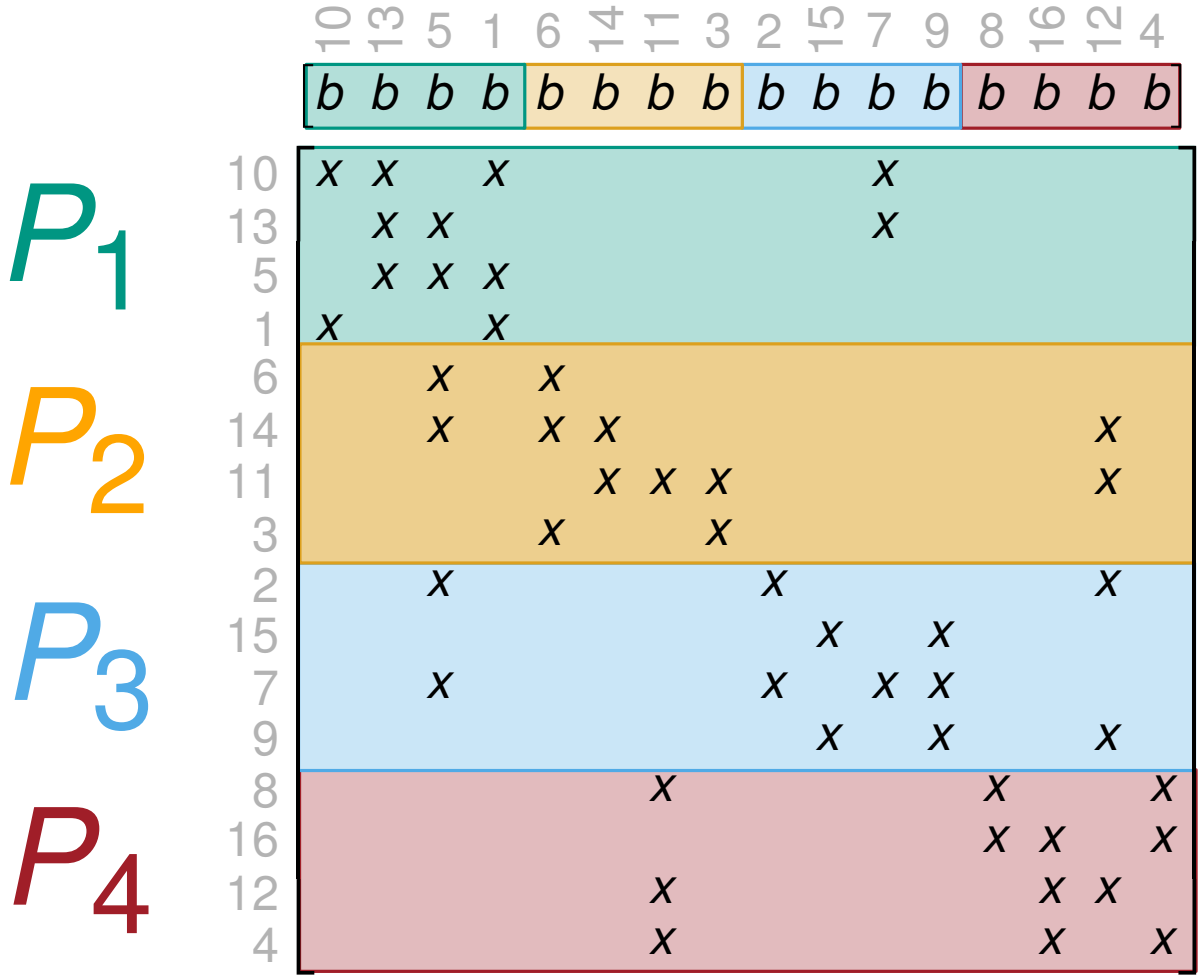
Institute of Theoretical Informatics
Algorithmics Group

From Hypergraph Partitioning to SpM×V



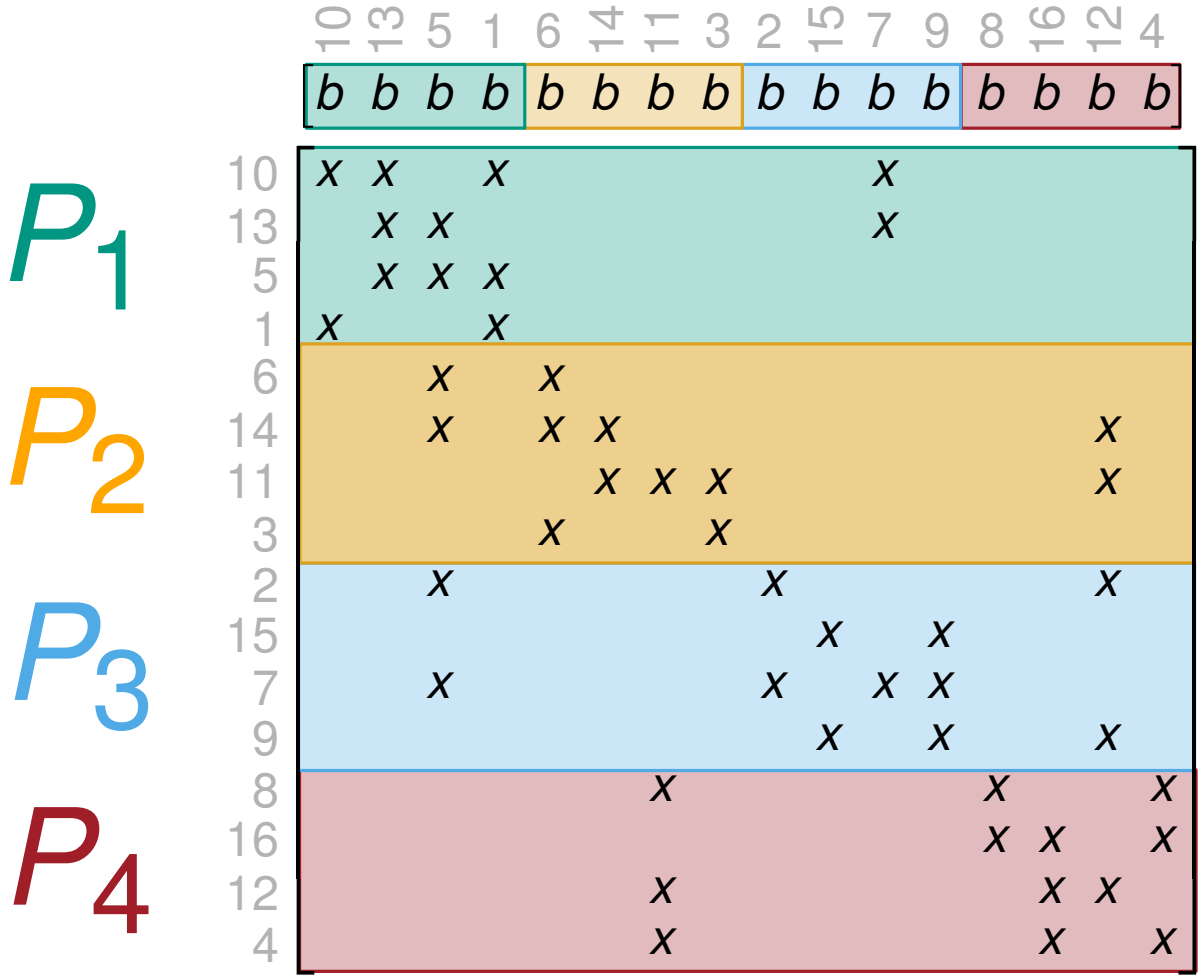
Institute of Theoretical Informatics
Algorithmics Group

From Hypergraph Partitioning to SpM×V



Load Balancing?

From Hypergraph Partitioning to SpM×V



Load Balancing?

$\Rightarrow 12$

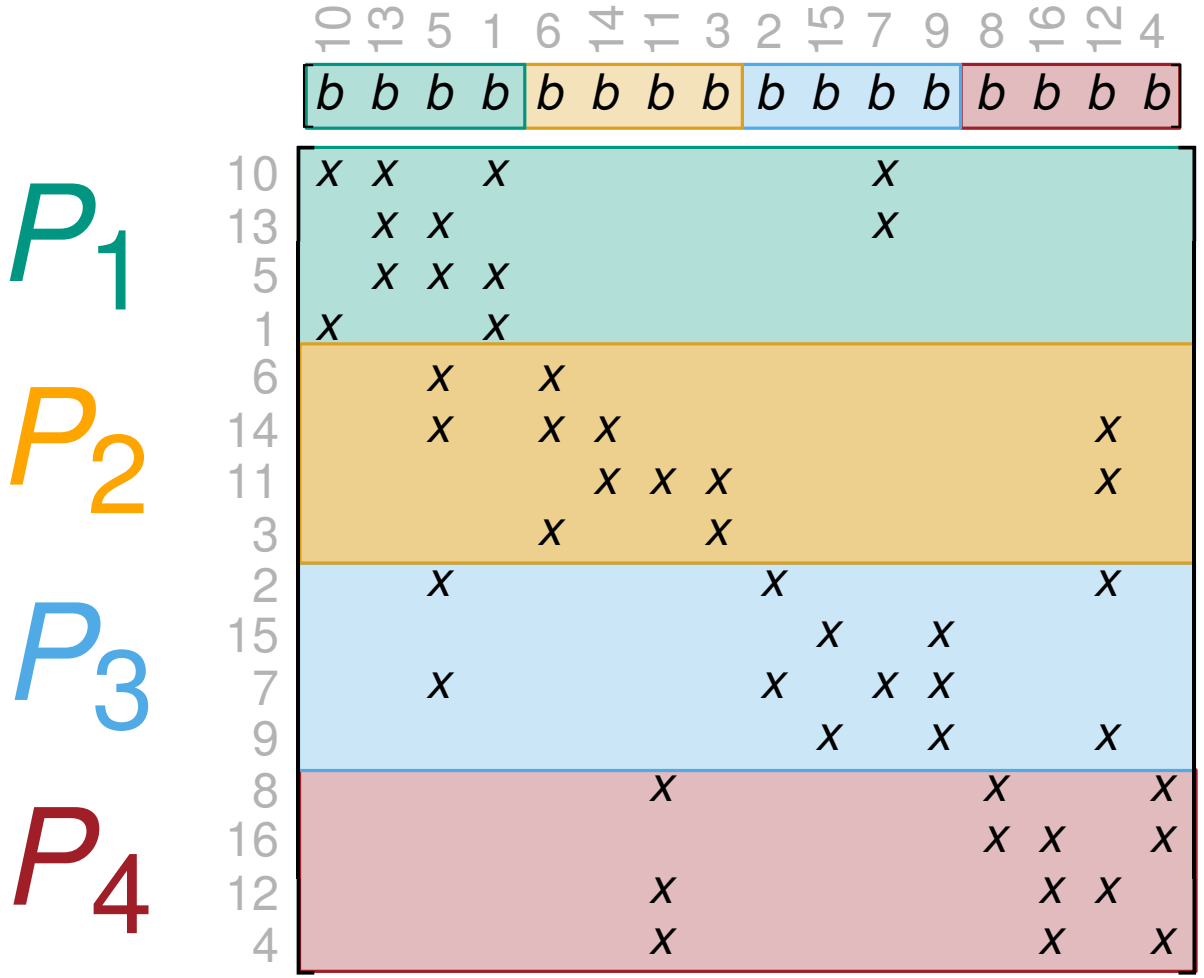
$\Rightarrow 12$

$\Rightarrow 12$

$\Rightarrow 12$

From Hypergraph Partitioning to SpM×V

Where are the cut-hyperedges?



Load Balancing?

$\Rightarrow 12$

$\Rightarrow 12$

$\Rightarrow 12$

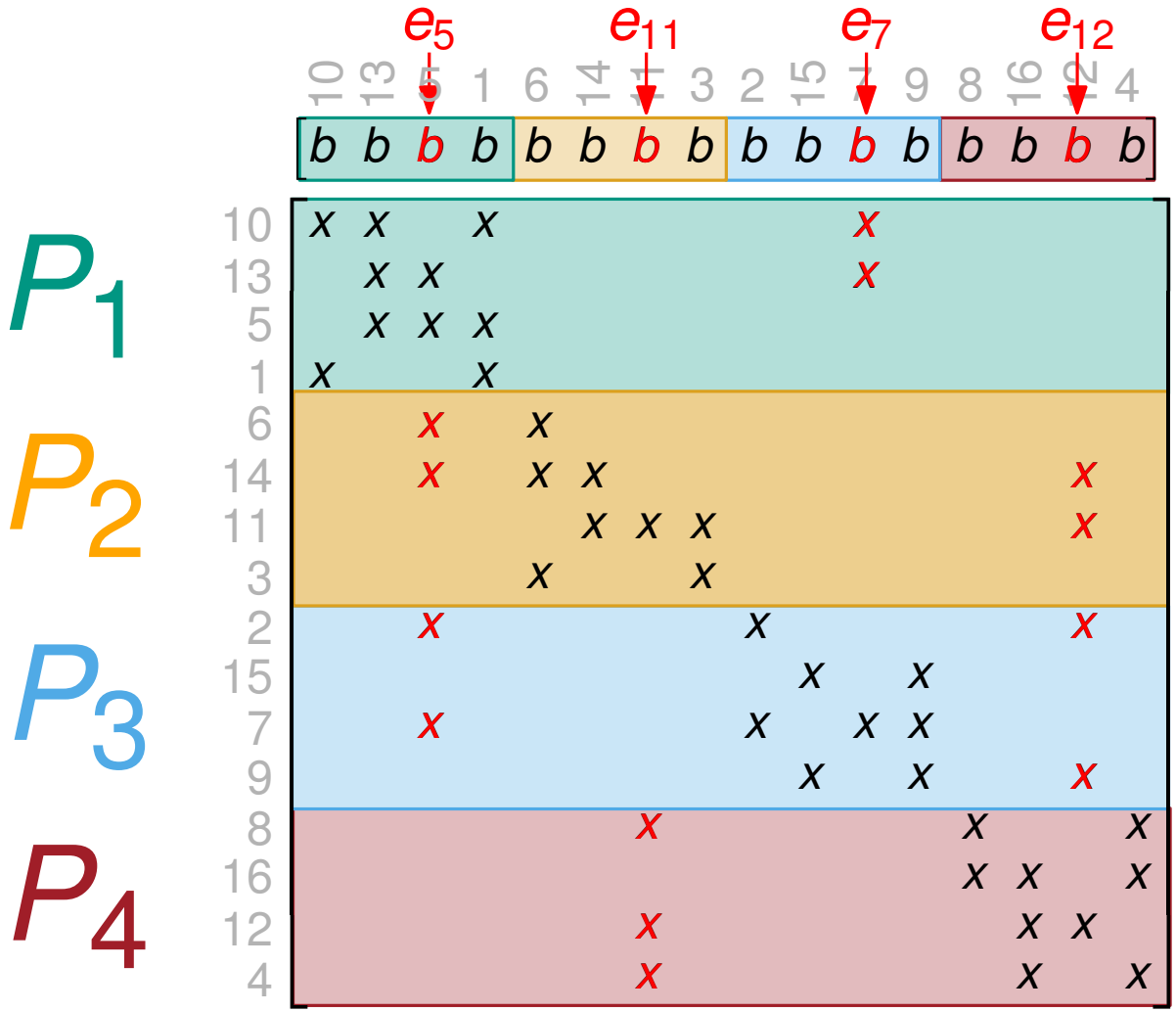
$\Rightarrow 12$

Commuication Volume?

From Hypergraph Partitioning to SpM×V

Karlsruhe Institute of Technology

Where are the cut-hyperedges?



Load Balancing?

$\Rightarrow$ 12

$\Rightarrow$ 12

$\Rightarrow$ 12

$\Rightarrow$ 12

Commuication Volume? $\Rightarrow$ 6 entries!

How does Hypergraph Partitioning work?

Institute of Theoretical Informatics
Algorithmics Group

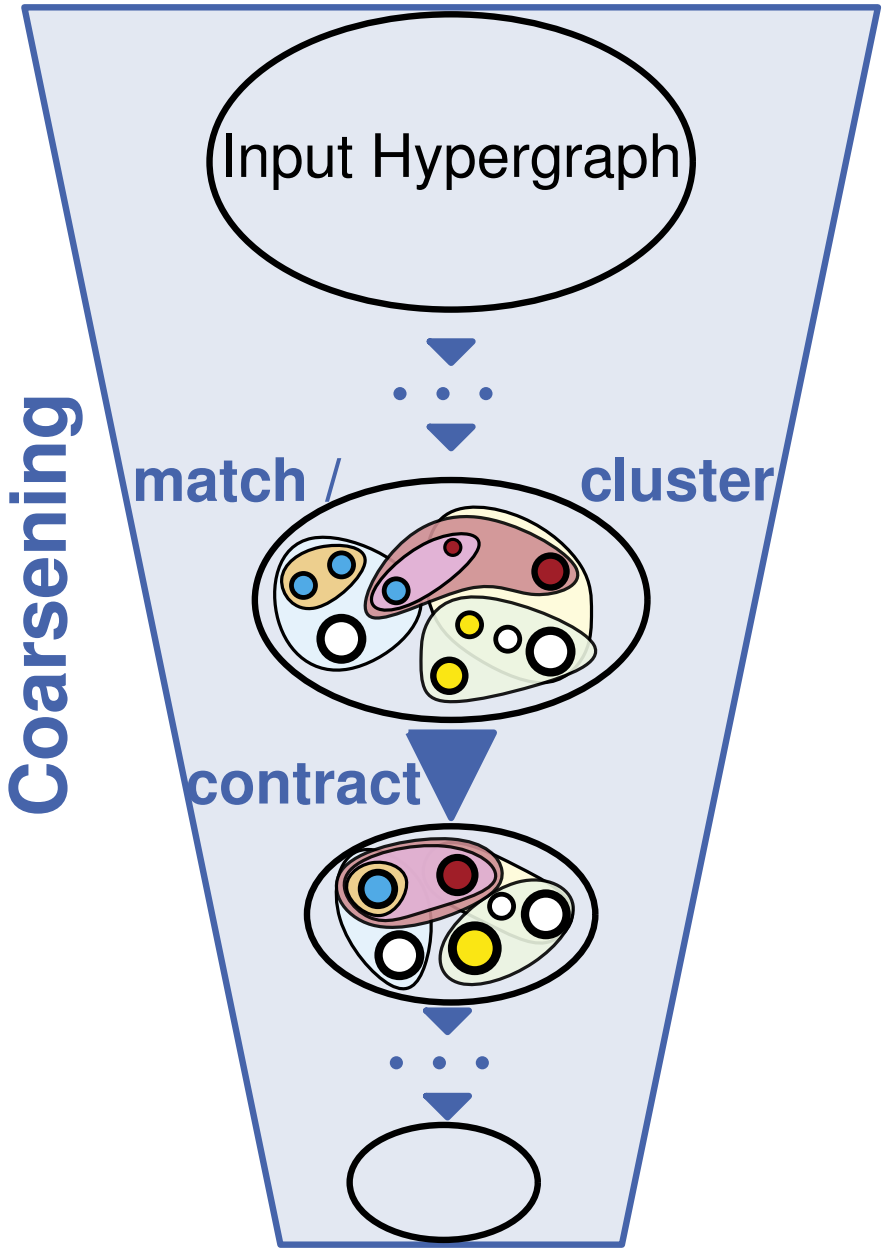
How does

Bad News:

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

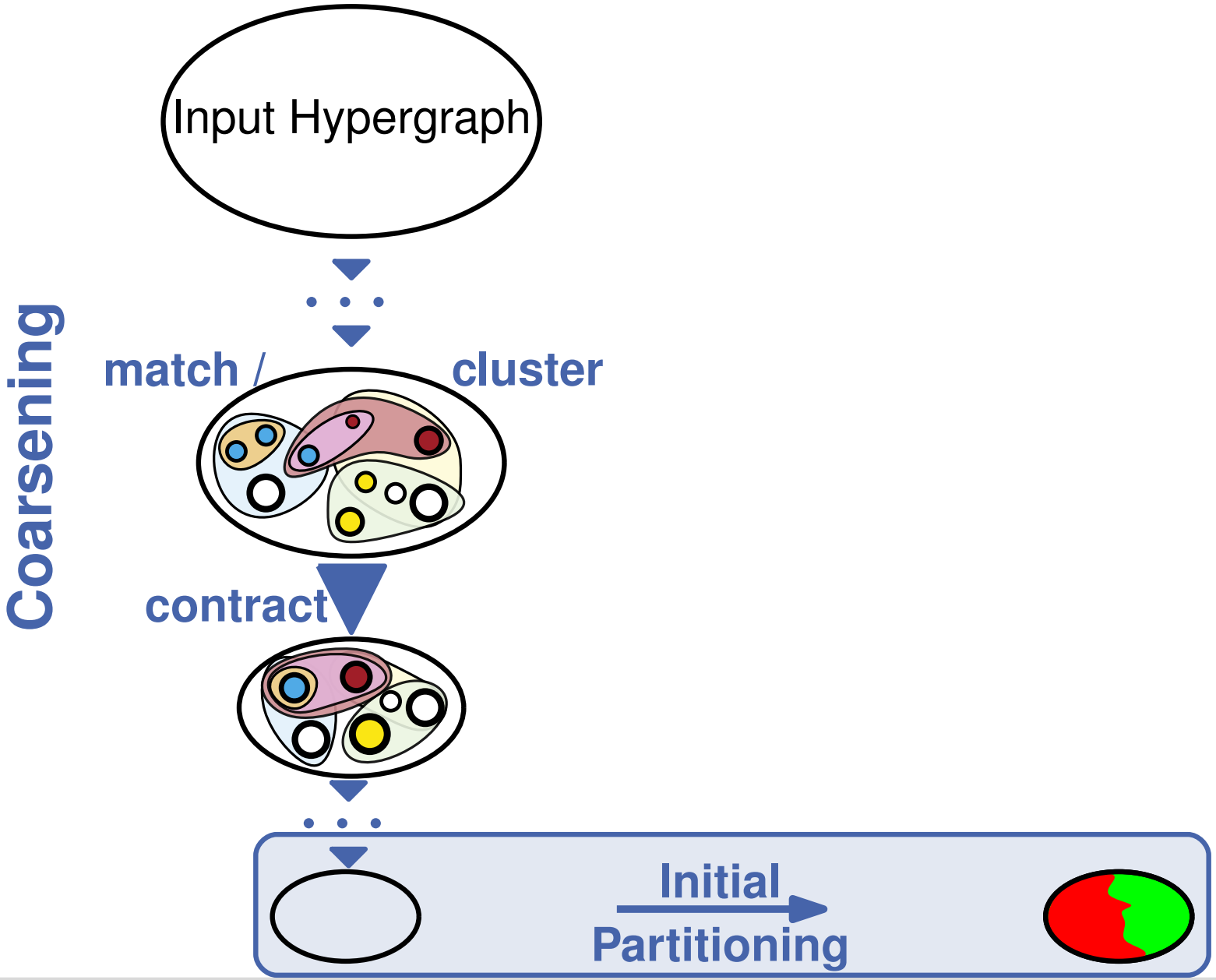
work?

Successful Heuristic: Multilevel Paradigm

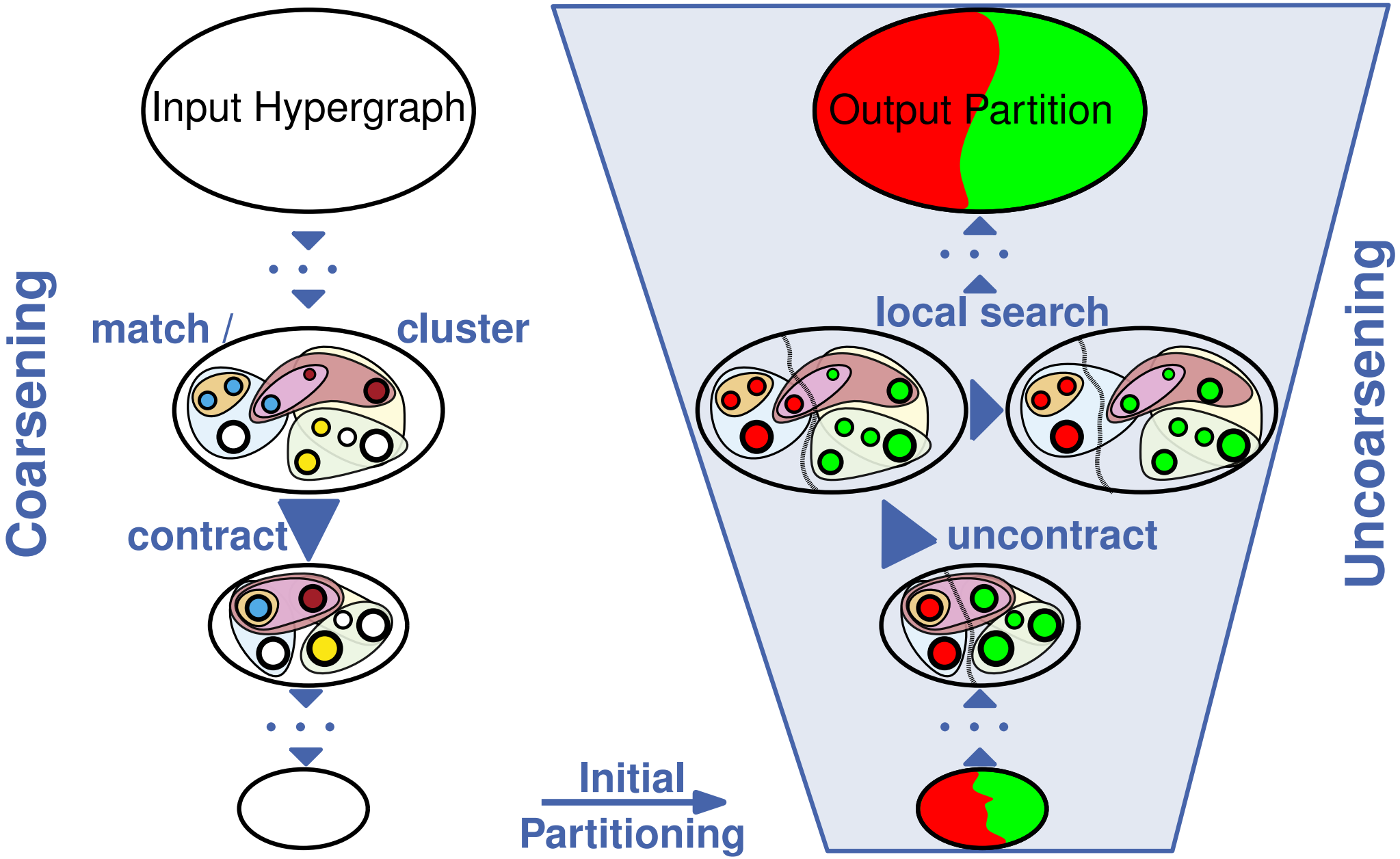


Institute of Theoretical Informatics
Algorithmics Group

Successful Heuristic: Multilevel Paradigm



Successful Heuristic: Multilevel Paradigm



Institute of Theoretical Informatics
Algorithmics Group

Coarsening

Clustering-based Coarsening

Common Strategy: **avoid** global decisions $\rightsquigarrow$ **local**, greedy algorithms

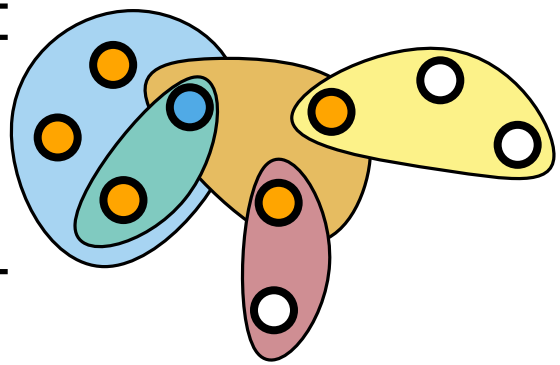
Objective: identify highly connected vertices

using...

foreach *vertex* v **do**

cluster[v] := argmax `rating`(v,u)

neighbor u



Clustering-based Coarsening

Common Strategy: **avoid** global decisions $\rightsquigarrow$ **local**, greedy algorithms

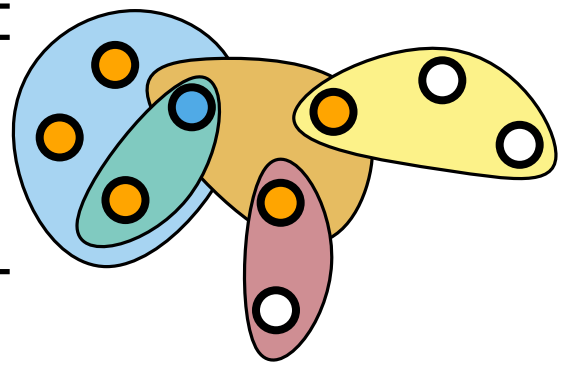
Objective: identify highly connected vertices

using...

foreach *vertex* v **do**

  cluster[v] := argmax `rating`(v,u)

  neighbor u



Main Design Goals: [Karypis,Kumar 99]

1: reduce **size** of nets $\rightsquigarrow$ easier local search

2: reduce **number** of nets $\rightsquigarrow$ easier initial partitioning

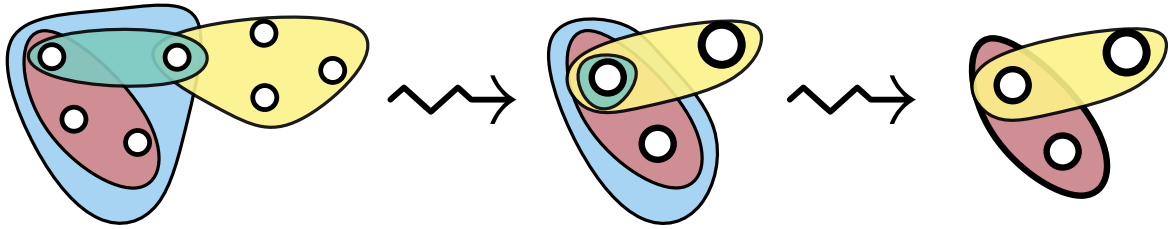
3: maintain **structural similarity** $\rightsquigarrow$ good coarse solutions

Clustering-based Coarsening

Main Design Goals:

1: reduce **size** of nets $\rightsquigarrow$ easier local search

2: reduce **number** of nets $\rightsquigarrow$ easier initial partitioning

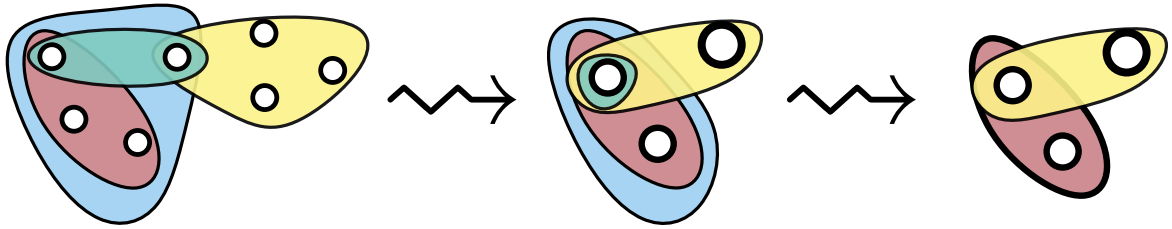


Clustering-based Coarsening

Main Design Goals:

1: reduce **size** of nets $\rightsquigarrow$ easier local search

2: reduce **number** of nets $\rightsquigarrow$ easier initial partitioning



$\Longrightarrow$ hypergraph-tailored rating functions:

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

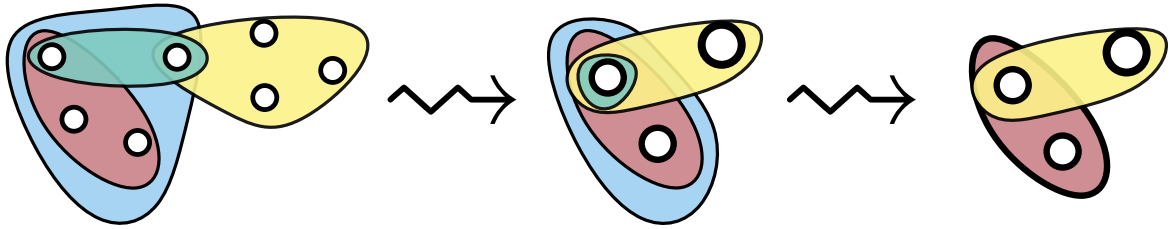
Institute of Theoretical Informatics
Algorithmics Group

Clustering-based Coarsening

Main Design Goals:

1: reduce **size** of nets $\rightsquigarrow$ easier local search

2: reduce **number** of nets $\rightsquigarrow$ easier initial partitioning



$\Longrightarrow$ hypergraph-tailored rating functions:

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

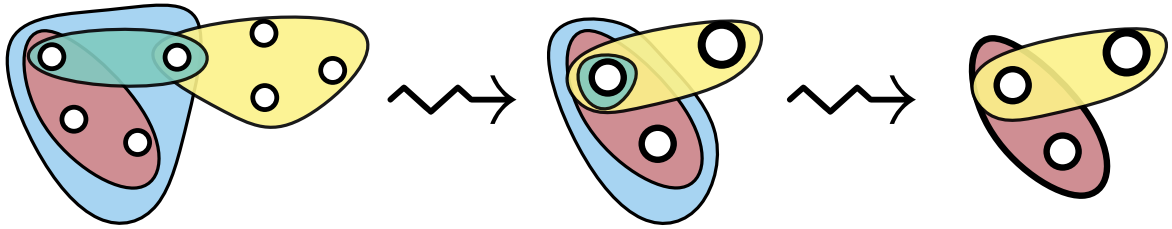
large number ...

Clustering-based Coarsening

Main Design Goals:

1: reduce **size** of nets $\rightsquigarrow$ easier local search

2: reduce **number** of nets $\rightsquigarrow$ easier initial partitioning



$\Longrightarrow$ hypergraph-tailored rating functions:

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

← of heavy nets ...

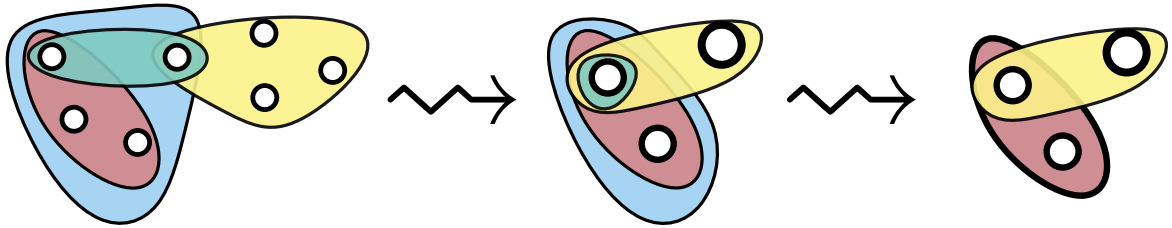
large number ... →

Clustering-based Coarsening

Main Design Goals:

1: reduce **size** of nets $\rightsquigarrow$ easier local search

2: reduce **number** of nets $\rightsquigarrow$ easier initial partitioning



$\Longrightarrow$ hypergraph-tailored rating functions:

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

of heavy nets ...

... with small size

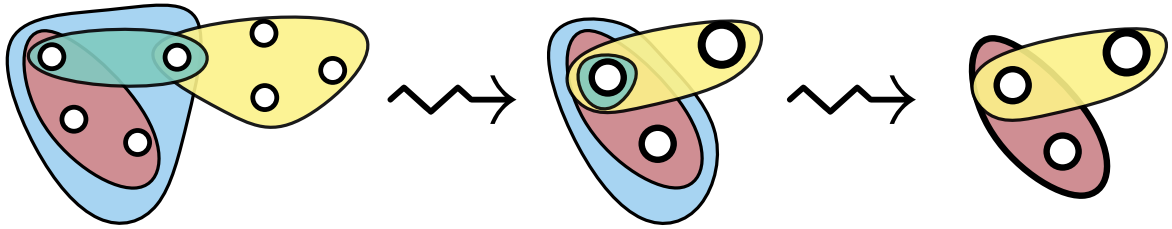
large number ...

Clustering-based Coarsening

Main Design Goals:

1: reduce **size** of nets $\rightsquigarrow$ easier local search ✓

2: reduce **number** of nets $\rightsquigarrow$ easier initial partitioning ✓



$\Longrightarrow$ hypergraph-tailored rating functions:

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

$\omega(e)$: of heavy nets ...

$|e|-1$: ... with small size

large number ...

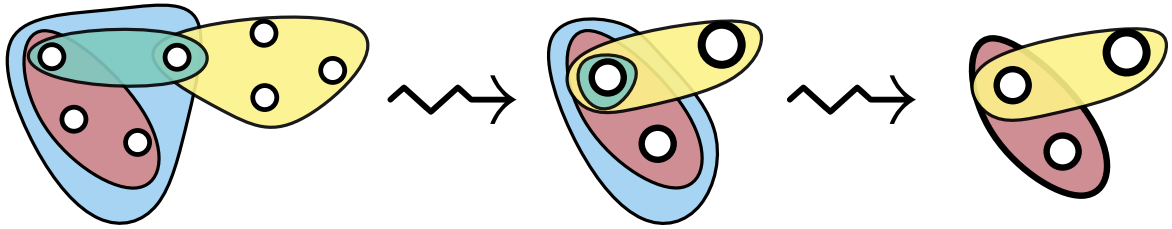
Institute of Theoretical Informatics
Algorithmics Group

Clustering-based Coarsening

Main Design Goals:

1: reduce **size** of nets $\rightsquigarrow$ easier local search ✓

2: reduce **number** of nets $\rightsquigarrow$ easier initial partitioning ✓



$\Longrightarrow$ hypergraph-tailored rating functions:

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

of heavy nets ...

... with small size

large number ...

3: maintain **structural similarity** $\rightsquigarrow$ good coarse solutions

$\Longrightarrow$ prefer clustering over matching

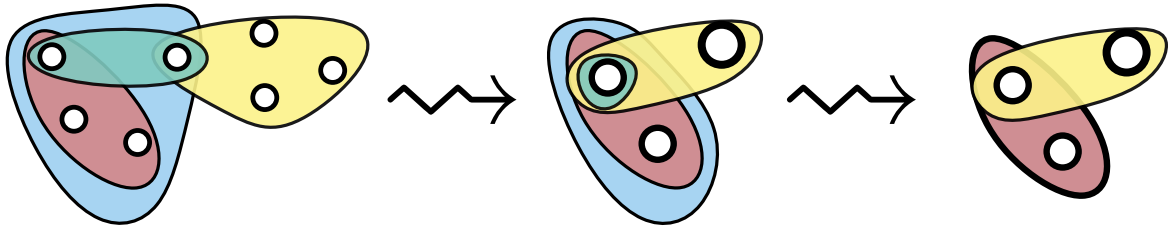
$\Longrightarrow$ ensure $\sim$balanced vertex weights

Clustering-based Coarsening

Main Design Goals:

1: reduce **size** of nets $\rightsquigarrow$ easier local search ✓

2: reduce **number** of nets $\rightsquigarrow$ easier initial partitioning ✓



$\Rightarrow$ hypergraph-tailored rating functions:

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

$\omega(e)$: of heavy nets ...

$|e|-1$: ... with small size

large number ...

3: maintain **structural similarity** $\rightsquigarrow$ good coarse solutions

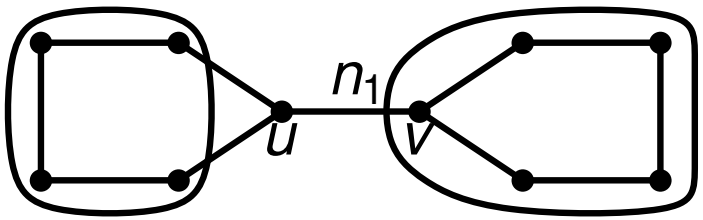
$\Rightarrow$ prefer clustering over matching

$\Rightarrow$ ensure $\sim$balanced vertex weights

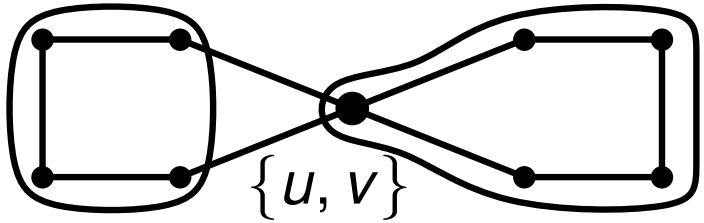
enough?

What could possibly go wrong?

... a lot:



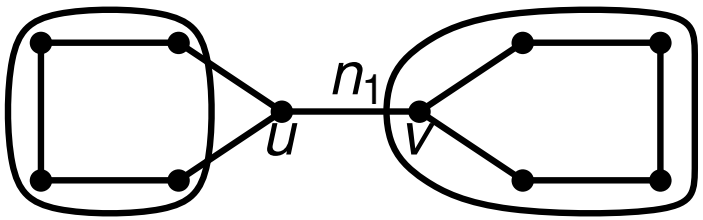
input



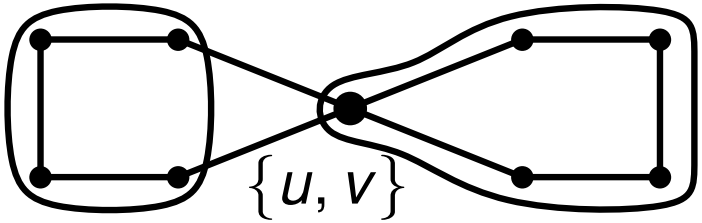
obscured clusters

What could possibly go wrong?

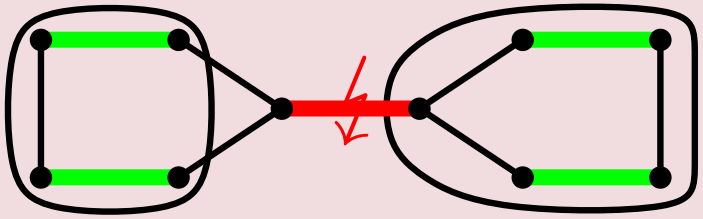
... a lot:



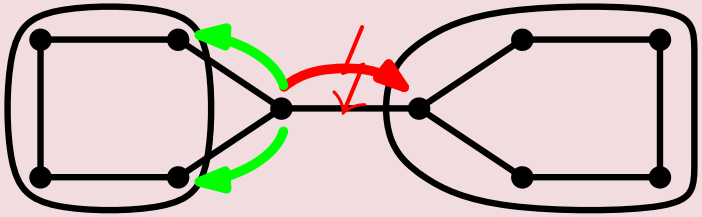
input



obscured clusters

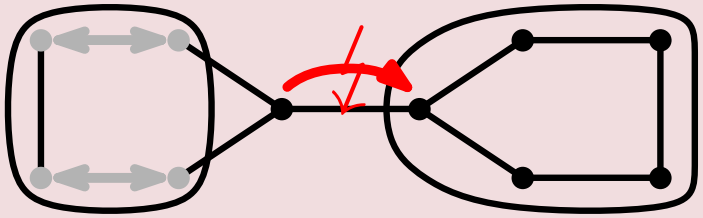


maximal matching

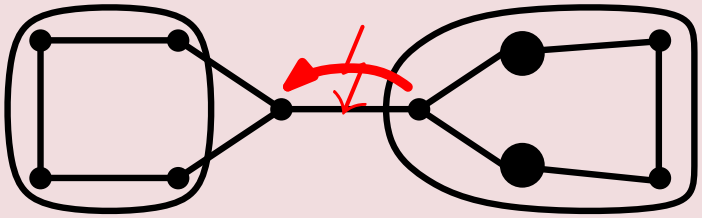


random tie-breaking

ISSUES



prefer unclustered

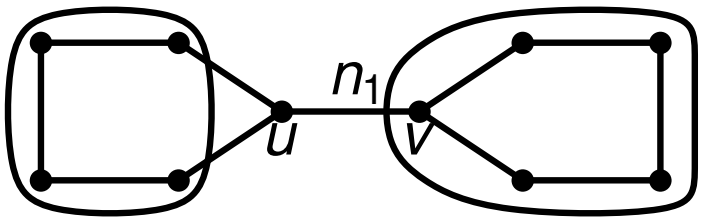


heavy neighbors

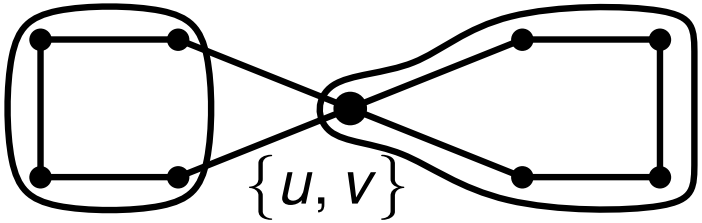
Institute of Theoretical Informatics
Algorithmics Group

What could possibly go wrong?

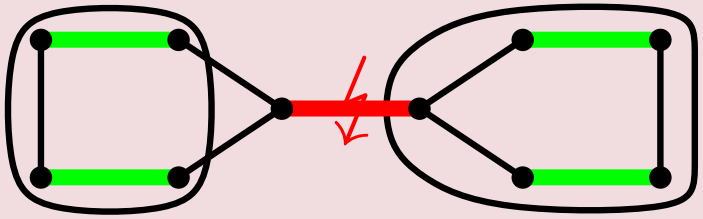
... a lot:



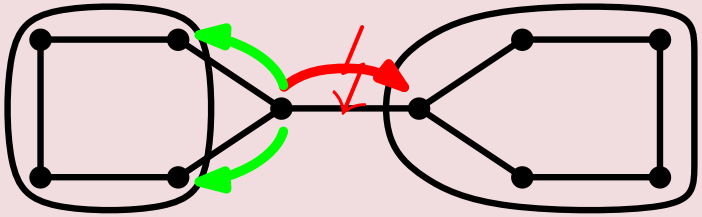
input



obscured clusters

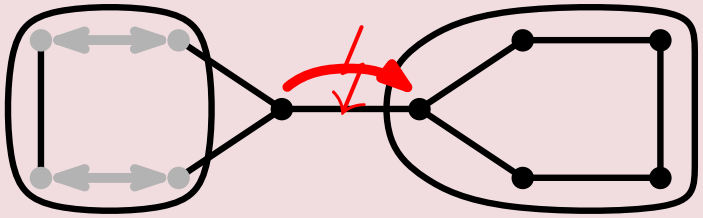


maximal matching

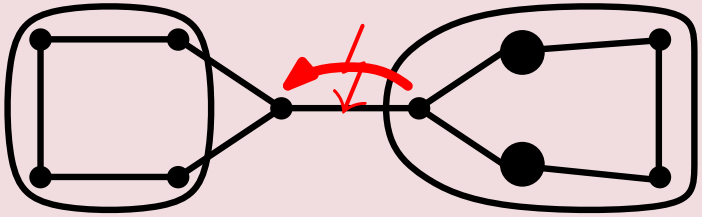


random tie-breaking

ISSUES



prefer unclustered

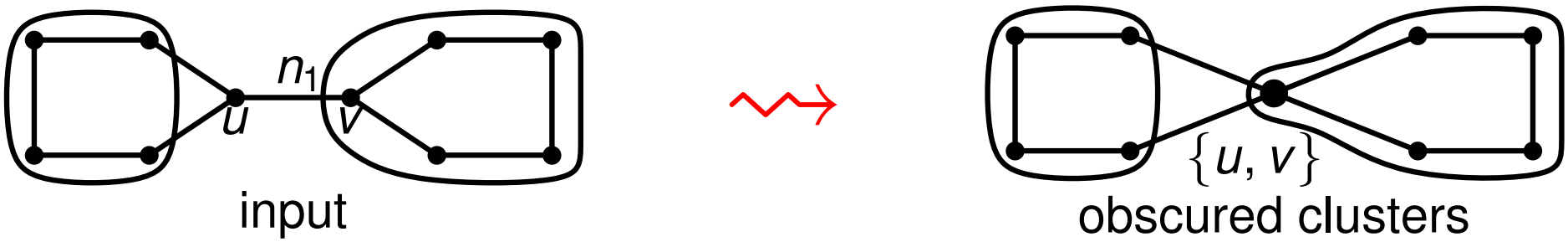


heavy neighbors

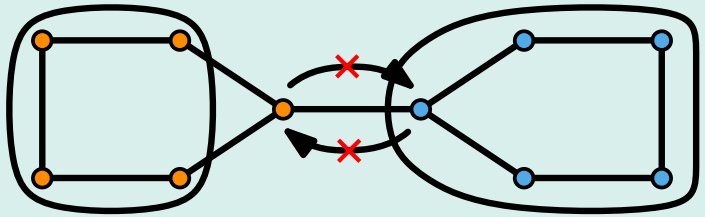
⇒ **Problem:** relying **only** on **local** information!

Community-aware Coarsening

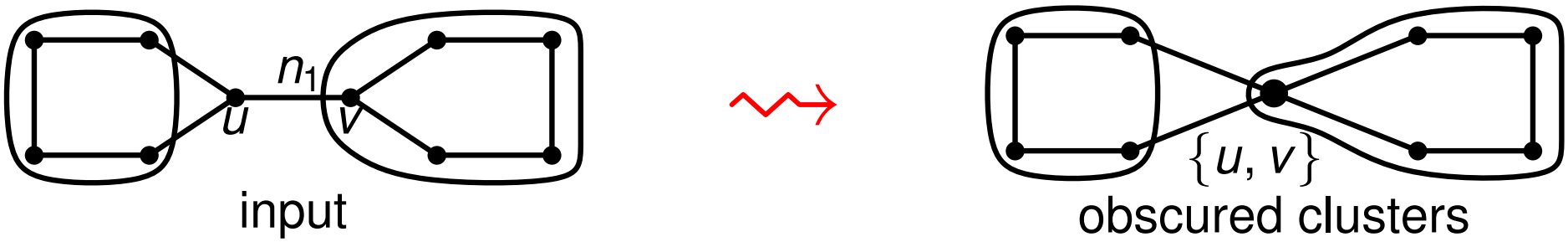
Community-aware Coarsening

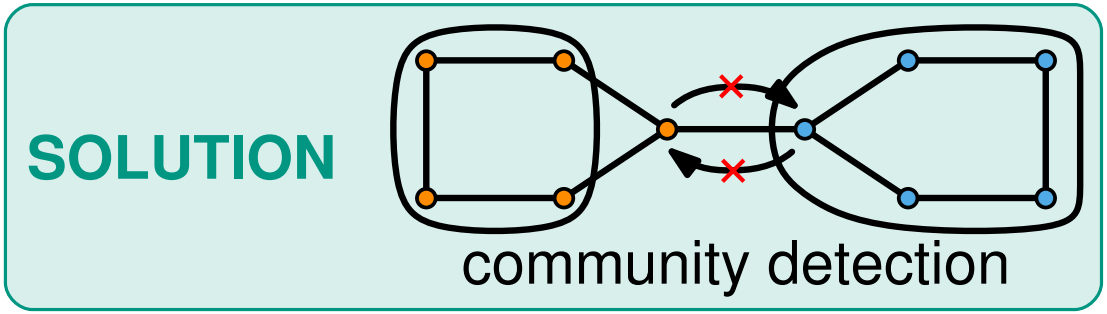


SOLUTION



Community-aware Coarsening





Framework:

- preprocessing: determine **community structure**
- only allow **intra-community** contractions

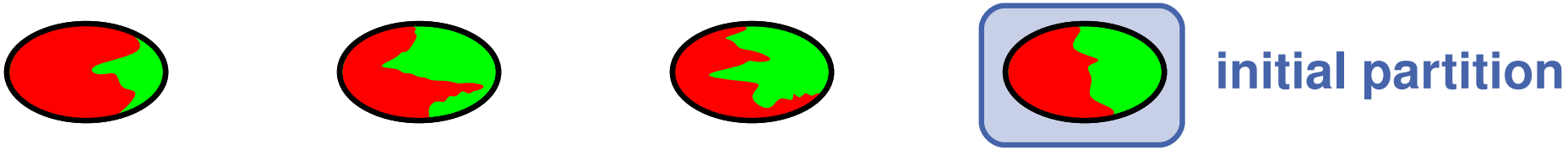
Initial Partitioning

Institute of Theoretical Informatics
Algorithmics Group

Initial Partitioning

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

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



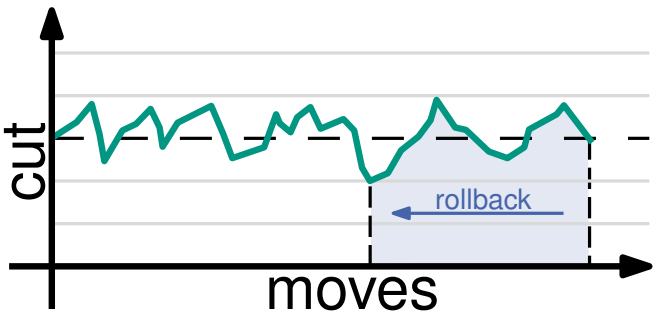
Local Search

Institute of Theoretical Informatics
Algorithmics Group

Fiduccia-Mattheyses Algorithm

Algorithm 1: FM Local Search

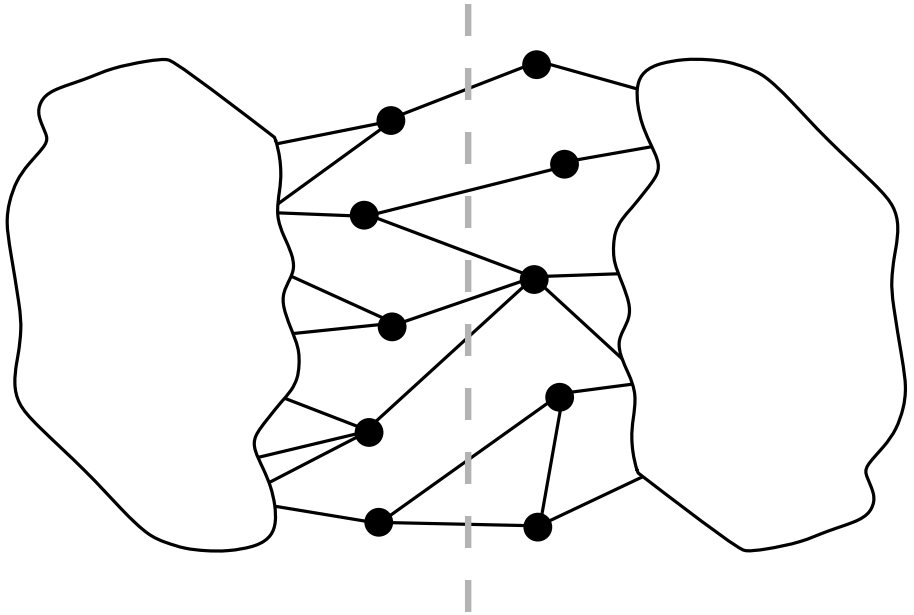
while ¬ *done* **do**
  `find` best move
  `perform` best move
`rollback` to best solution



can worsen solution

Example for Graphs:

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

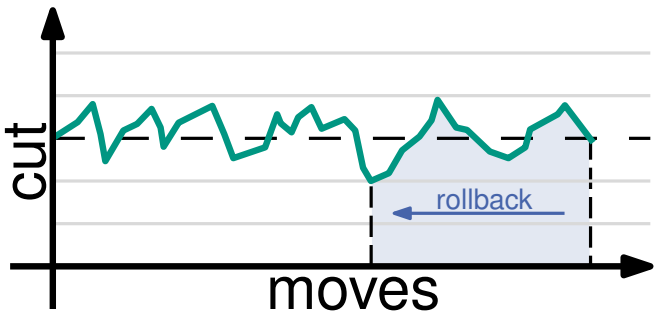


Institute of Theoretical Informatics
Algorithmics Group

Fiduccia-Mattheyses Algorithm

Algorithm 1: FM Local Search

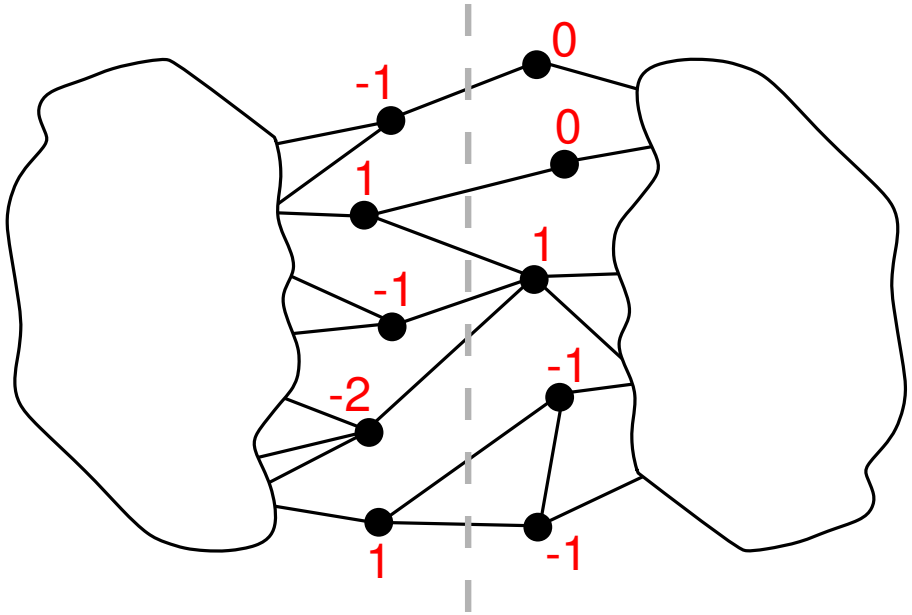
while ¬ *done* **do**
  `find` best move
  `perform` best move
`rollback` to best solution



can worsen solution

Example for Graphs:

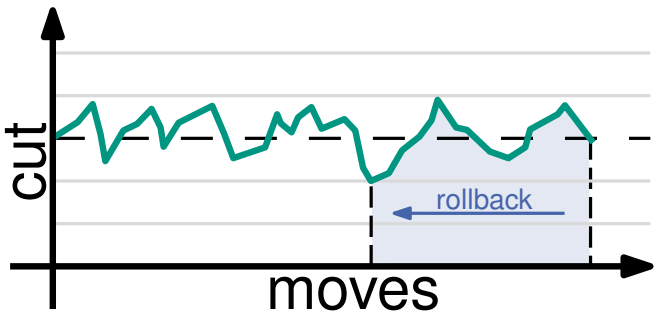
- compute gain $g(v) = d_{\text{ext}}(v) - d_{\text{int}}(v)$
- alternate between blocks
- edge-cut: **7**



Fiduccia-Mattheyses Algorithm

Algorithm 1: FM Local Search

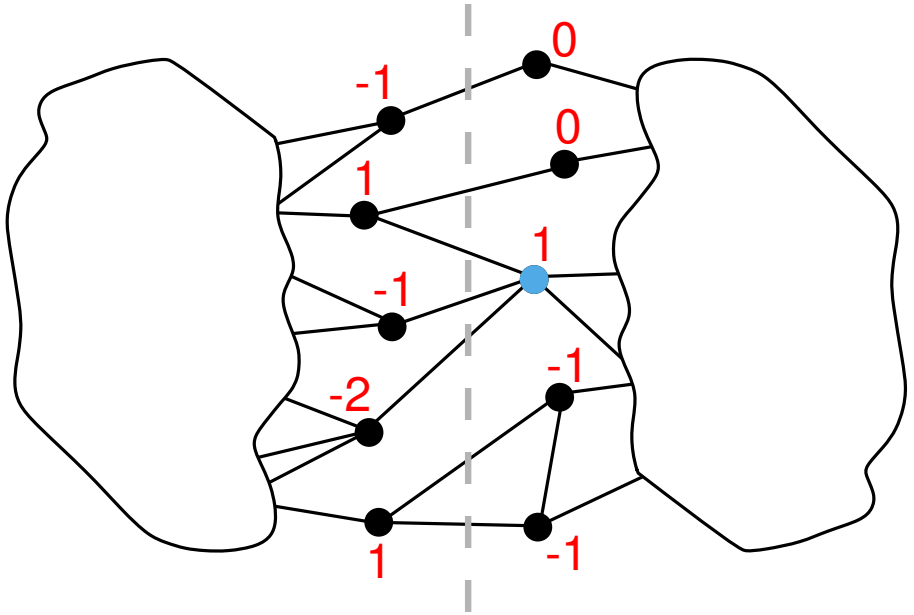
while ¬ *done* **do**
  `find` best move
  `perform` best move
`rollback` to best solution



can worsen solution

Example for Graphs:

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

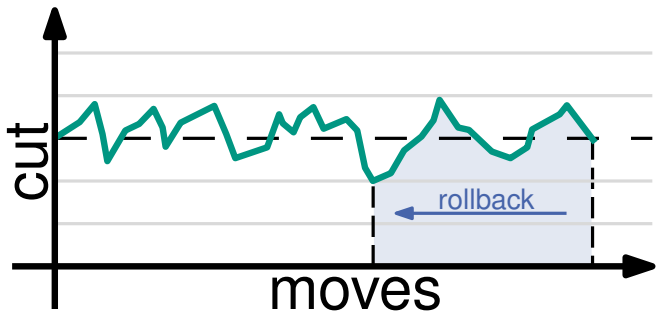


Institute of Theoretical Informatics
Algorithmics Group

Fiduccia-Mattheyses Algorithm

Algorithm 1: FM Local Search

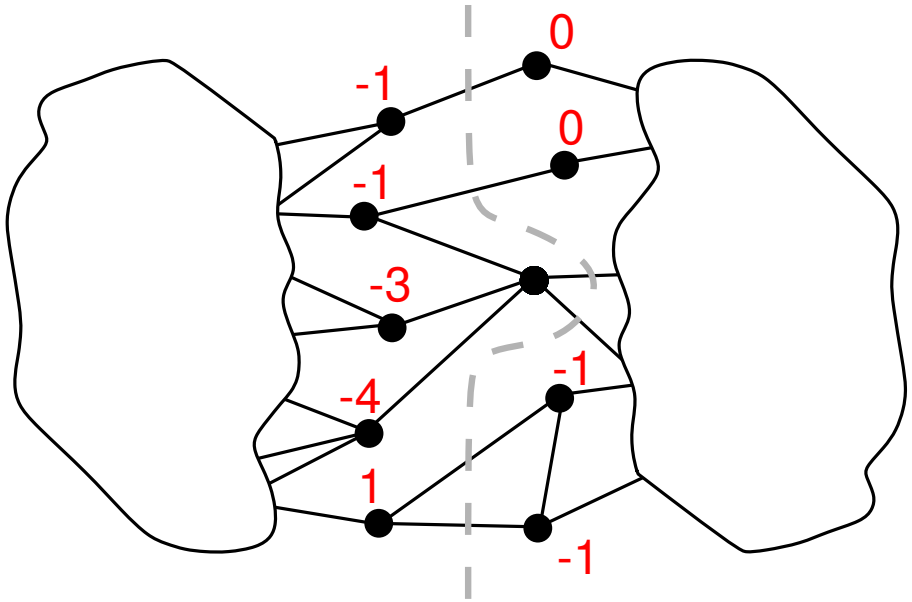
while ¬ *done* **do**
  `find` best move
  `perform` best move
`rollback` to best solution



can worsen solution

Example for Graphs:

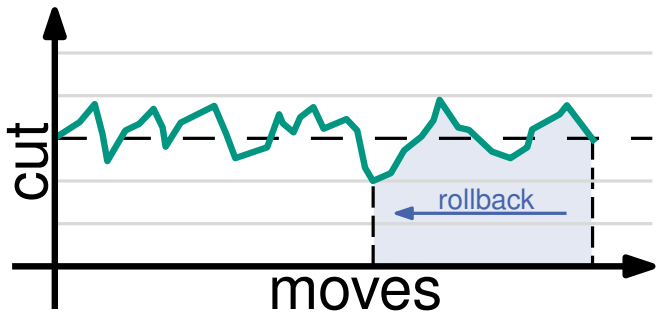
- recalculate gain $g(v)$ of neighbors
- move each node at most once
- edge-cut: **7**, **6**



Fiduccia-Mattheyses Algorithm

Algorithm 1: FM Local Search

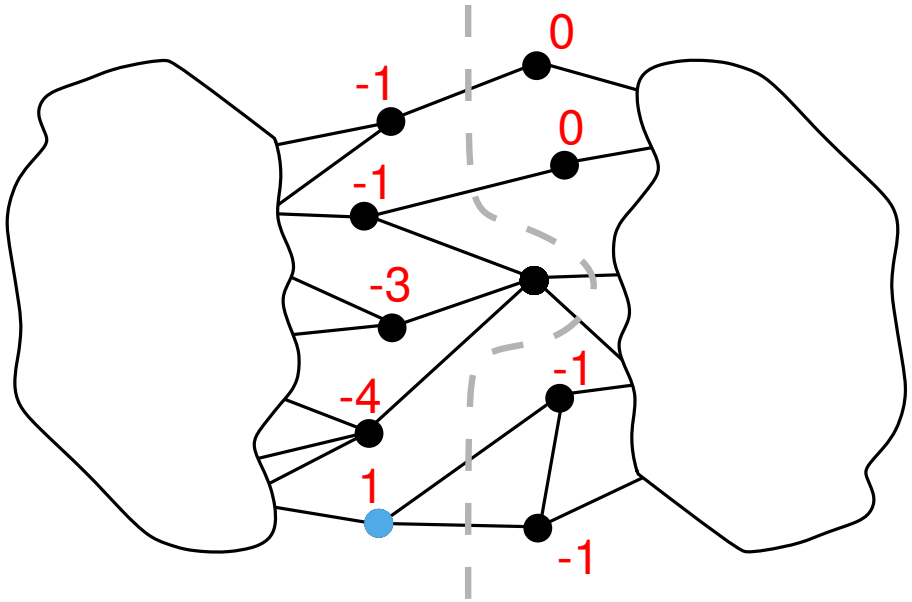
while ¬ *done* **do**
  `find` best move
  `perform` best move
`rollback` to best solution



can worsen solution

Example for Graphs:

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

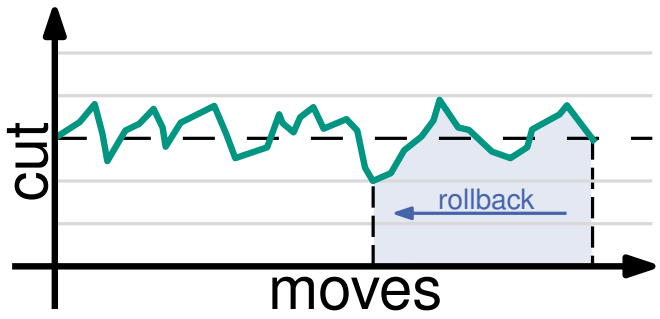


Institute of Theoretical Informatics
Algorithmics Group

Fiduccia-Mattheyses Algorithm

Algorithm 1: FM Local Search

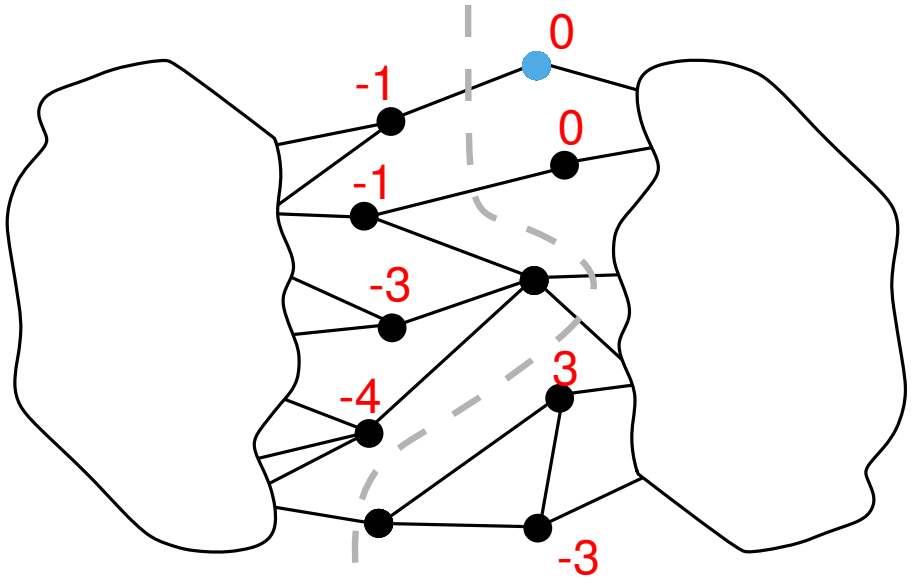
while ¬ *done* **do**
  `find` best move
  `perform` best move
`rollback` to best solution



can worsen solution

Example for Graphs:

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

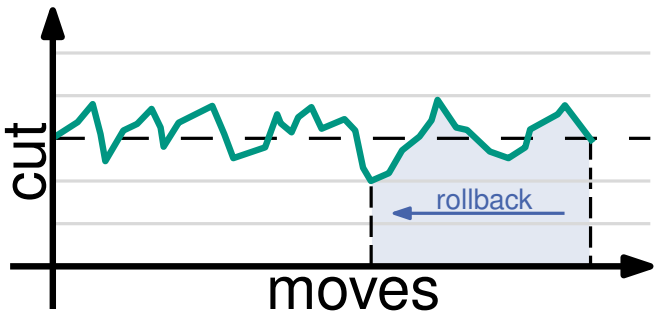


Institute of Theoretical Informatics
Algorithmics Group

Fiduccia-Mattheyses Algorithm

Algorithm 1: FM Local Search

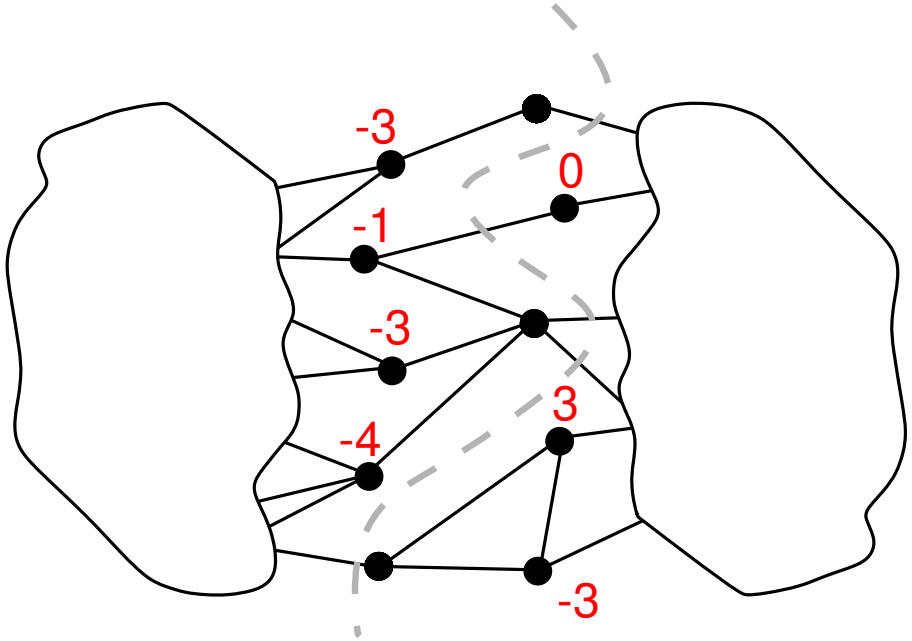
while ¬ *done* **do**
  `find` best move
  `perform` best move
`rollback` to best solution



can worsen solution

Example for Graphs:

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

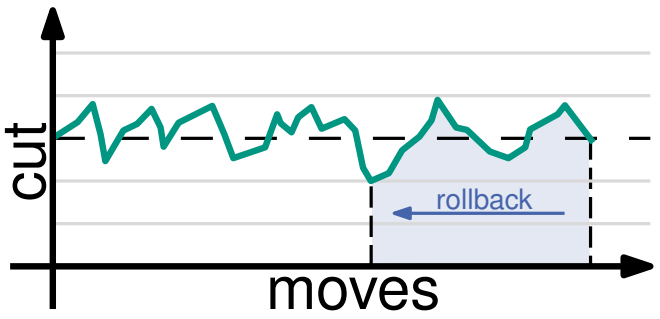


Institute of Theoretical Informatics
Algorithmics Group

Fiduccia-Mattheyses Algorithm

Algorithm 1: FM Local Search

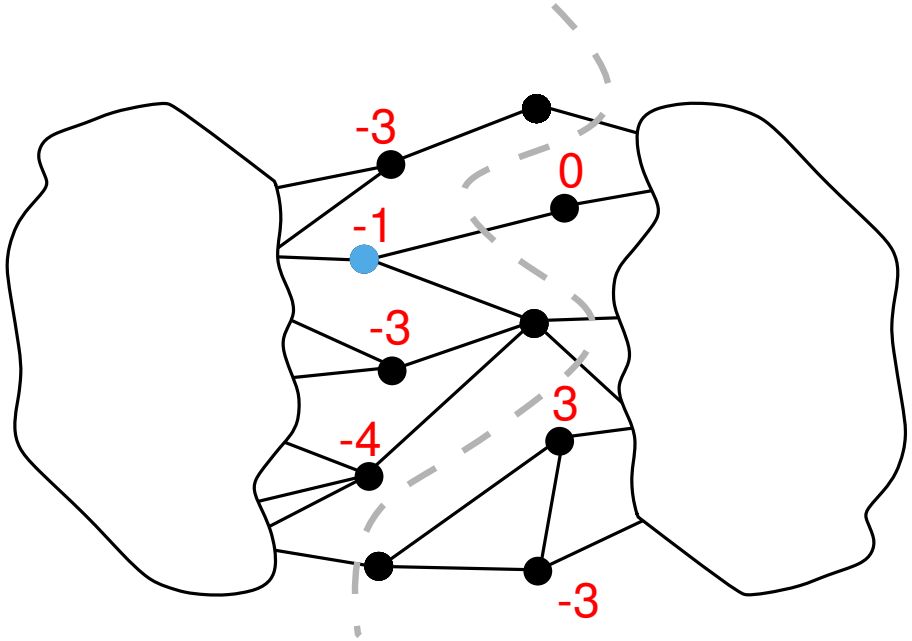
while ¬ *done* **do**
  `find` best move
  `perform` best move
`rollback` to best solution



can worsen solution

Example for Graphs:

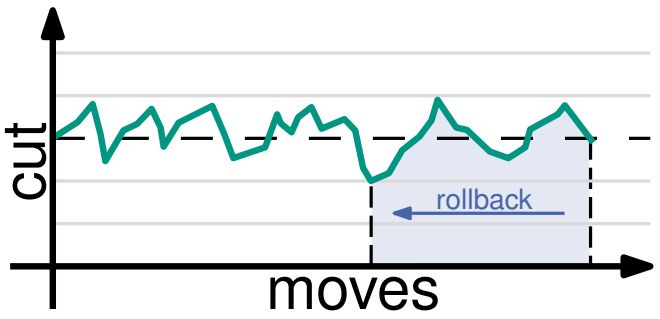
- recalculate gain $g(v)$ of neighbors
- move each node at most once
- edge-cut: **7**, **6**,**5**,**5**



Fiduccia-Mattheyses Algorithm

Algorithm 1: FM Local Search

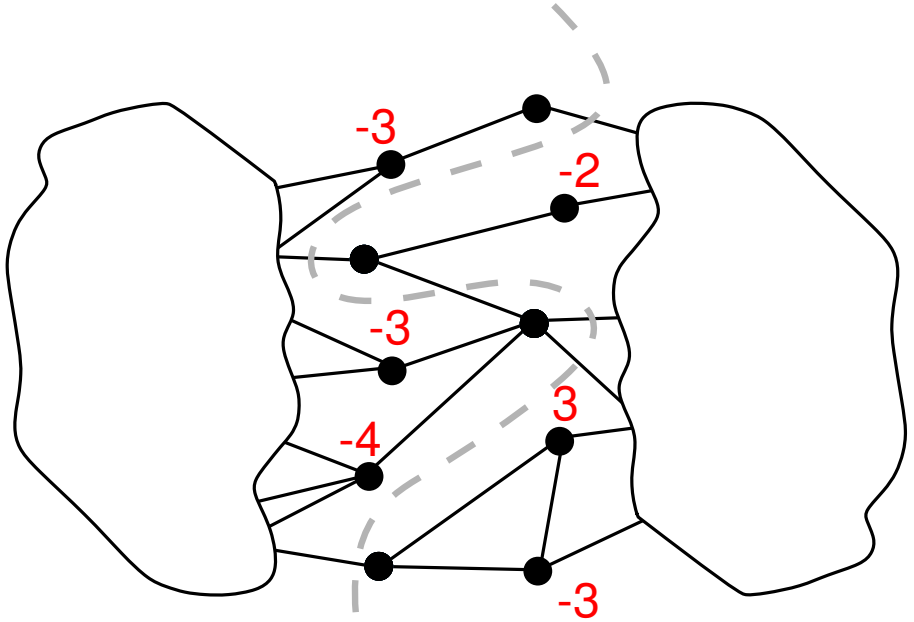
while ¬ *done* **do**
  `find` best move
  `perform` best move
`rollback` to best solution



can worsen solution

Example for Graphs:

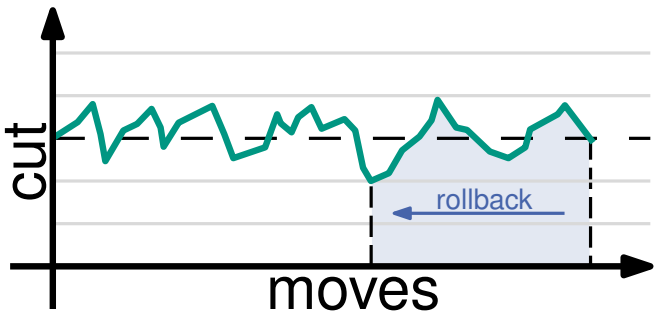
- recalculate gain $g(v)$ of neighbors
- move each node at most once
- edge-cut: **7**, **6**,**5**,**5**,**6**



Fiduccia-Mattheyses Algorithm

Algorithm 1: FM Local Search

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

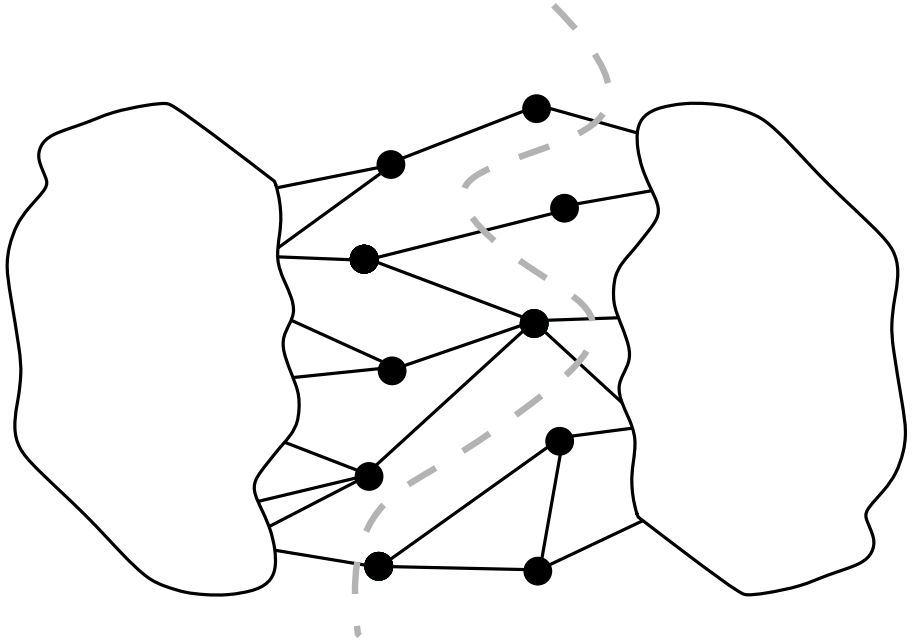


can worsen solution

Example for Graphs:

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

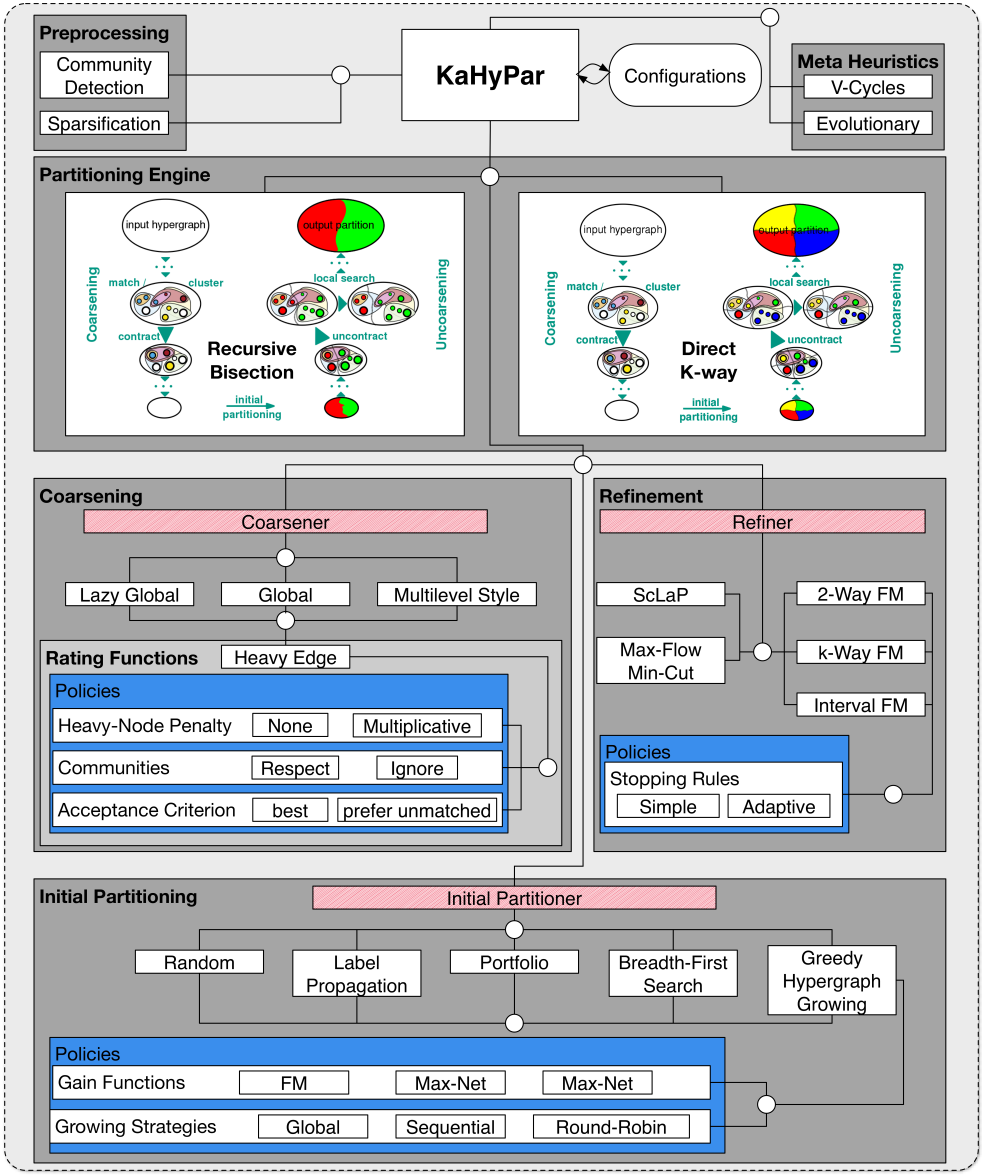
rollback



Institute of Theoretical Informatics
Algorithmics Group

KaHyPar - Karlsruhe Hypergraph Partitioning

- *n*-Level Partitioning Framework
- Objectives:
  - hyperedge cut
  - connectivity ($\lambda - 1$)
- Partitioning Modes:
  - recursive bisection
  - direct *k*-way
- Additional Features:
  - evolutionary algorithm
  - flow-based refinement
  - fixed vertices
  - variable block weights
- `http://www.kahypar.org`



Institute of Theoretical Informatics
Algorithmics Group

References

[ALENEX'16]: S. Schlag, V. Henne, T. Heuer, H. Meyerhenke, P. Sanders, Christian Schulz. *k*-way Hypergraph Partitioning via *n*-Level Recursive Bisection. In 18th Workshop on Algorithm Engineering and Experiments, (ALENEX), pages 53–67, 2016.

[ALENEX'17]: Y. Akhremtsev, T. Heuer, P. Sanders, and S. Schlag. Engineering a direct k-way hypergraph partitioning algorithm. In 19th Workshop on Algorithm Engineering and Experiments, (ALENEX), pages 28–42, 2017.

[SEA'17]: T. Heuer and S. Schlag. Improving Coarsening Schemes for Hypergraph Partitioning by Exploiting Community Structure. In 16th International Symposium on Experimental Algorithms, (SEA), page 21:121:19, 2017.

[SEA'18]: T. Heuer, P. Sanders, S. Schlag. Network Flow-Based Refinement for Multilevel Hypergraph Partitioning. In 17th International Symposium on Experimental Algorithms (SEA), 2018, preprint arXiv:1802.03587.

[GECCO'18]: R. Andre, S. Schlag, and C. Schulz. Memetic Multilevel Hypergraph Partitioning. In Genetic and Evolutionary Computation Conference (GECCO), 2018, preprint arXiv:1710.01968.

[Karypis, Kumar 99]: G. Karypis and V. Kumar. Multilevel K-way Hypergraph Partitioning. In Proceedings of the 36th ACM/IEEE Design Automation Conference, pages 343–348. ACM, 1999.