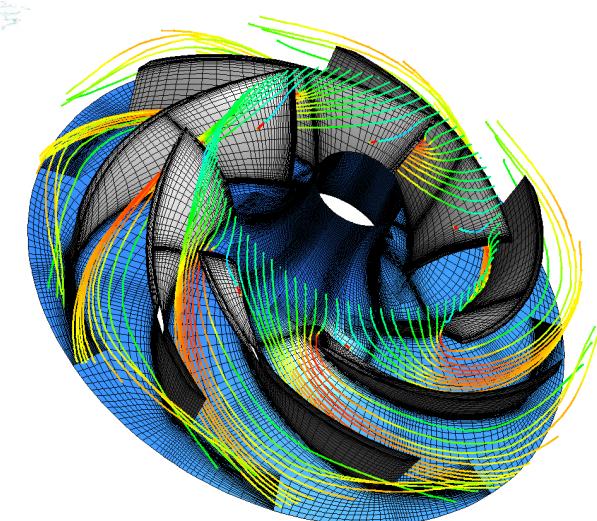
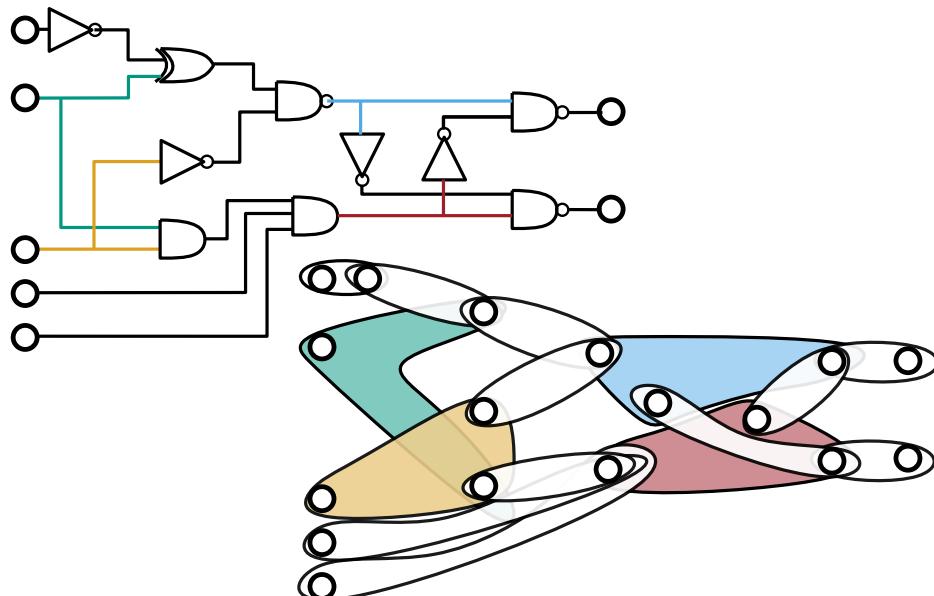


Brief Introduction to Hypergraph Partitioning

Bioinformatics Programming Practical Kickoff Meeting · April 19, 2018
Sebastian Schlag

INSTITUTE OF THEORETICAL INFORMATICS · ALGORITHMIC GROUP

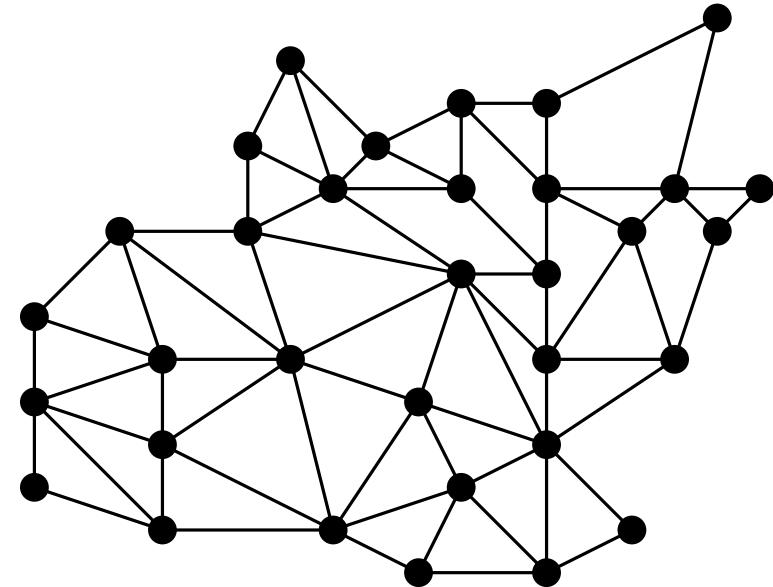


Graphs and Hypergraphs

Graph $G = (V, E)$

vertices edges

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



Graphs and Hypergraphs

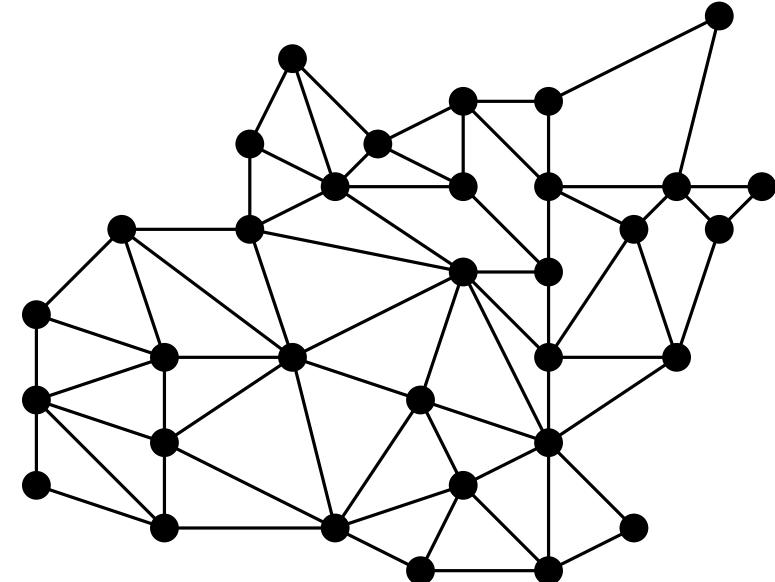
Graph $G = (V, E)$

vertices edges

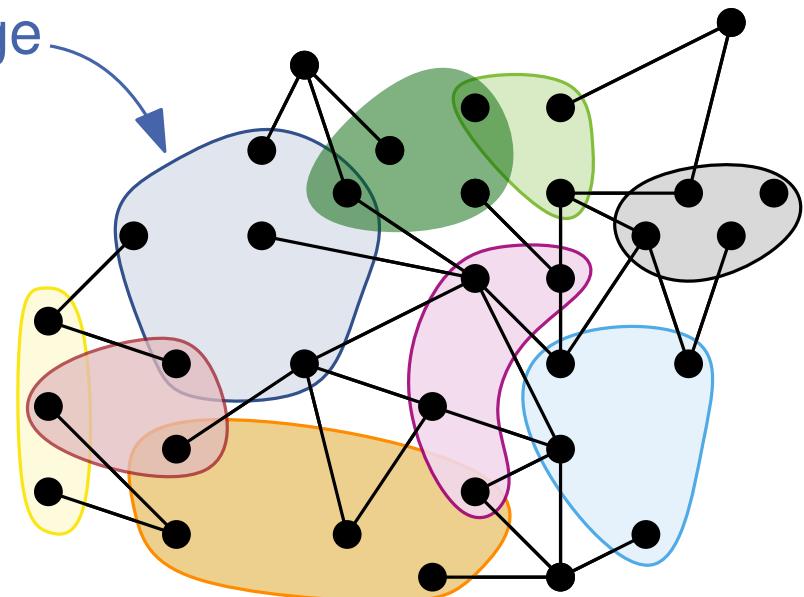
- models **relationships** between **objects**
- dyadic (**2-ary**) relationships

Hypergraph $H = (V, E)$

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



hyperedge



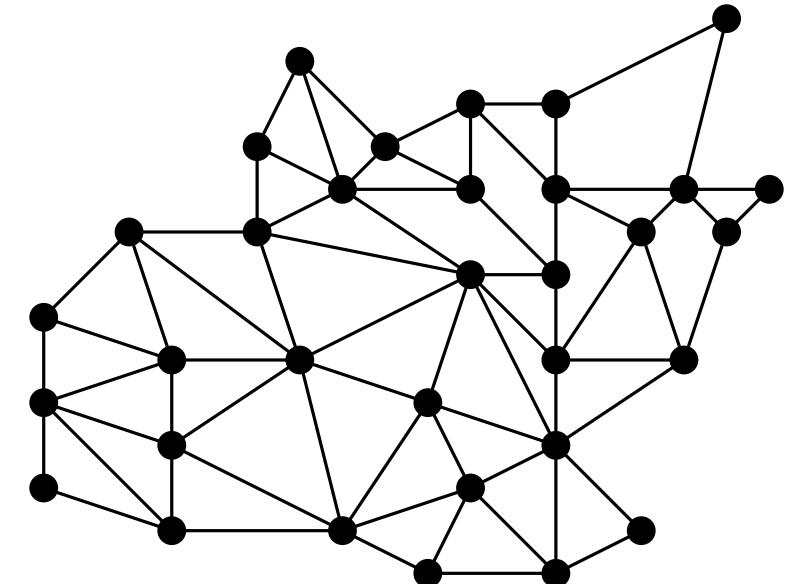
ε -Balanced Graph and Hypergraph Partitioning

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

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

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

- **objective function on edges is minimized**



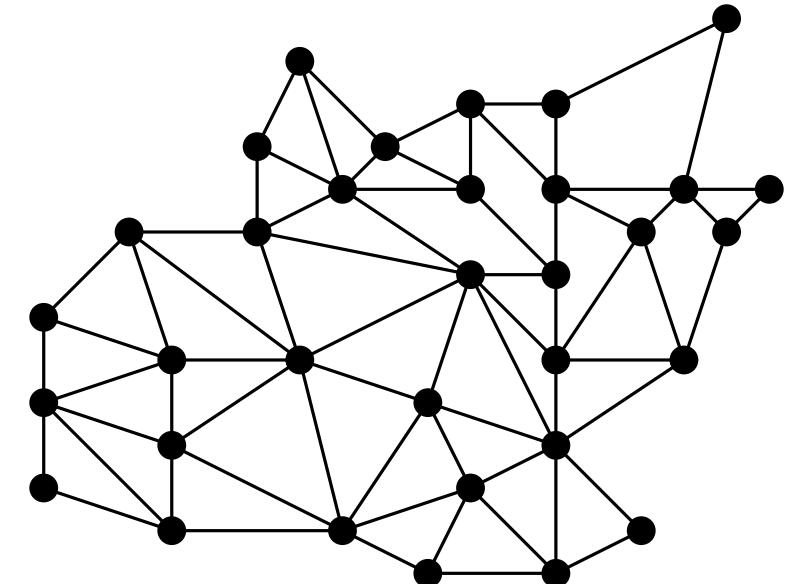
ε -Balanced Graph and Hypergraph Partitioning

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

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

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

- **objective function on edges is minimized**



ε -Balanced Graph and Hypergraph Partitioning

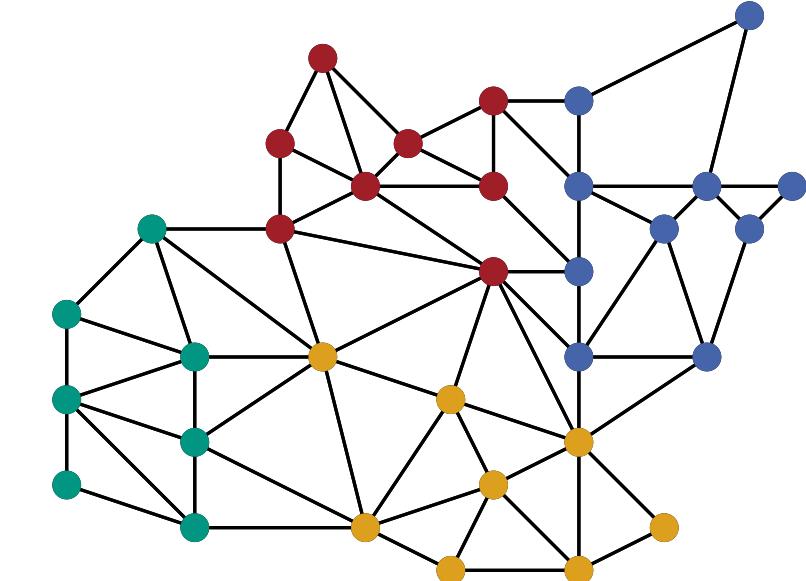
Partition (hyper)graph $G = (V, E, c : V \rightarrow \mathbb{R}_{>0}, \omega : E \rightarrow \mathbb{R}_{>0})$
into k disjoint blocks V_1, \dots, V_k s.t.

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

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

imbalance parameter

- **objective function on edges is minimized**



ε -Balanced Graph and Hypergraph Partitioning

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

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

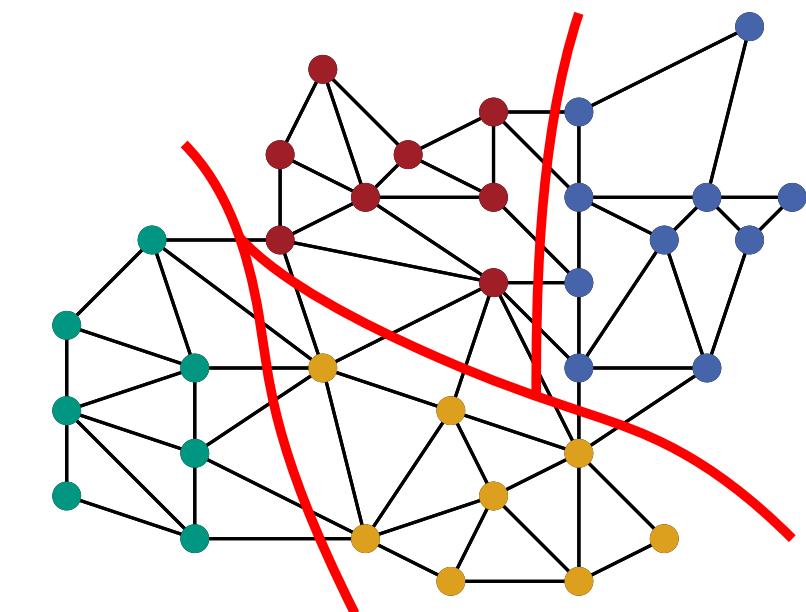
$$c(V_i) \leq (1 + \varepsilon) \lceil \frac{c(V)}{k} \rceil$$

imbalance parameter

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

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

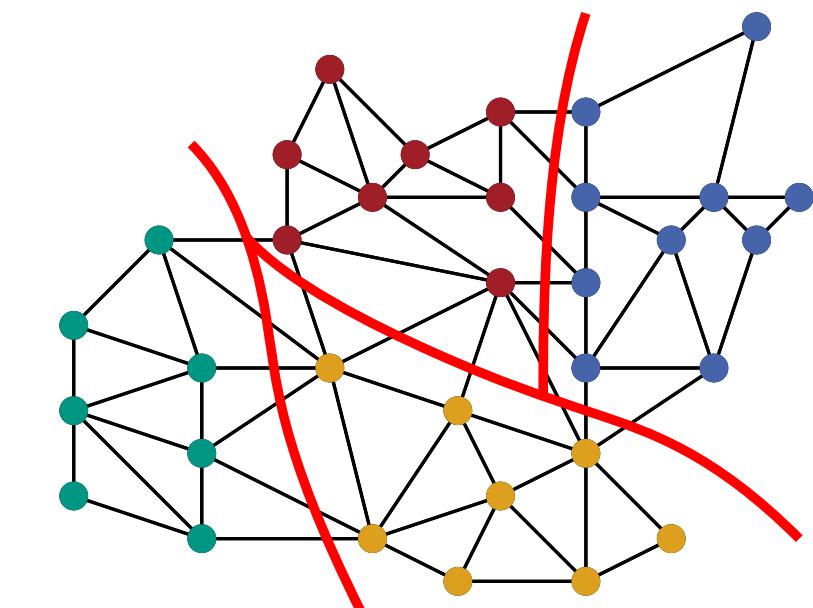
$$c(V_i) \leq (1 + \varepsilon) \lceil \frac{c(V)}{k} \rceil$$

imbalance parameter

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

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

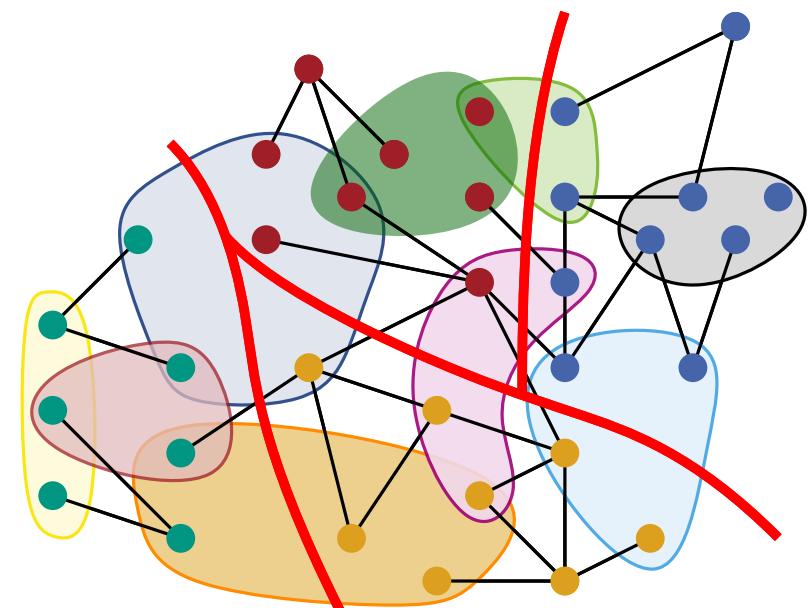
$$c(V_i) \leq (1 + \varepsilon) \lceil \frac{c(V)}{k} \rceil$$

imbalance parameter

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

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

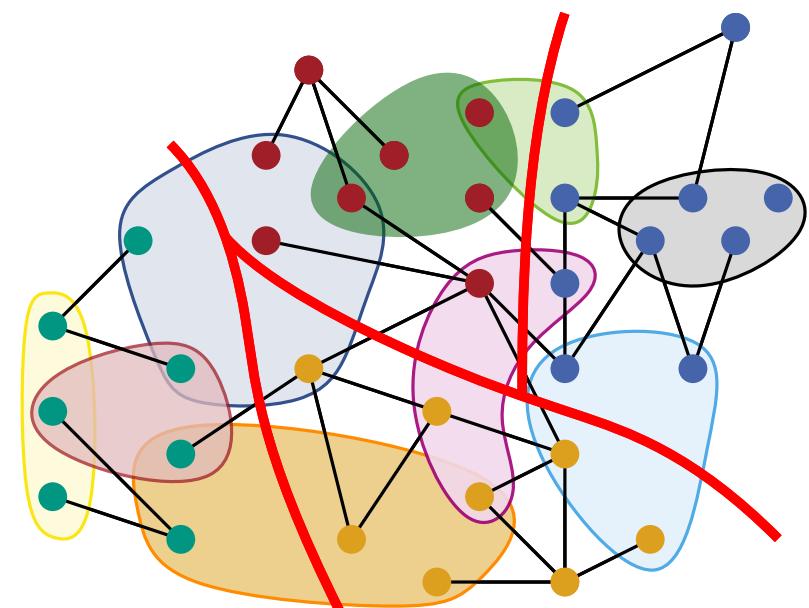
$$c(V_i) \leq (1 + \varepsilon) \lceil \frac{c(V)}{k} \rceil$$

imbalance parameter

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

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

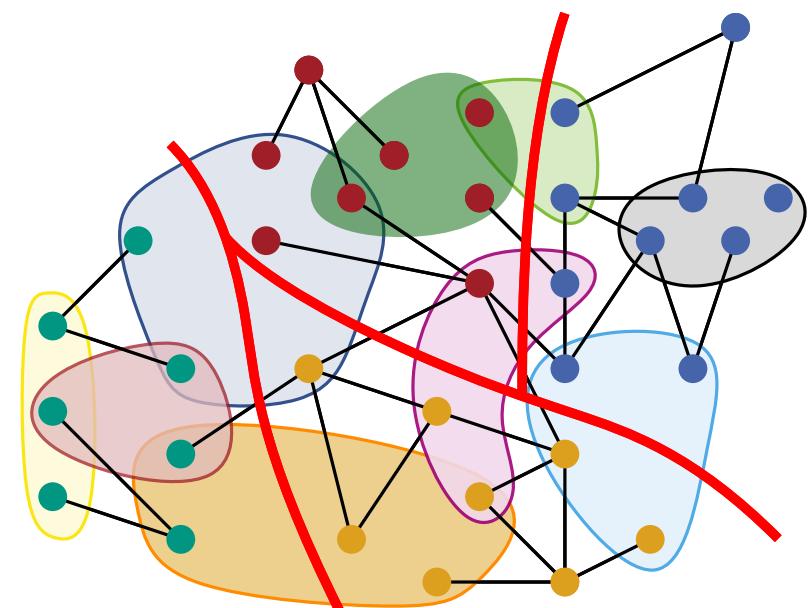
$$c(V_i) \leq (1 + \varepsilon) \lceil \frac{c(V)}{k} \rceil$$

imbalance parameter

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

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

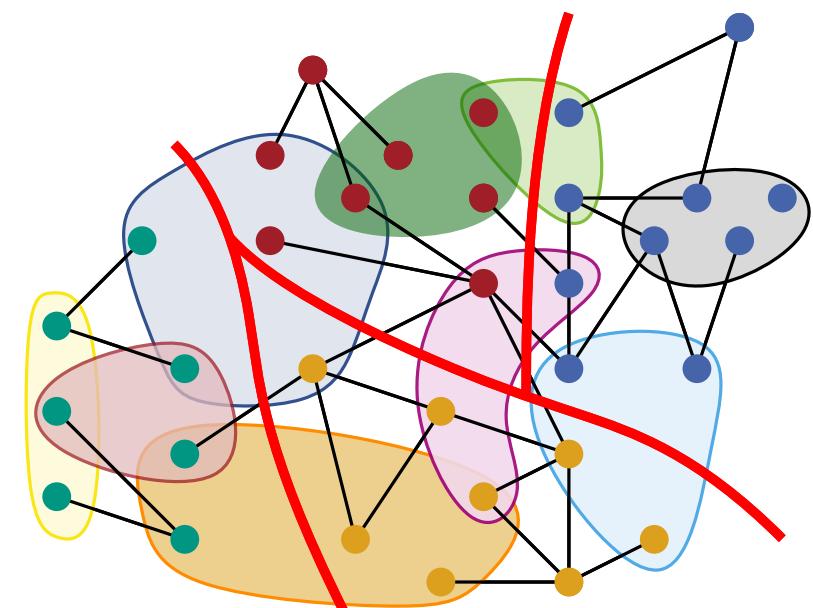
$$c(V_i) \leq (1 + \varepsilon) \lceil \frac{c(V)}{k} \rceil$$

imbalance parameter

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

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

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

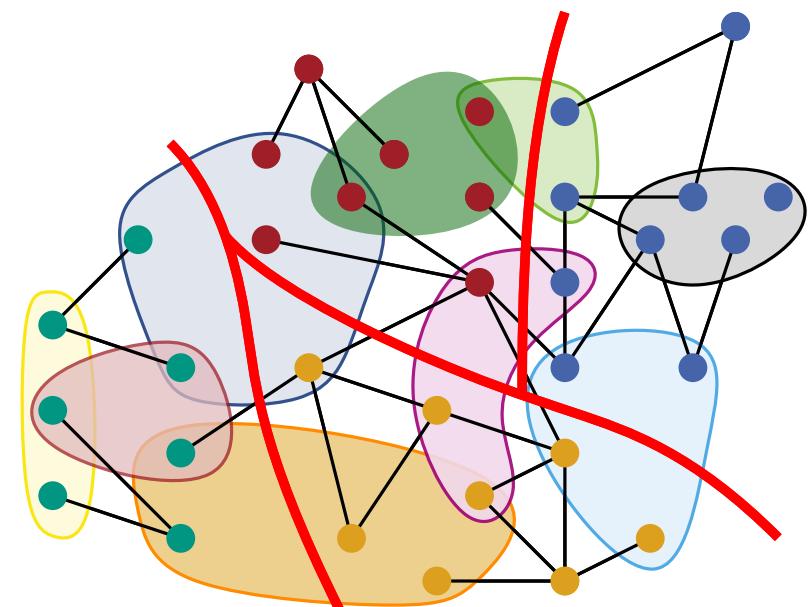
imbalance parameter

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

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

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

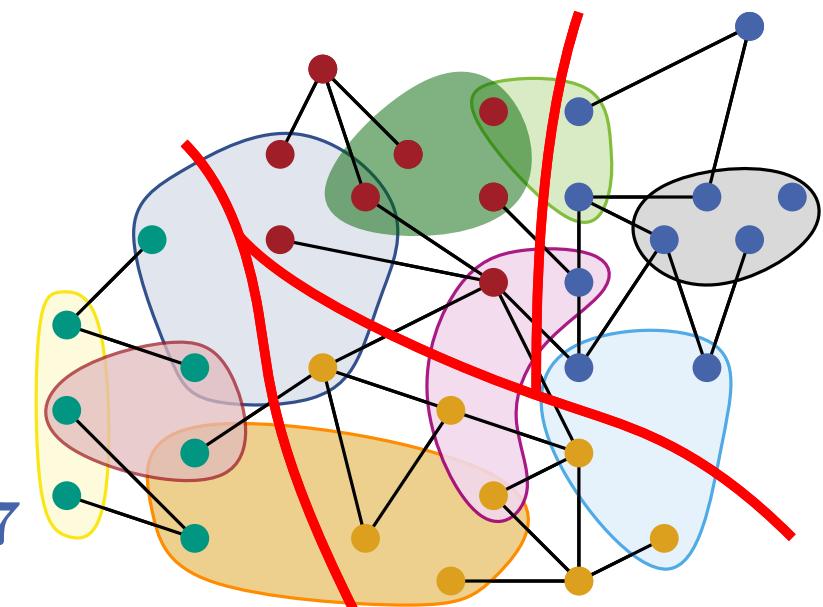
imbalance parameter

- **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

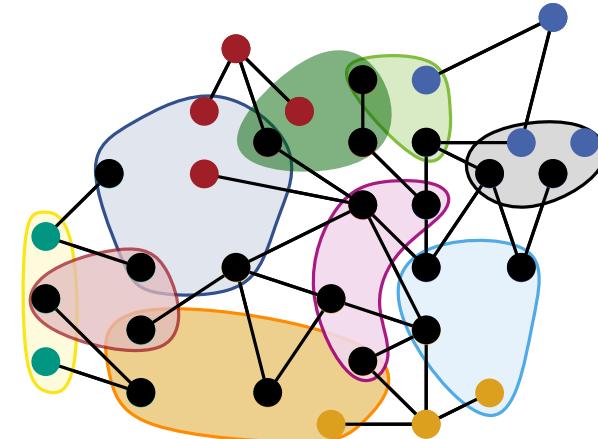


Variants of Standard Hypergraph Partitioning

Possibly relevant/interesting variants:

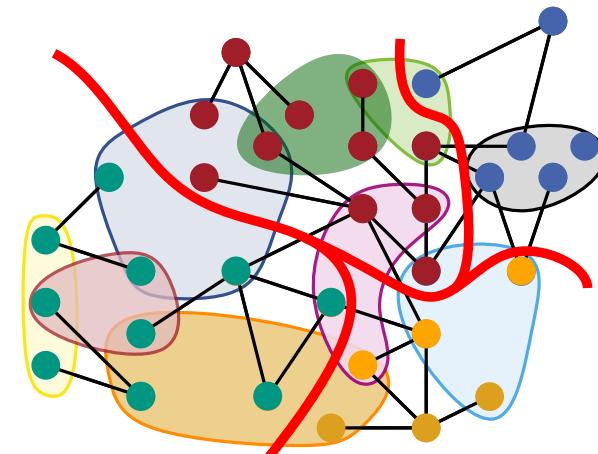
Partitioning with Fixed Vertices:

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



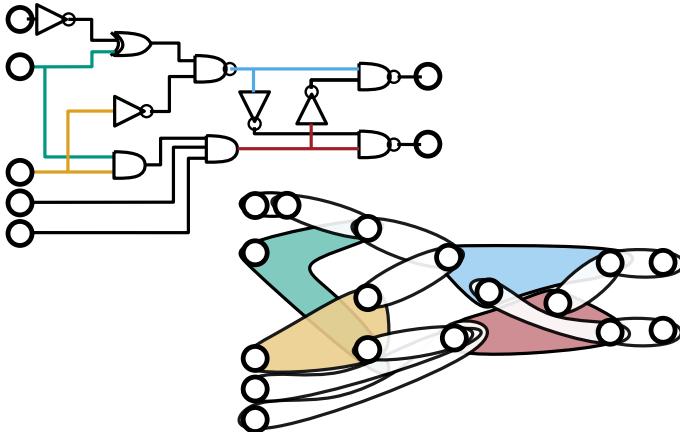
Partitioning with Variable Block Weights:

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



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

Applications

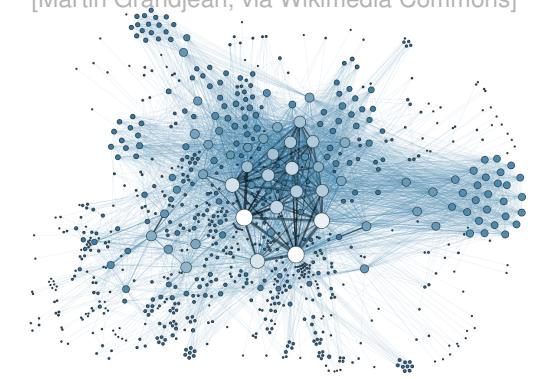


VLSI Design



Warehouse Optimization

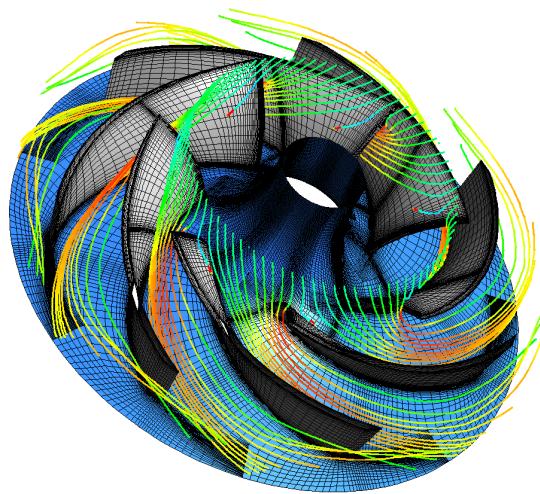
[Martin Grandjean, via Wikimedia Commons]



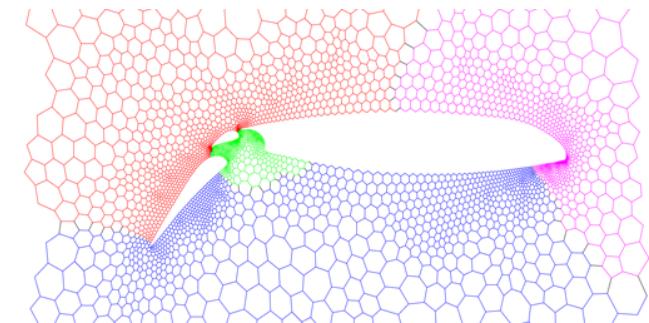
Complex Networks



Route Planning



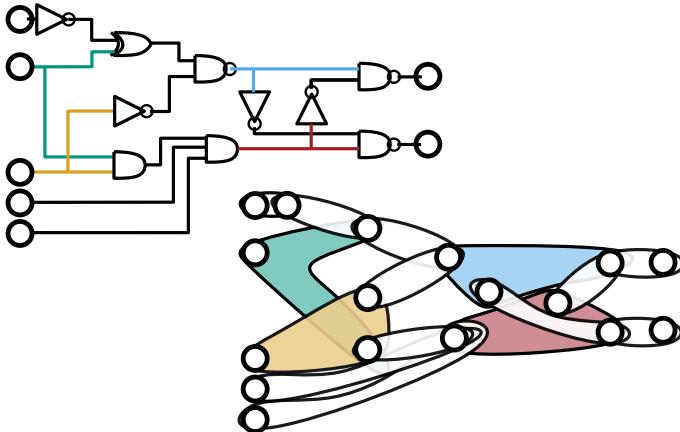
Simulation



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

Scientific Computing

Applications

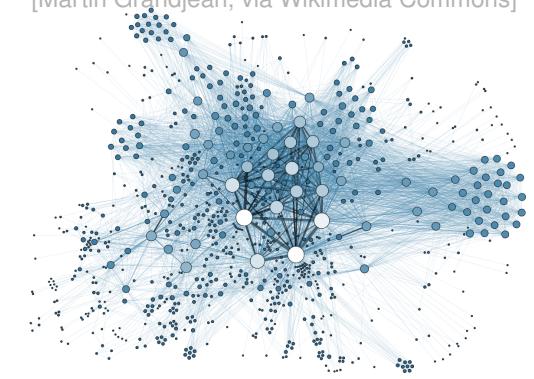


VLSI Design



Warehouse Optimization

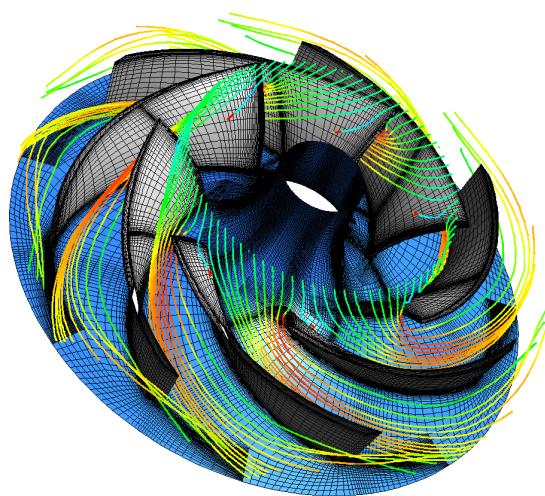
[Martin Grandjean, via Wikimedia Commons]



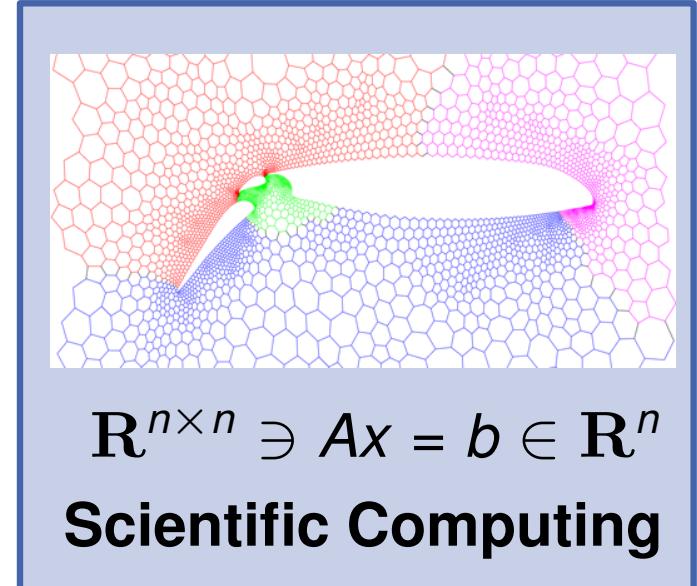
Complex Networks



Route Planning



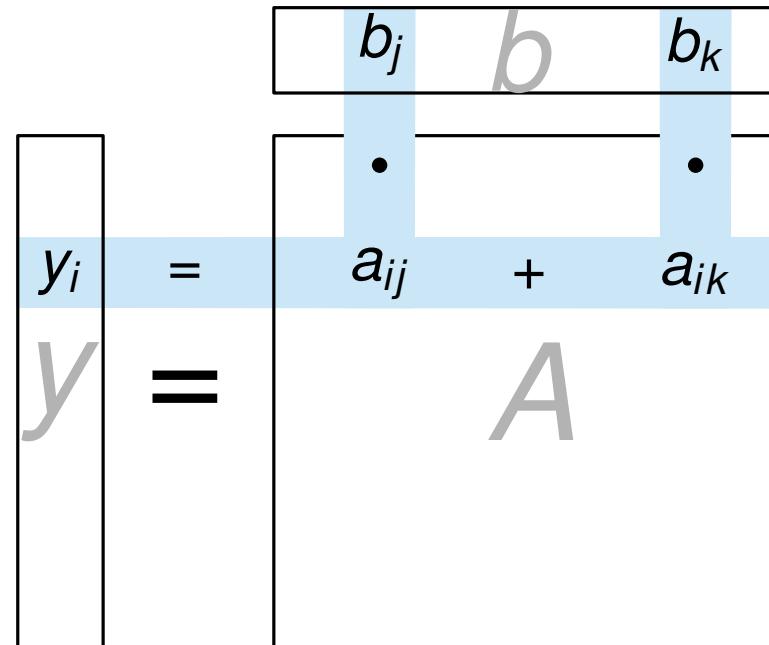
Simulation



Scientific Computing

Parallel Sparse-Matrix Vector Product (SpM×V)

$$y = A b$$

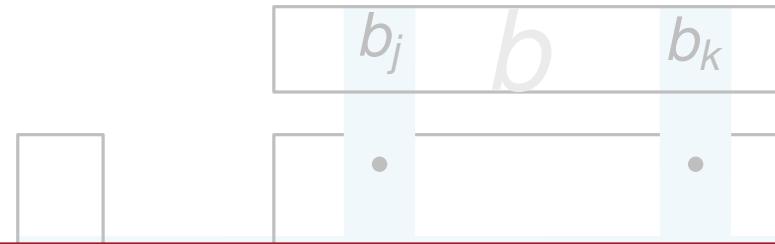


Setting:

- 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 \times V)

$$y = A b$$



Task: distribute A to nodes of supercomputer such that

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

Setting:

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

Naive Approach: Rowwise Decomposition

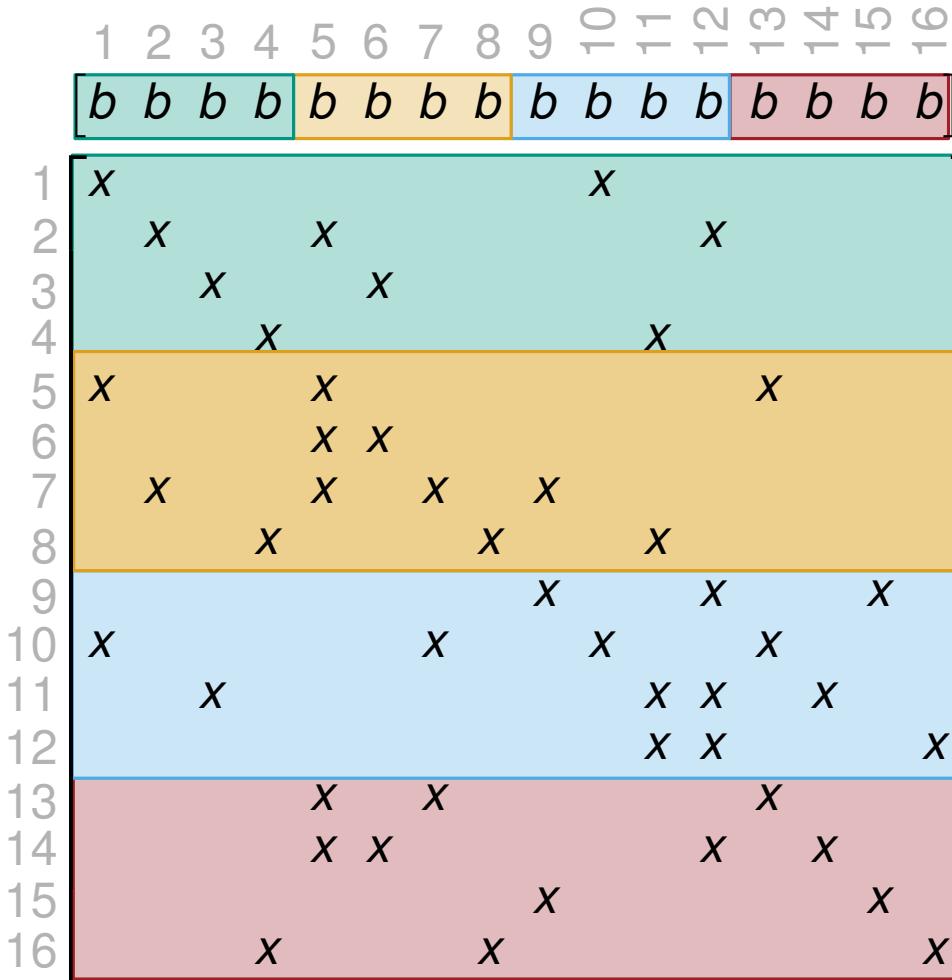
$A \in \mathbb{R}^{16 \times 16}$

	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			x						
9						x		x	x		x		x			x
10	x					x		x	x	x	x	x				
11		x									x	x	x			
12											x	x		x		
13			x	x							x	x				
14			x	x							x	x				
15					x	x						x		x		
16						x		x					x		x	x

Naive Approach: Rowwise Decomposition

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

P_1
 P_2
 P_3
 P_4



Naive Approach: Rowwise Decomposition

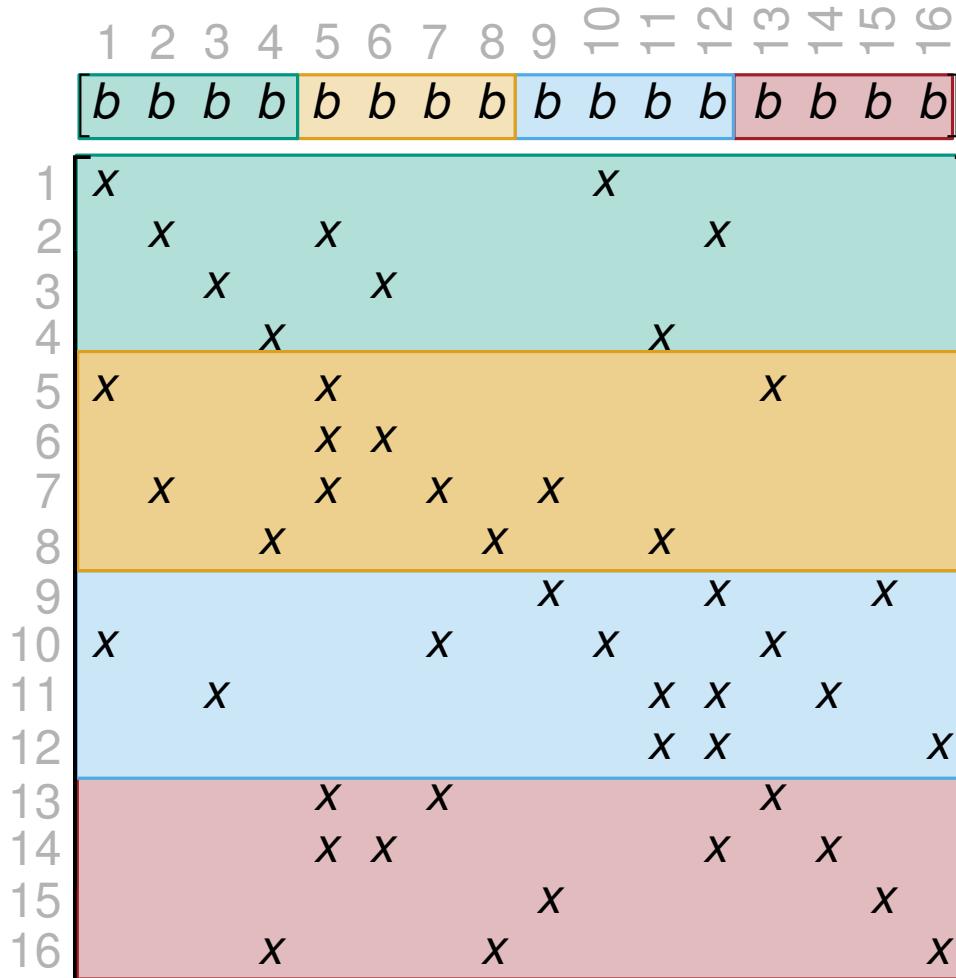
$A \in \mathbb{R}^{16 \times 16}$

P_1

P_2

P_3

P_4



Load Balancing?

$\Rightarrow 9$

$\Rightarrow 12$

$\Rightarrow 14$

$\Rightarrow 12$

Naive Approach: Rowwise Decomposition

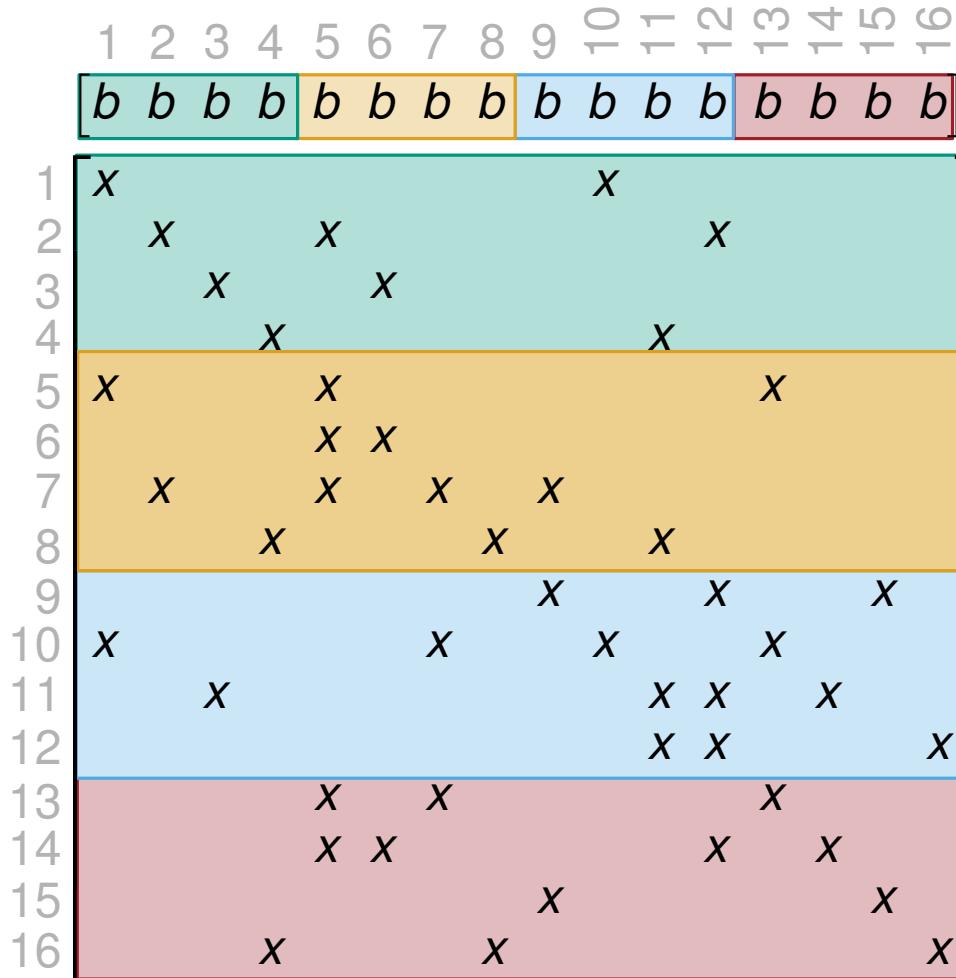
$A \in \mathbb{R}^{16 \times 16}$

P_1

P_2

P_3

P_4



Load Balancing?

$\Rightarrow 9$

$\Rightarrow 12$

$\Rightarrow 14$

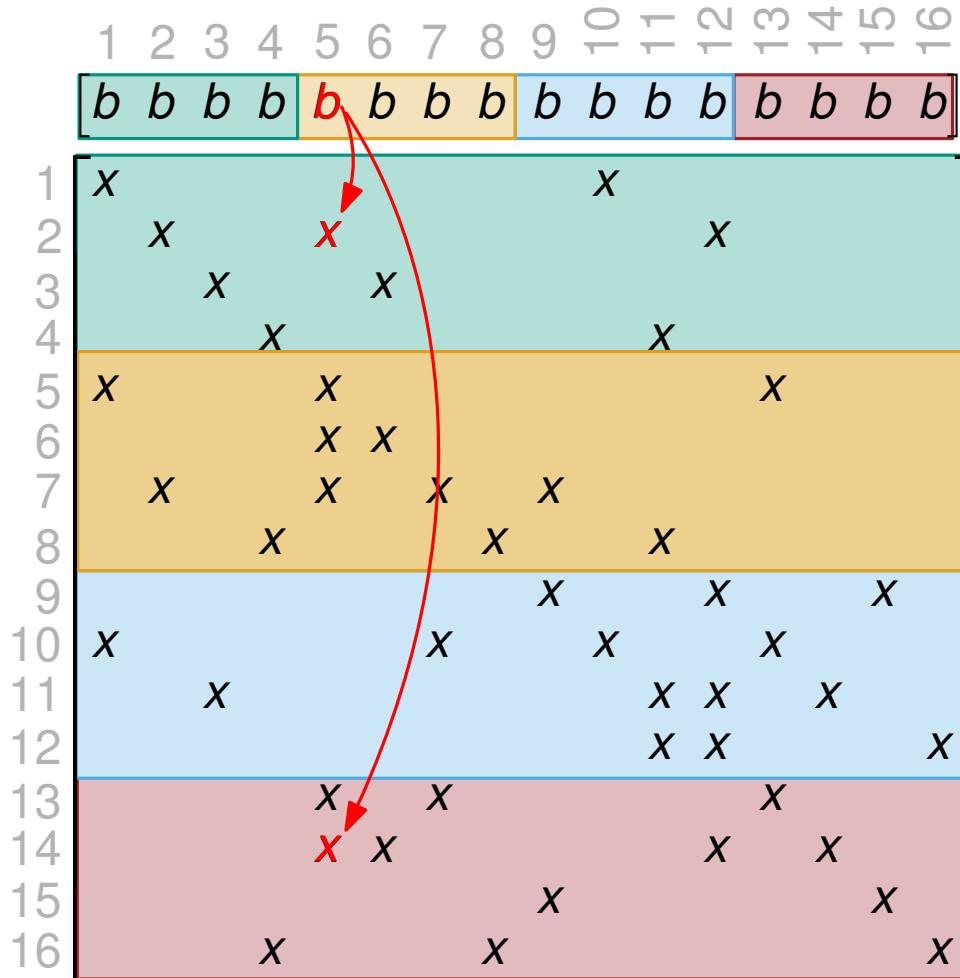
$\Rightarrow 12$

Communication Volume?

Naive Approach: Rowwise Decomposition

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

P_1
 P_2
 P_3
 P_4



Load Balancing?

$\Rightarrow 9$

$\Rightarrow 12$

$\Rightarrow 14$

$\Rightarrow 12$

Communication Volume?

Naive Approach: Rowwise Decomposition

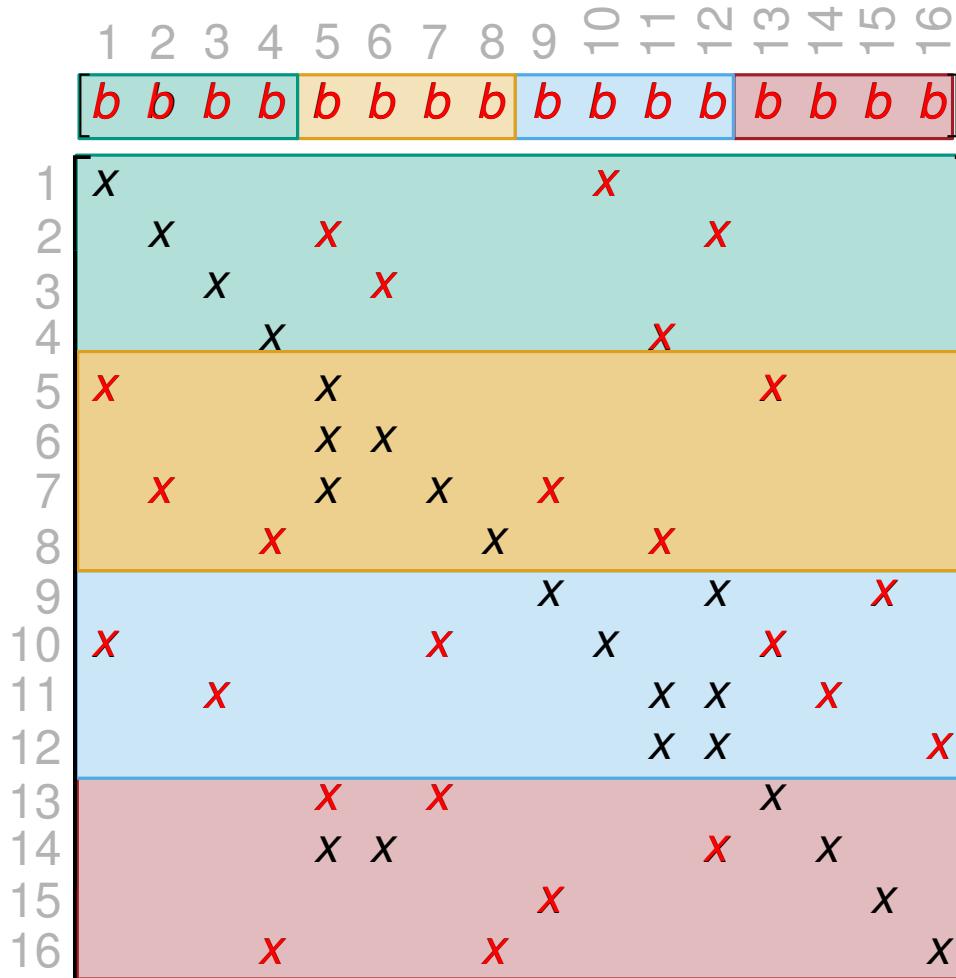
$A \in \mathbb{R}^{16 \times 16}$

P_1

P_2

P_3

P_4



Load Balancing?

$\Rightarrow 9$

$\Rightarrow 12$

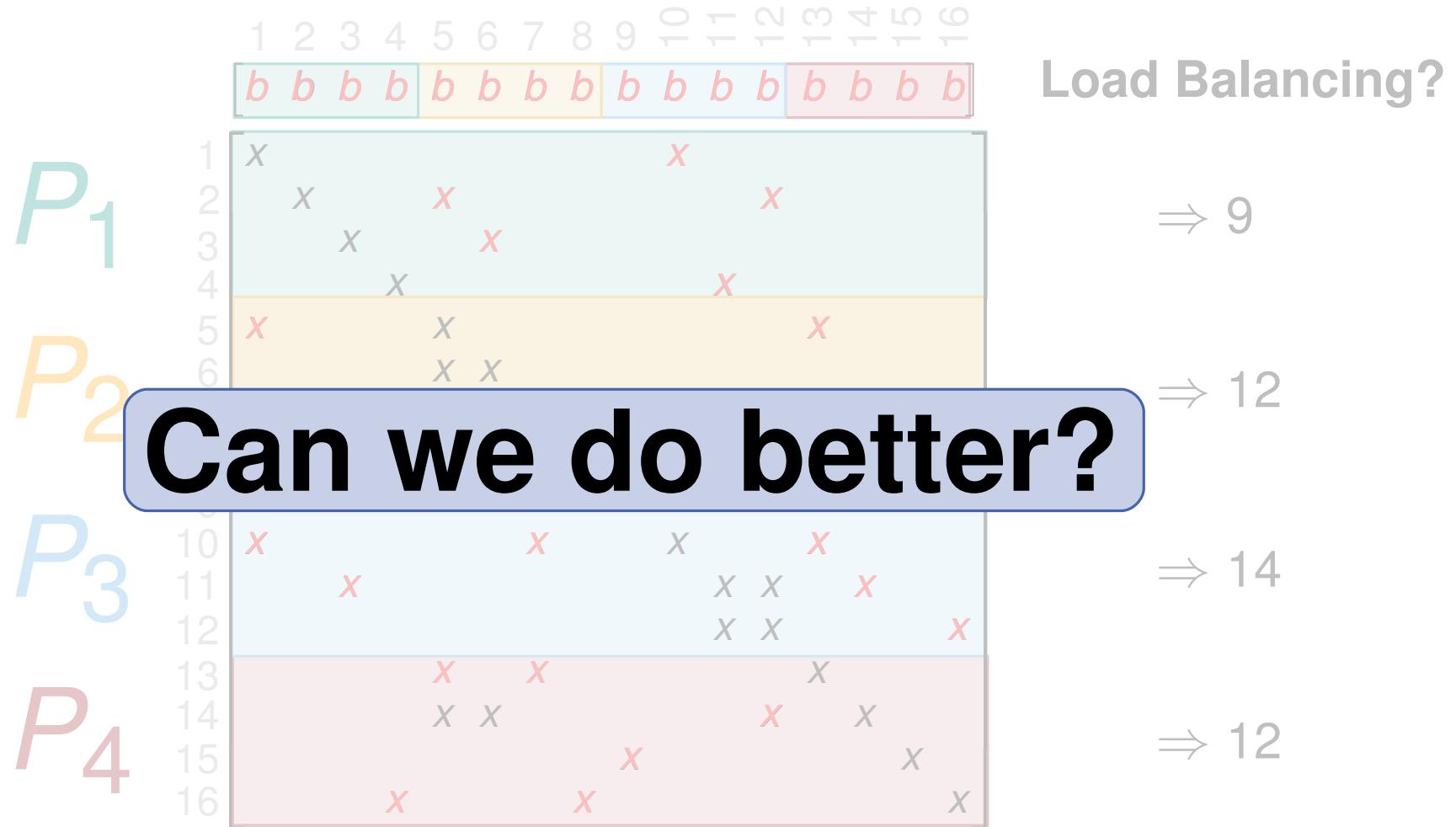
$\Rightarrow 14$

$\Rightarrow 12$

Communication Volume? $\Rightarrow 24$ entries!

Naive Approach: Rowwise Decomposition

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



Communication Volume? $\Rightarrow 24$ entries!

From SpM×V to Hypergraph Partitioning

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

- one vertex per row:

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

- one hyperedge per column:

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

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

From SpM \times V to Hypergraph Partitioning

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

- one vertex per row:

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

- one hyperedge per column:

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

$v_i \in V_R :$

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

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

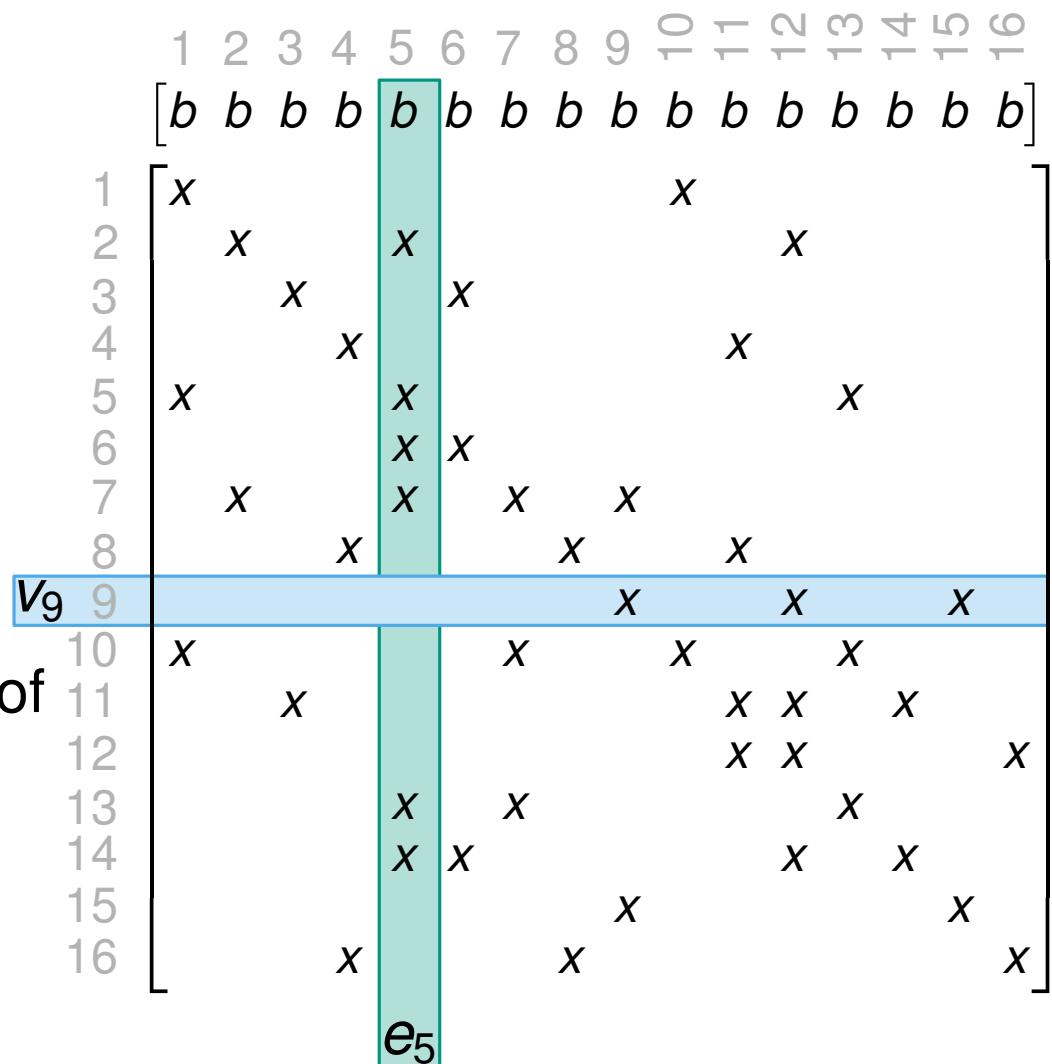
From SpM \times V to Hypergraph Partitioning

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

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

$v_i \in V_R :$

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



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

From SpM \times V to Hypergraph Partitioning

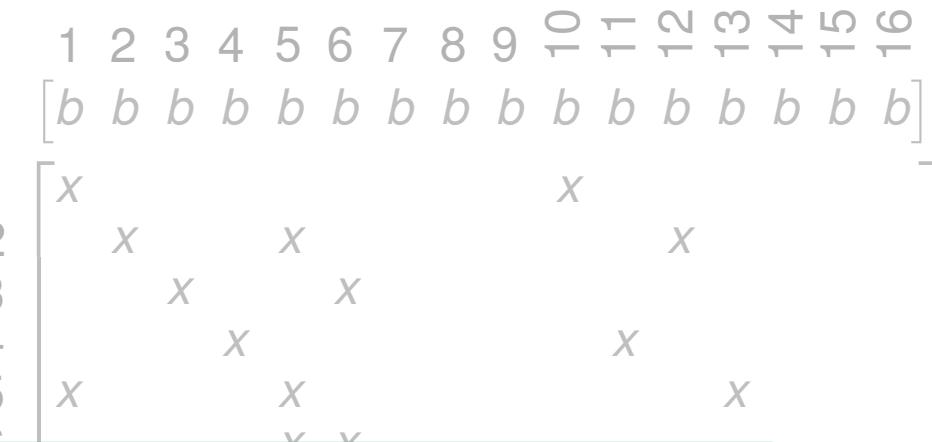
$$A \in \mathbf{R}^{16 \times 16} \Rightarrow H = (V_R, E_C)$$

- one vertex per row:

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

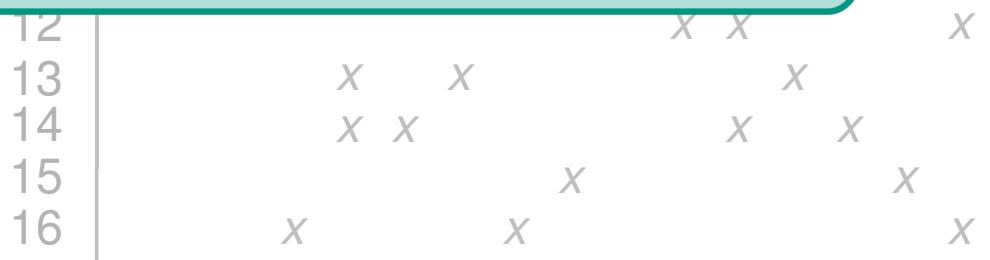
- one hyperedge per column:

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

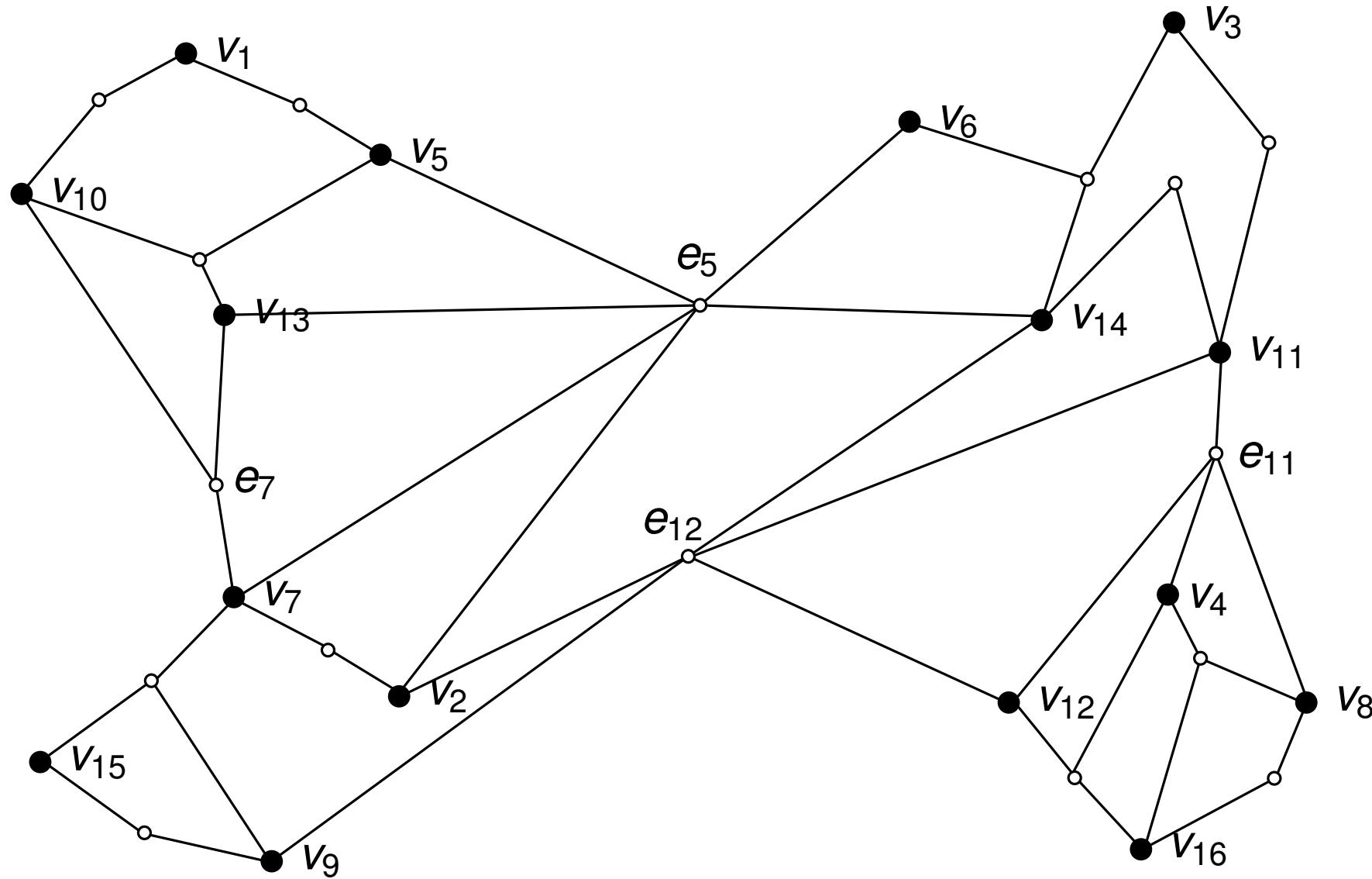


Solution: ε -balanced partition of H

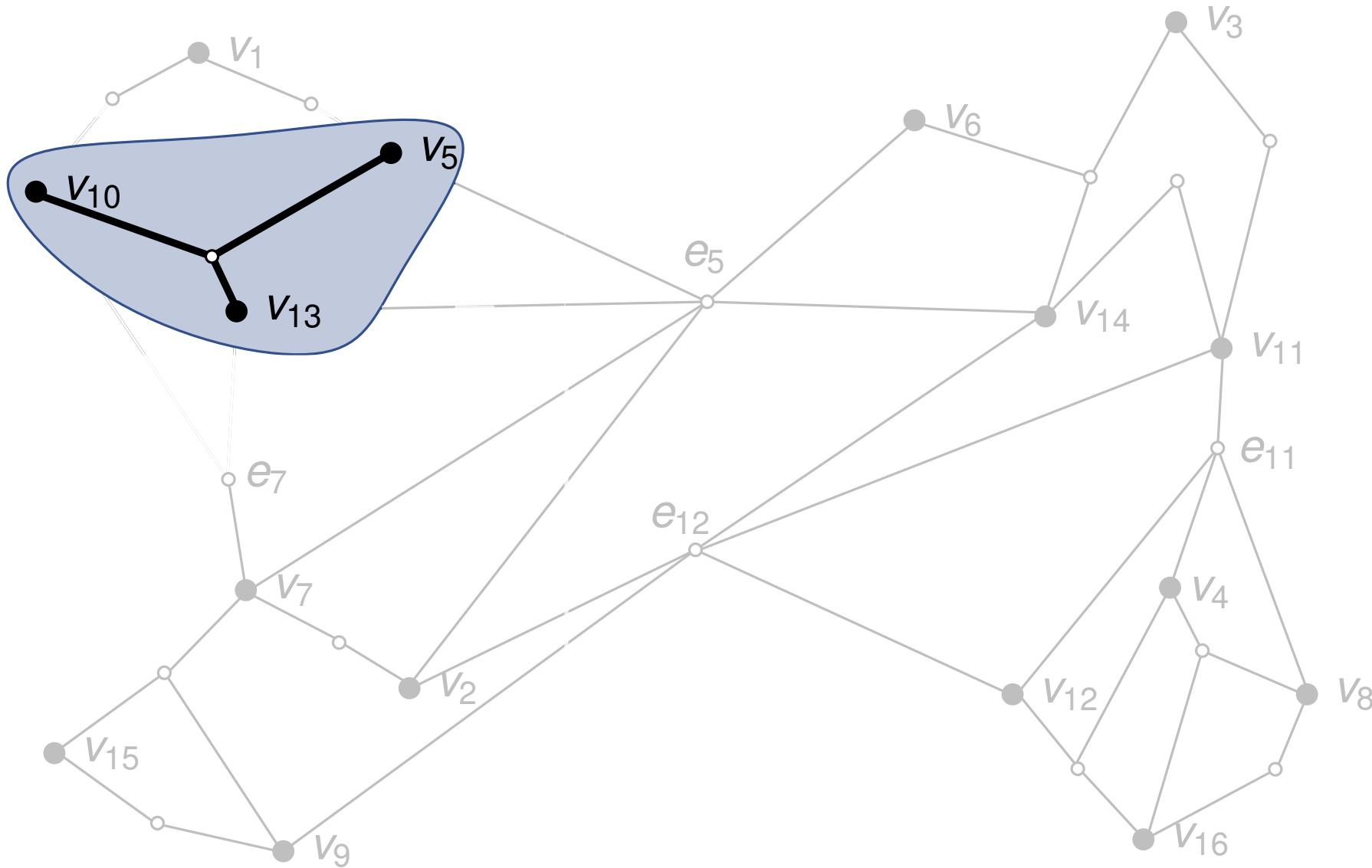
- balanced partition \rightsquigarrow computational load balance
- small $(\lambda - 1)$ -cutsize \rightsquigarrow minimizing communication volume



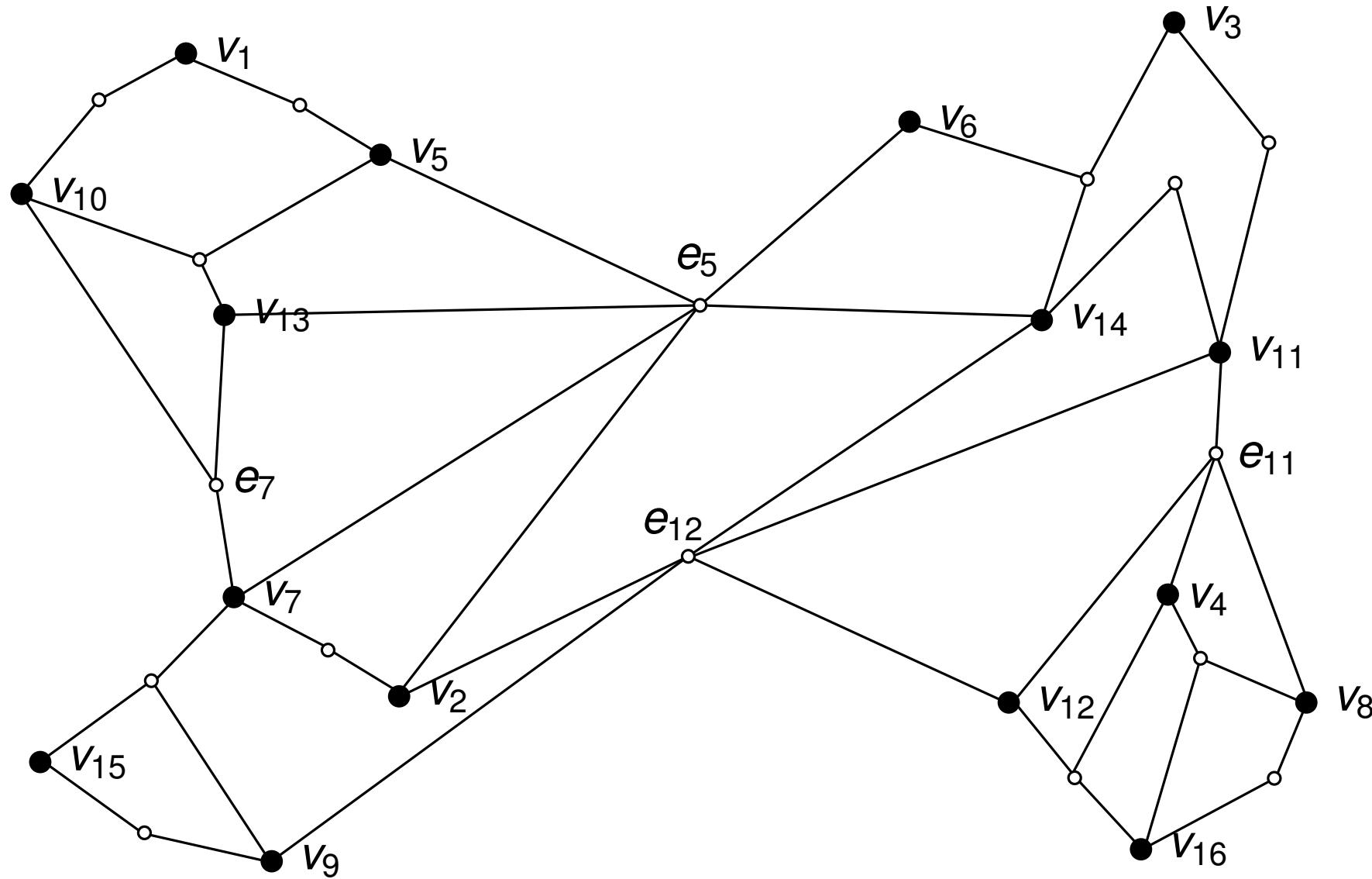
From SpM×V to Hypergraph Partitioning



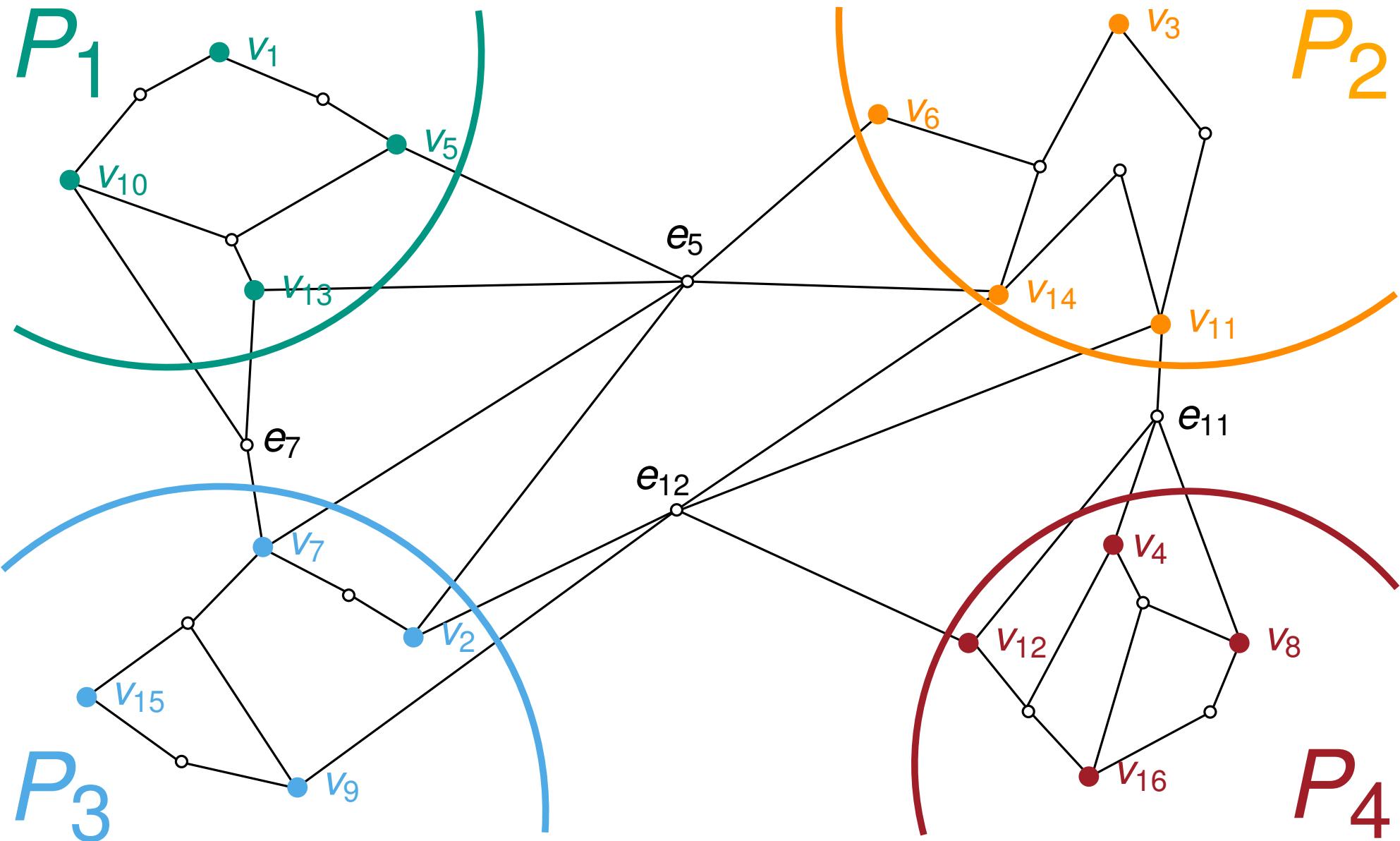
From SpM×V to Hypergraph Partitioning



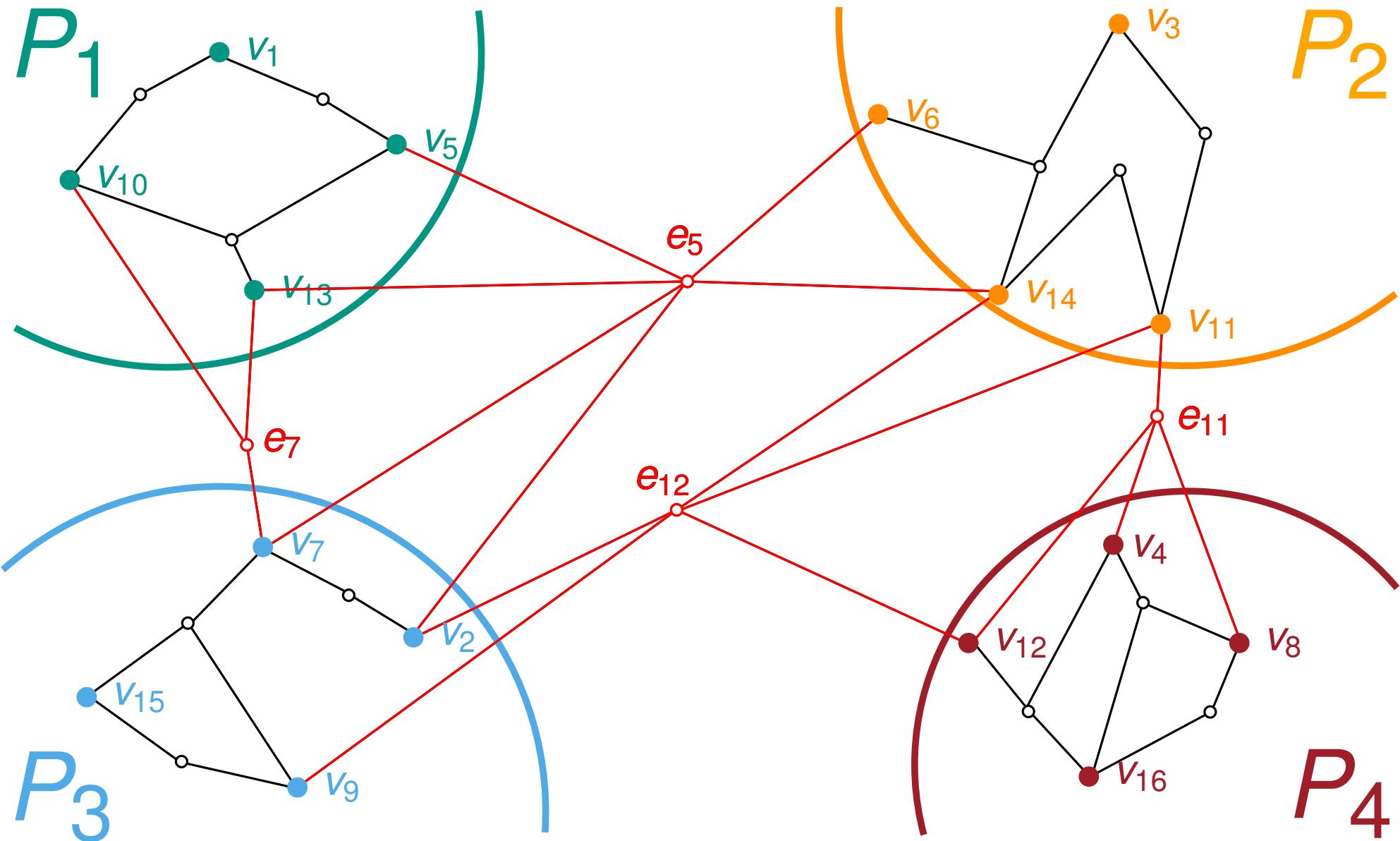
From SpM×V to Hypergraph Partitioning



From SpM×V to Hypergraph Partitioning



From SpM×V to Hypergraph Partitioning



From Hypergraph Partitioning to SpM \times V

P_1
 P_2
 P_3
 P_4

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

From Hypergraph Partitioning to SpM \times V

P_1
 P_2
 P_3
 P_4

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

Load Balancing?

From Hypergraph Partitioning to SpM×V

P_1
 P_2
 P_3
 P_4

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

Load Balancing?

⇒ 12

⇒ 12

⇒ 12

⇒ 12

From Hypergraph Partitioning to SpM \times V

Where are the cut-hyperedges?

P_1
 P_2
 P_3
 P_4

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

Load Balancing?

$\Rightarrow 12$

$\Rightarrow 12$

$\Rightarrow 12$

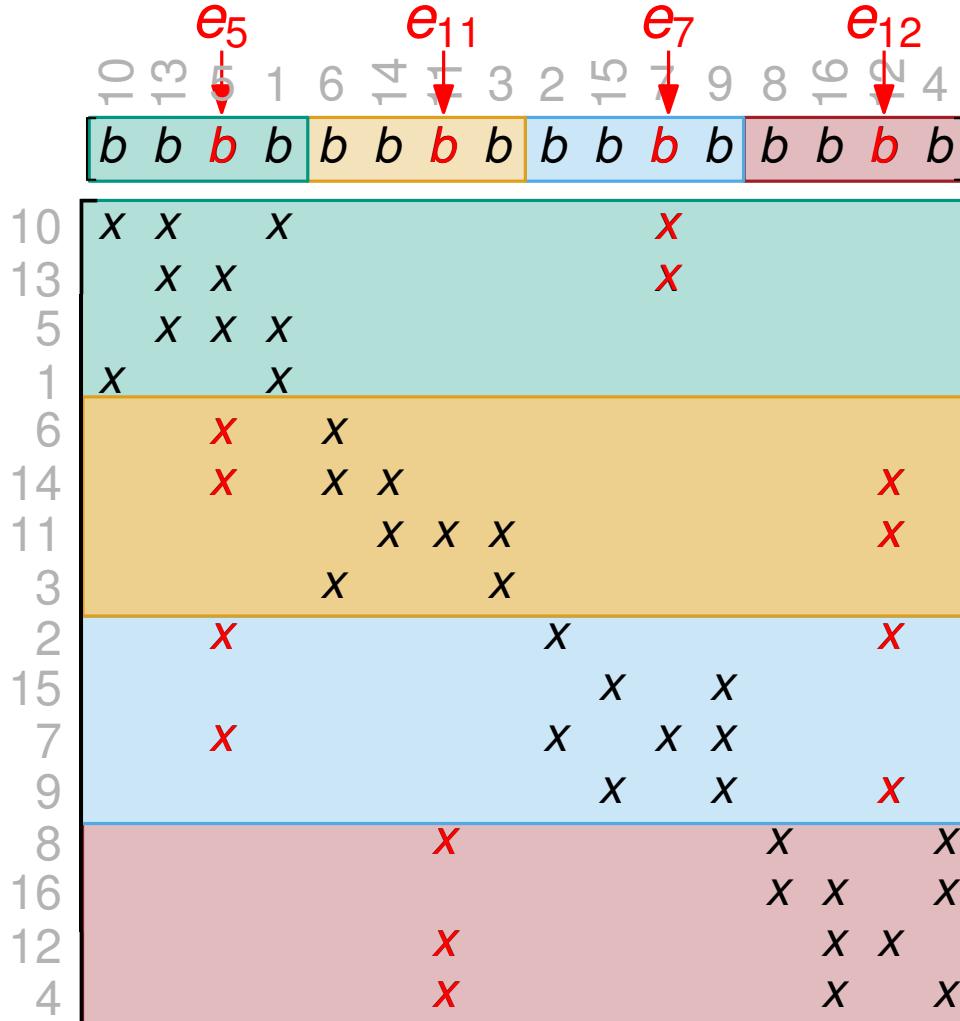
$\Rightarrow 12$

Communication Volume?

From Hypergraph Partitioning to SpM \times V

Where are the cut-hyperedges?

P_1
 P_2
 P_3
 P_4



Load Balancing?

$\Rightarrow 12$
 $\Rightarrow 12$
 $\Rightarrow 12$
 $\Rightarrow 12$

Communication Volume? $\Rightarrow 6$ entries!

How does Hypergraph Partitioning work?

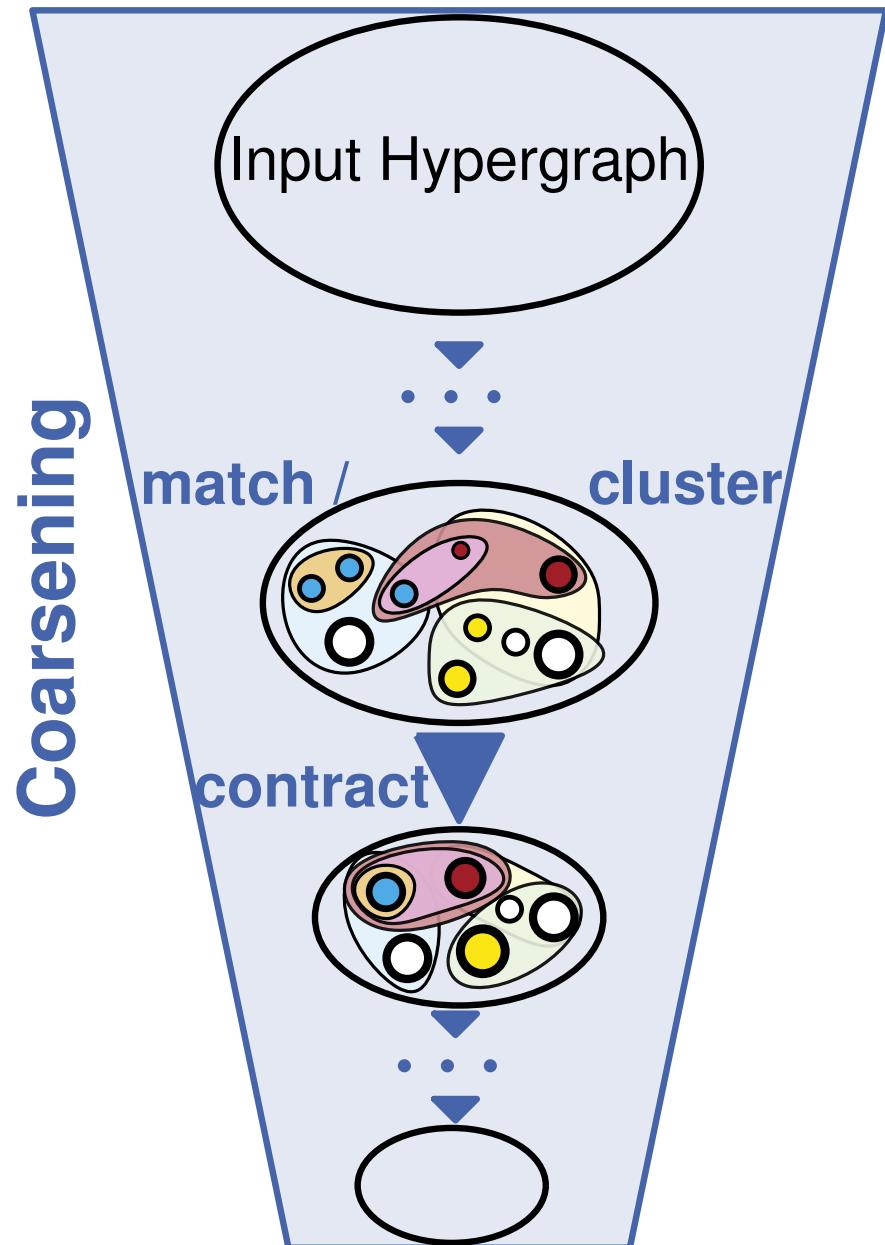
How does

Bad News:

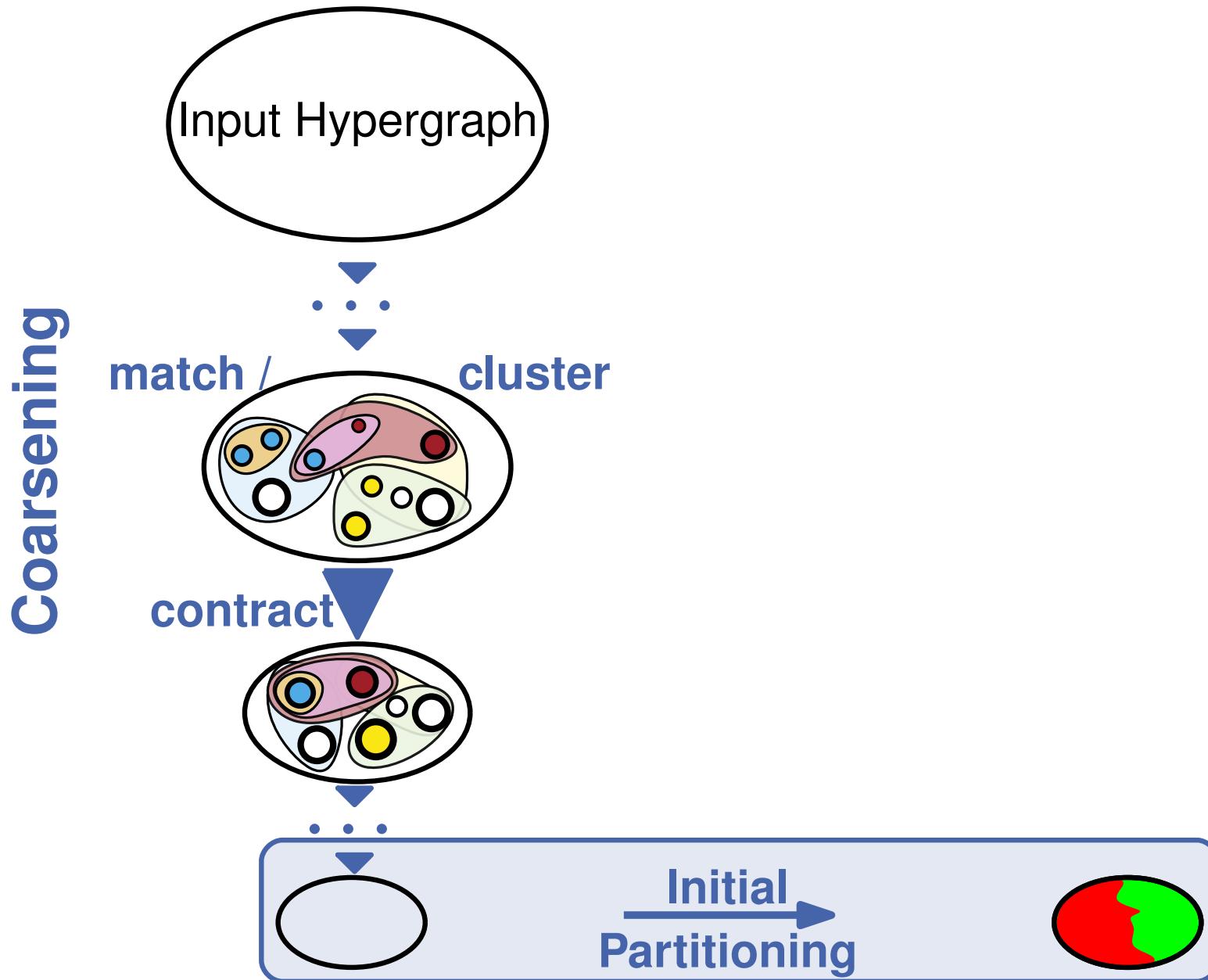
- hypergraph partitioning is **NP-hard**
- even finding **good approximate** solutions for graphs is **NP-hard**

work?

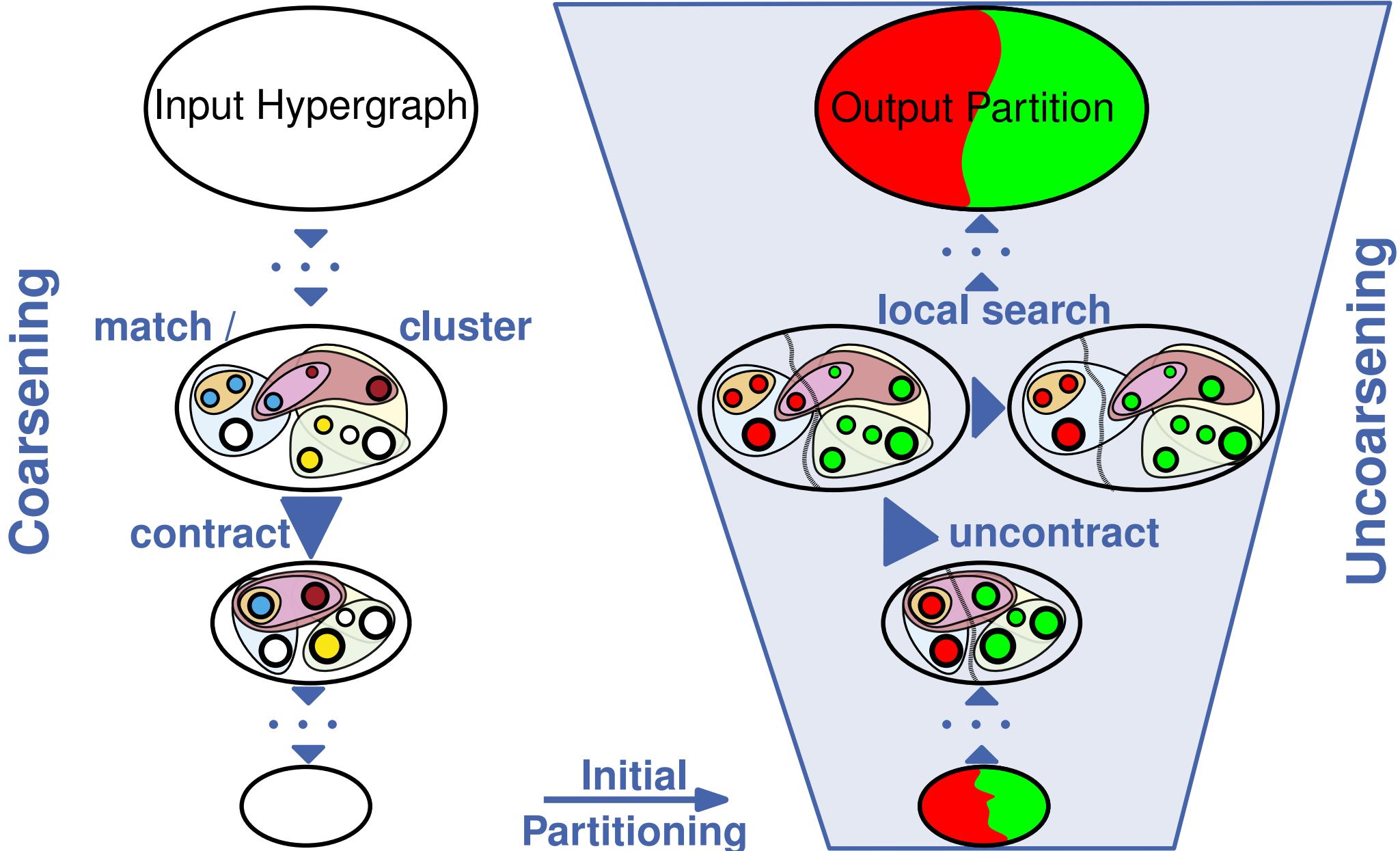
Successful Heuristic: Multilevel Paradigm



Successful Heuristic: Multilevel Paradigm



Successful Heuristic: Multilevel Paradigm



Coarsening

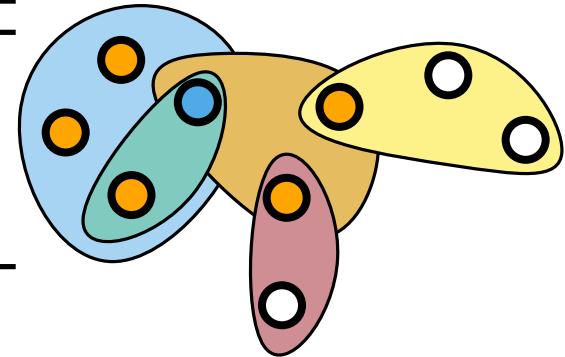
Clustering-based Coarsening

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

Objective: identify highly connected vertices

```
foreach vertex v do
    cluster[v] := argmax rating(v,u)
        neighbor u
```

using...



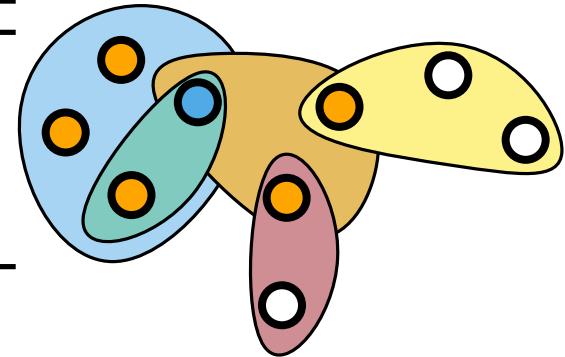
Clustering-based Coarsening

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

Objective: identify highly connected vertices

```
foreach vertex  $v$  do
    cluster[ $v$ ] := argmax rating( $v, u$ )
        neighbor  $u$ 
```

using...



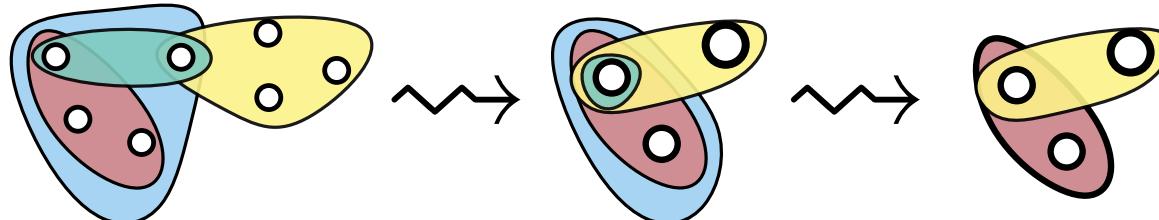
Main Design Goals: [Karypis, Kumar 99]

- 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:

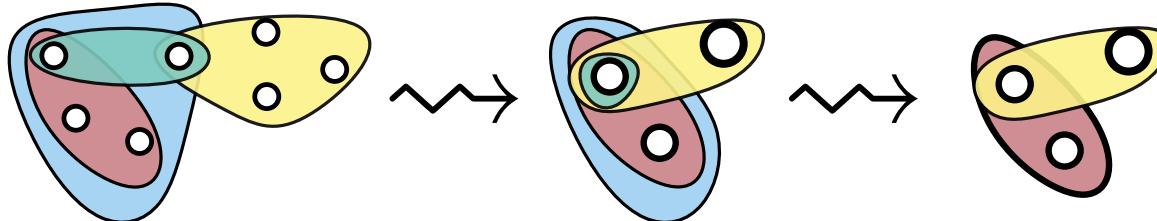
- 1: reduce **size** of nets \rightsquigarrow easier local search
- 2: reduce **number** of nets \rightsquigarrow easier initial partitioning



Clustering-based Coarsening

Main Design Goals:

- 1: reduce **size** of nets \rightsquigarrow easier local search
- 2: reduce **number** of nets \rightsquigarrow easier initial partitioning



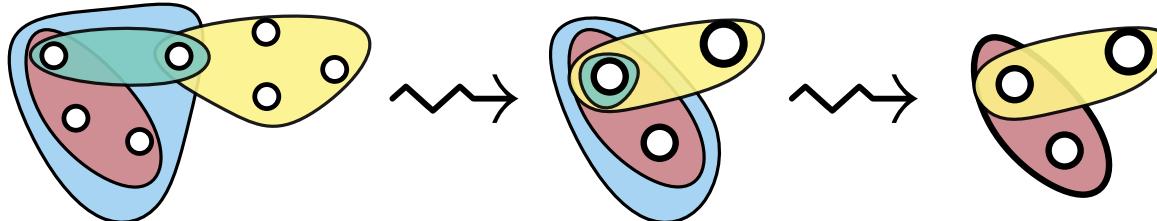
\Rightarrow hypergraph-tailored rating functions:

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

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:

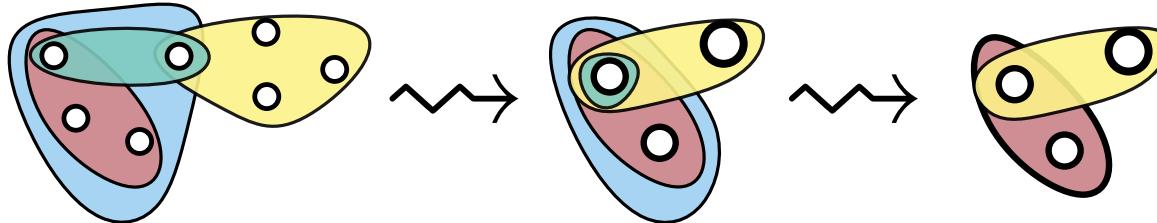
$$r(u, v) := \sum_{\substack{\text{net } e \\ \text{containing } u, v}} \frac{\omega(e)}{|e|-1}$$

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



\Rightarrow hypergraph-tailored rating functions:

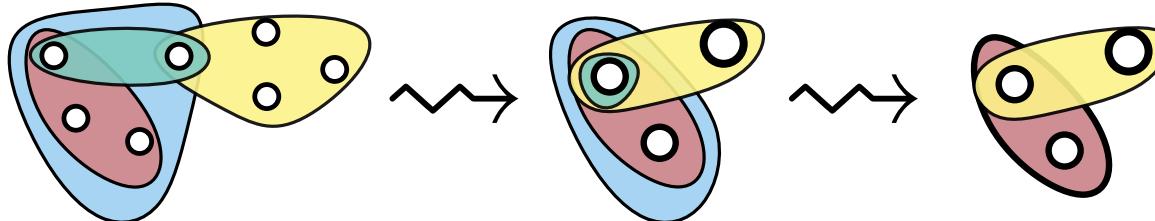
$$r(u, v) := \sum_{\substack{\text{net } e \\ \text{containing } u, v}} \frac{\omega(e)}{|e|-1}$$

large number ... \longrightarrow of heavy nets ...

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:

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

large number ... \longrightarrow of heavy nets ...
 \longrightarrow ... with small size

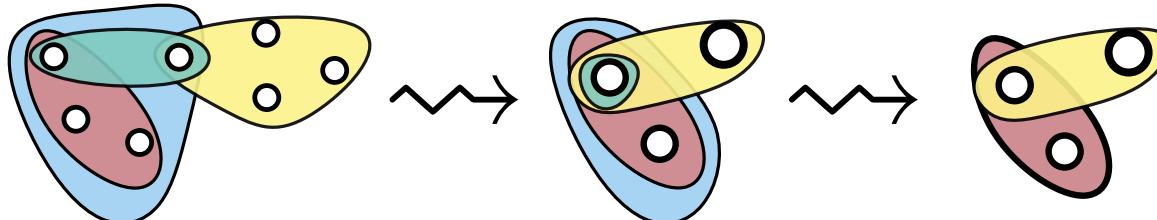
Clustering-based Coarsening

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:

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

large number ... \longrightarrow of heavy nets ...
 \longrightarrow ... with small size

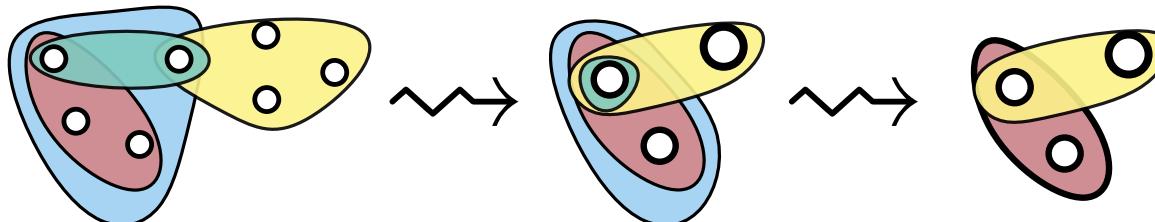
Clustering-based Coarsening

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:

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

large number ... \longrightarrow of heavy nets ...
 \longrightarrow ... with small size

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

\Rightarrow prefer clustering over matching

\Rightarrow ensure \sim balanced vertex weights

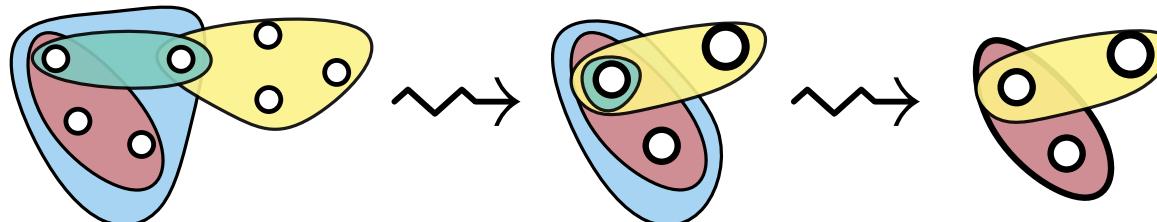
Clustering-based Coarsening

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:

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

large number ... \longrightarrow of heavy nets ...
 \longrightarrow ... with small size

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

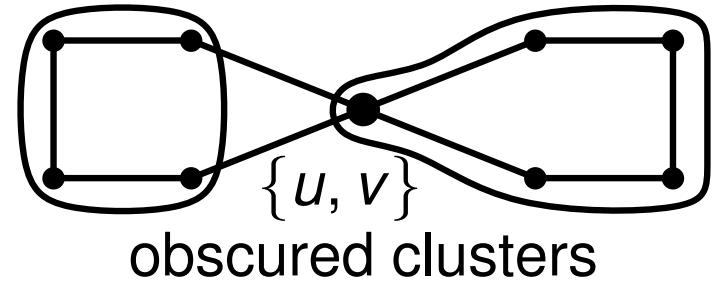
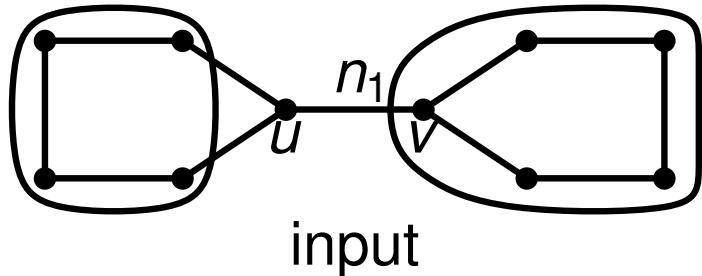
\Rightarrow prefer clustering over matching

\Rightarrow ensure \sim balanced vertex weights

enough?

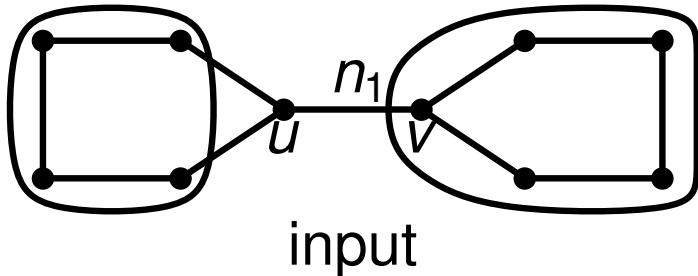
What could possibly go wrong?

... a lot:

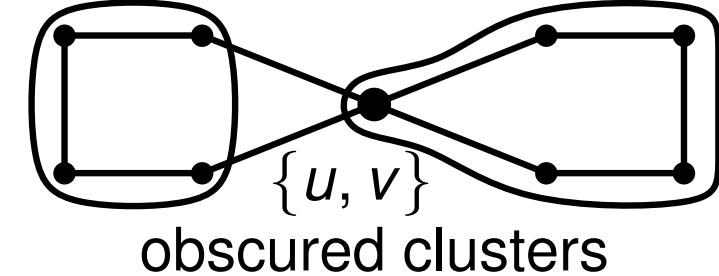


What could possibly go wrong?

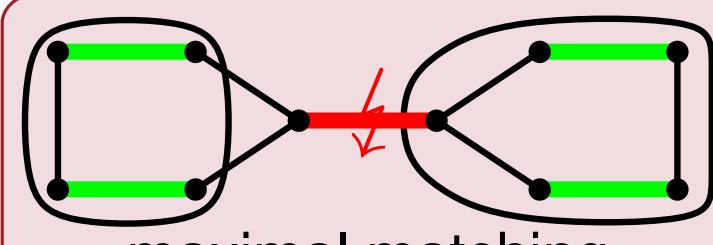
... a lot:



input

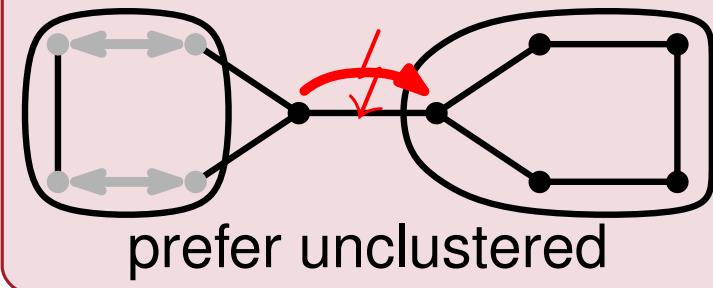


obscured clusters

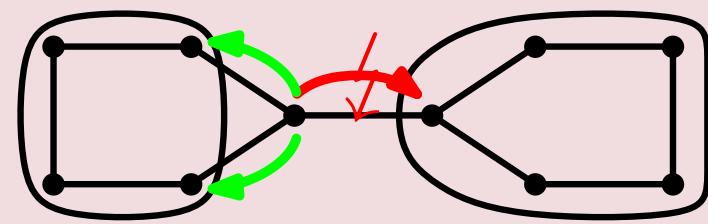


maximal matching

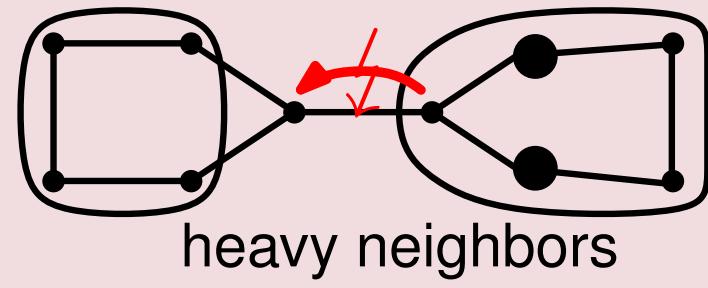
ISSUES



prefer unclustered



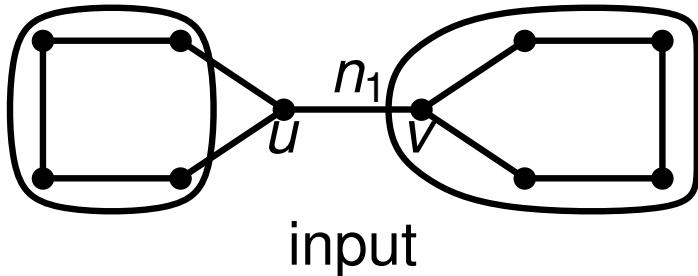
random tie-breaking



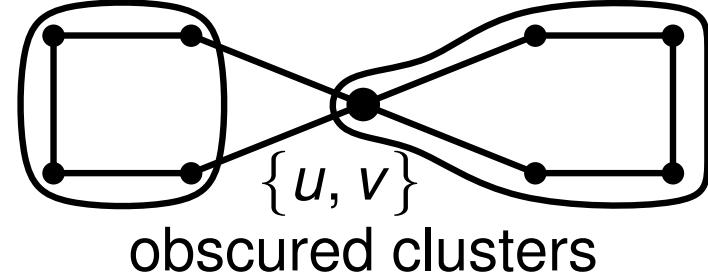
heavy neighbors

What could possibly go wrong?

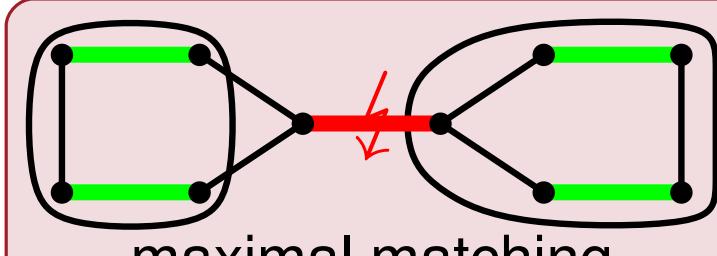
... a lot:



input

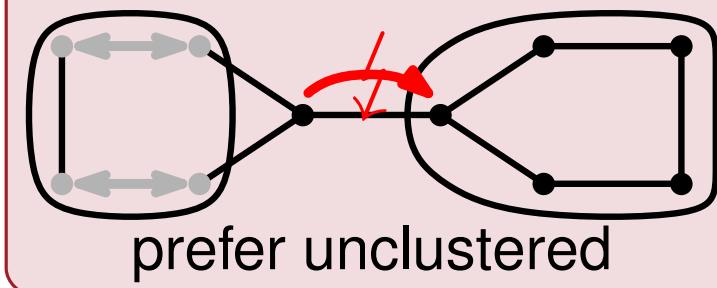


obscured clusters

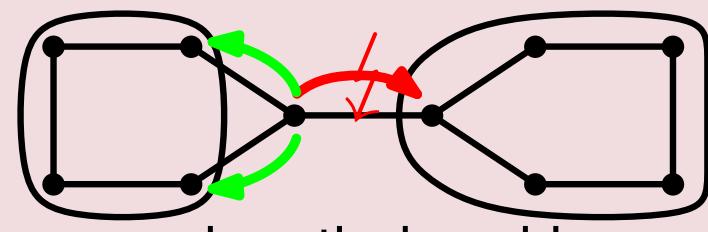


maximal matching

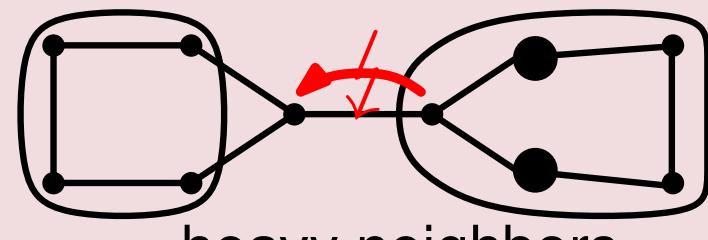
ISSUES



prefer unclustered



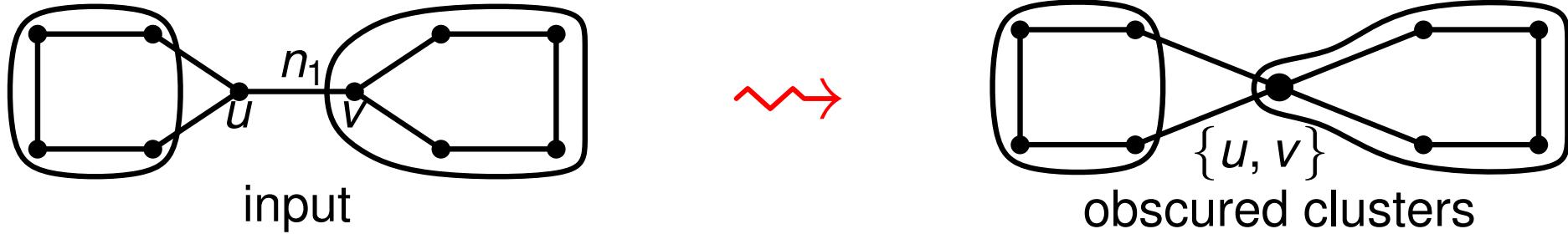
random tie-breaking



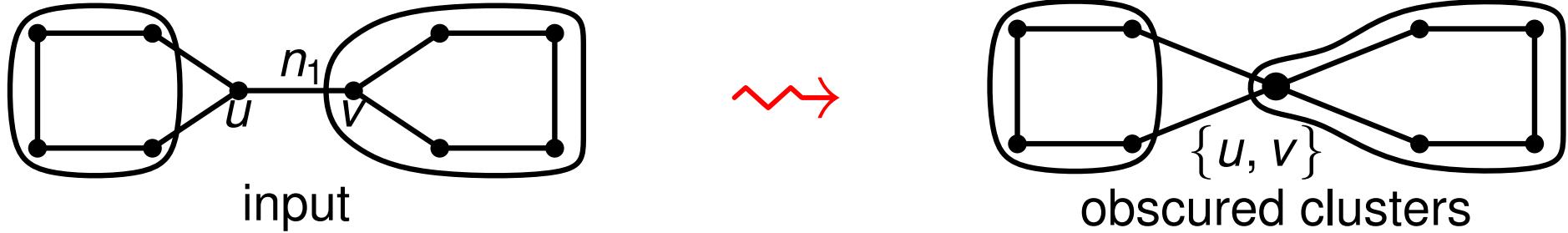
heavy neighbors

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

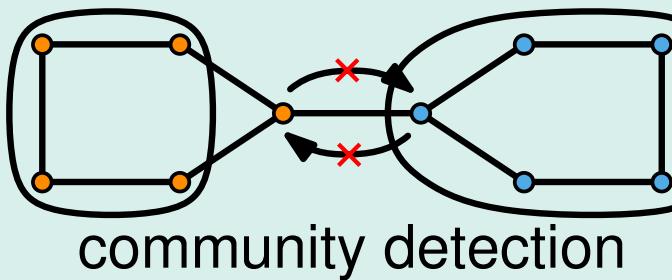
Community-aware Coarsening



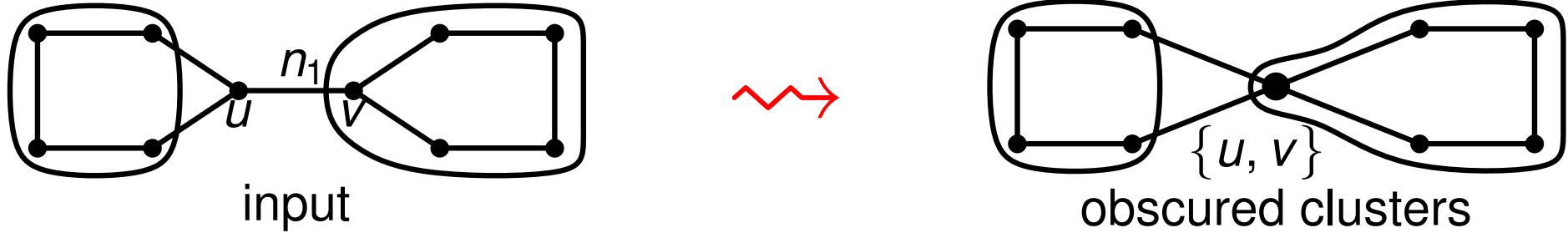
Community-aware Coarsening



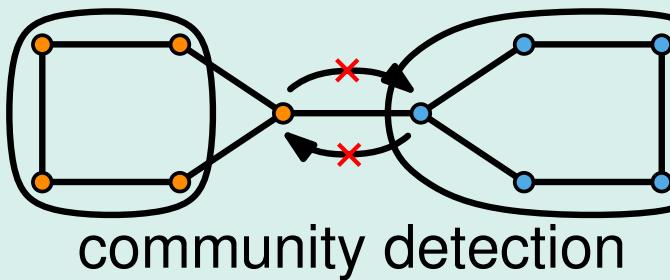
SOLUTION



Community-aware Coarsening



SOLUTION



Framework:

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

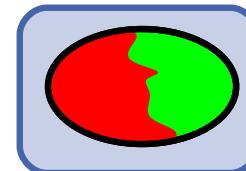
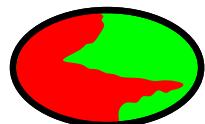
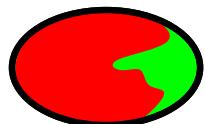
Initial Partitioning

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



initial partition

Local Search

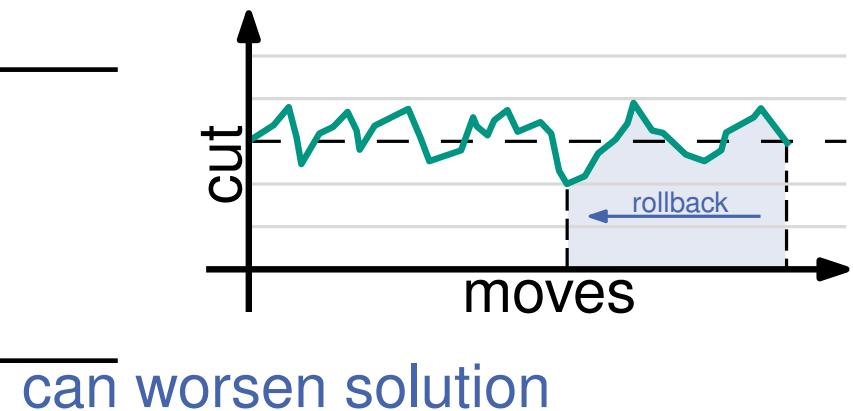
Fiduccia-Mattheyses Algorithm

Algorithm 1: FM Local Search

while $\neg done$ **do**

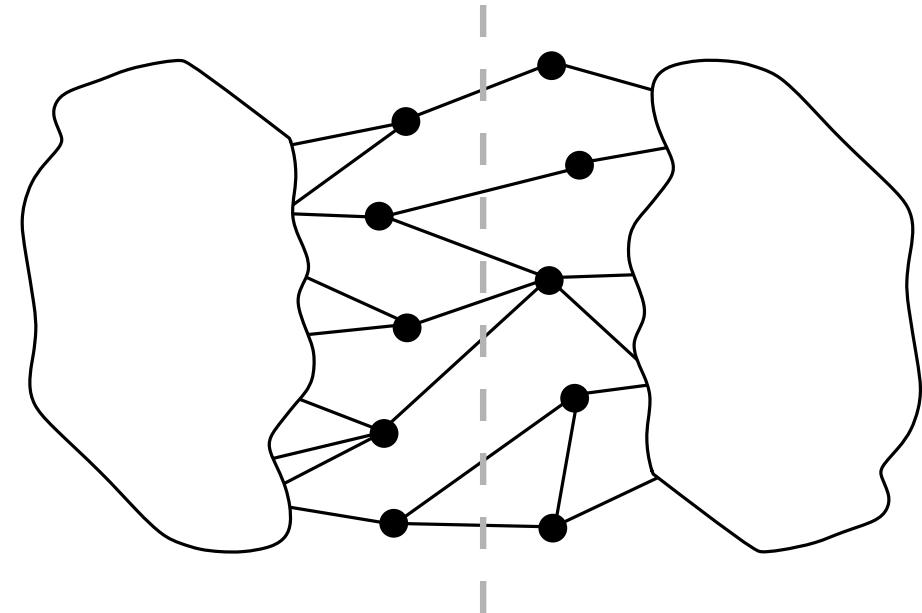
 | find best move
 | perform best move

rollback to best solution



Example for Graphs:

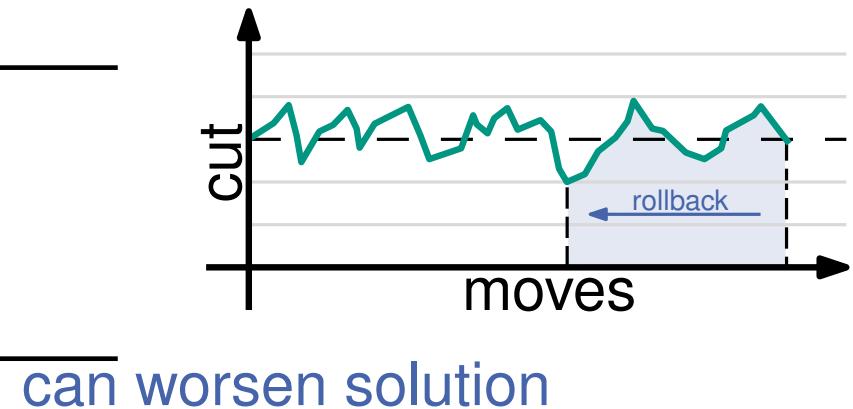
- compute gain $g(v) = d_{\text{ext}}(v) - d_{\text{int}}(v)$
- alternate between blocks
- edge-cut: 7



Fiduccia-Mattheyses Algorithm

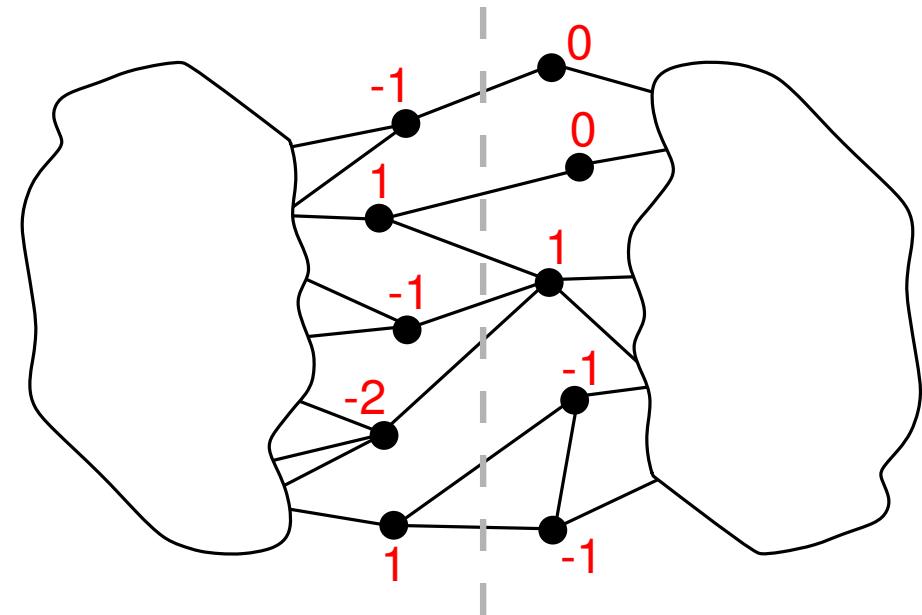
Algorithm 1: FM Local Search

```
while  $\neg done$  do
    find best move
    perform best move
    rollback to best solution
```



Example for Graphs:

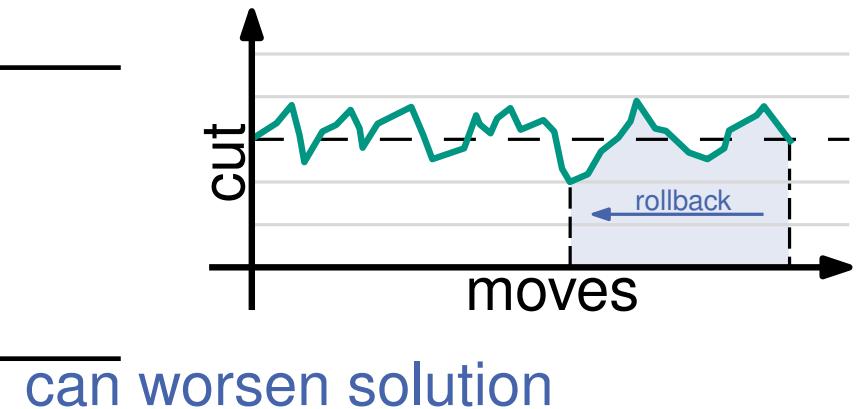
- compute gain $g(v) = d_{\text{ext}}(v) - d_{\text{int}}(v)$
- alternate between blocks
- edge-cut: 7



Fiduccia-Mattheyses Algorithm

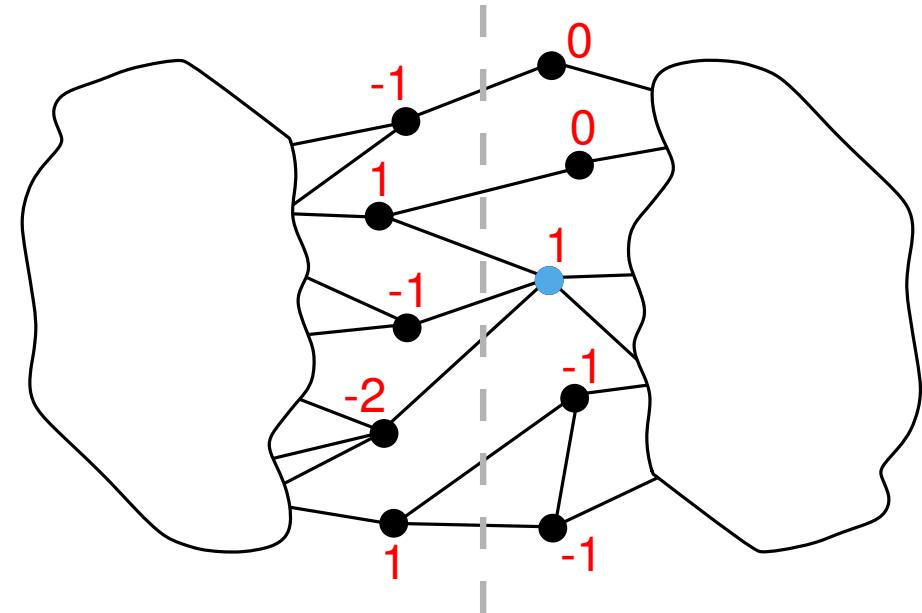
Algorithm 1: FM Local Search

```
while  $\neg done$  do
    find best move
    perform best move
    rollback to best solution
```



Example for Graphs:

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



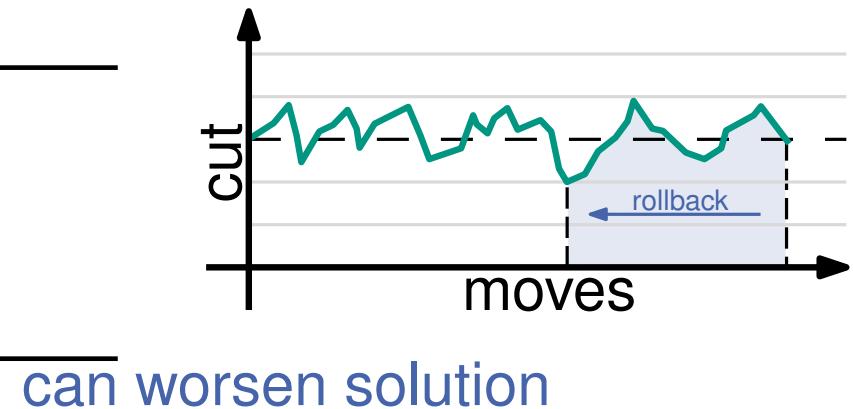
Fiduccia-Mattheyses Algorithm

Algorithm 1: FM Local Search

while $\neg done$ **do**

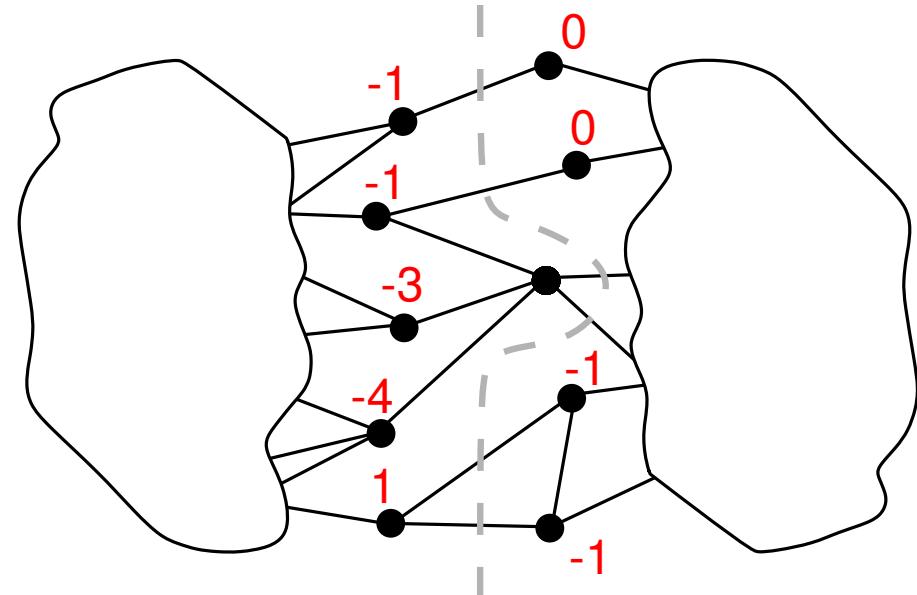
 | find best move
 | perform best move

rollback to best solution



Example for Graphs:

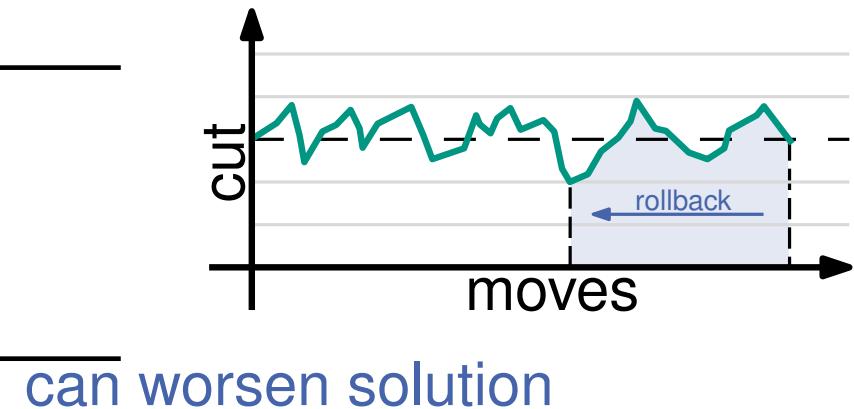
- **recalculate** gain $g(v)$ of neighbors
- move each node at most once
- edge-cut: 7, 6



Fiduccia-Mattheyses Algorithm

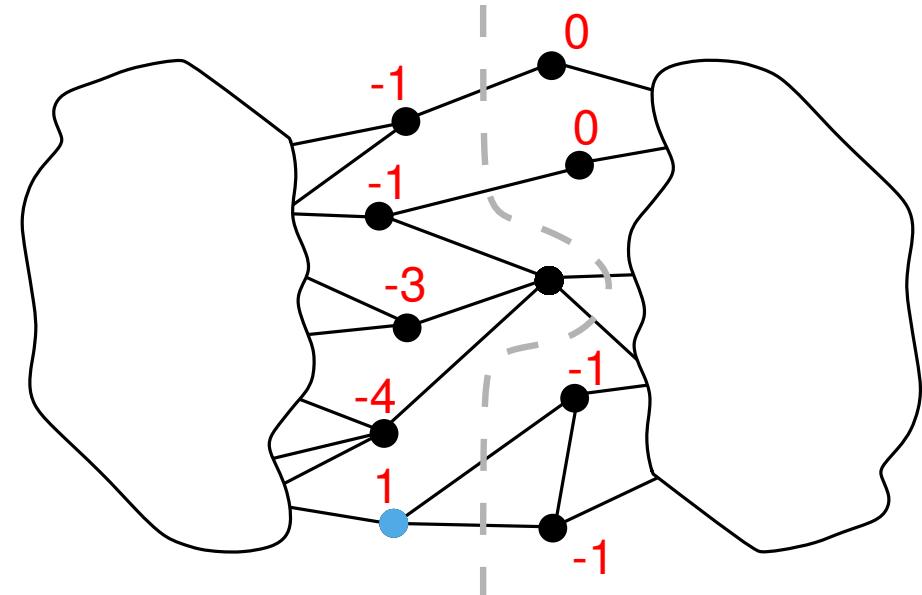
Algorithm 1: FM Local Search

```
while  $\neg done$  do
    find best move
    perform best move
    rollback to best solution
```



Example for Graphs:

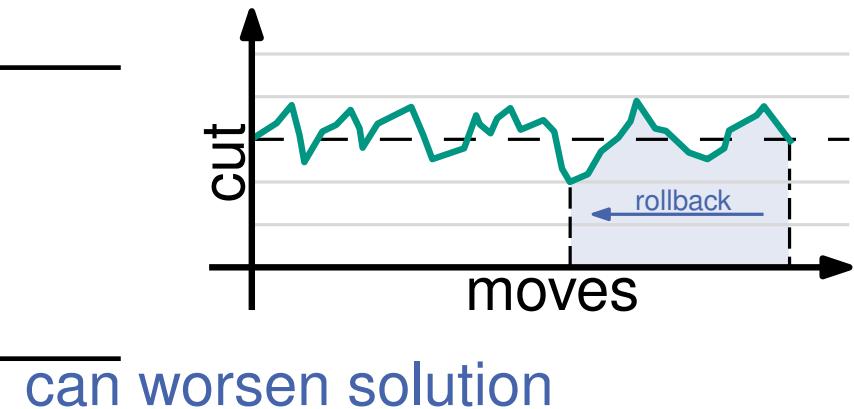
- **recalculate** gain $g(v)$ of neighbors
- move each node at most once
- edge-cut: 7, 6



Fiduccia-Mattheyses Algorithm

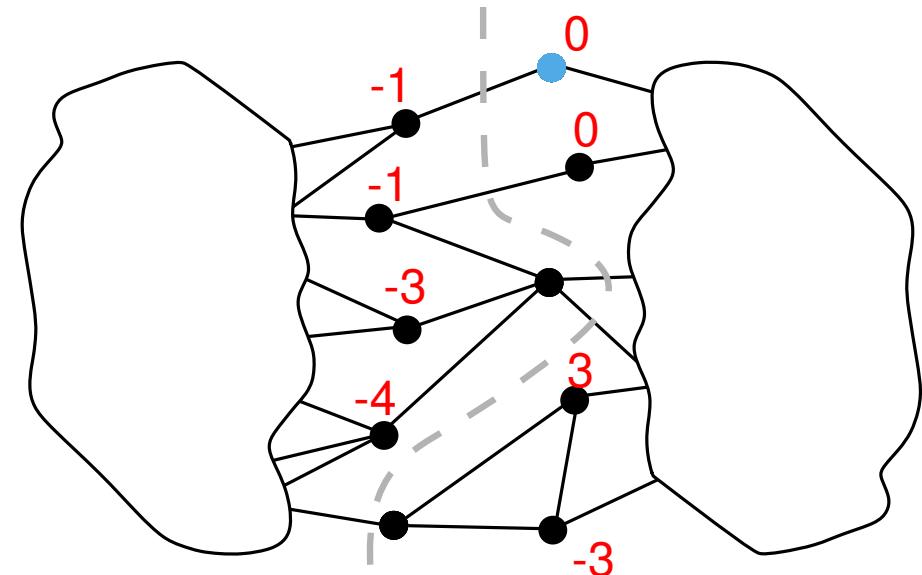
Algorithm 1: FM Local Search

```
while  $\neg done$  do
    find best move
    perform best move
    rollback to best solution
```



Example for Graphs:

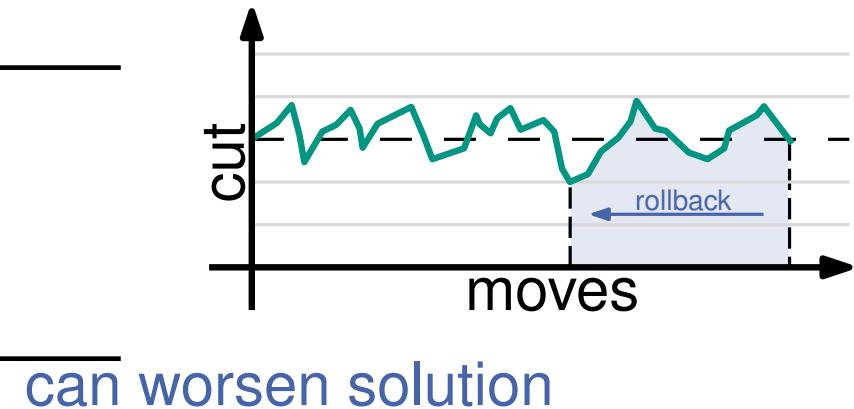
- recalculate gain $g(v)$ of neighbors
- move each node at most once
- edge-cut: 7, 6, 5



Fiduccia-Mattheyses Algorithm

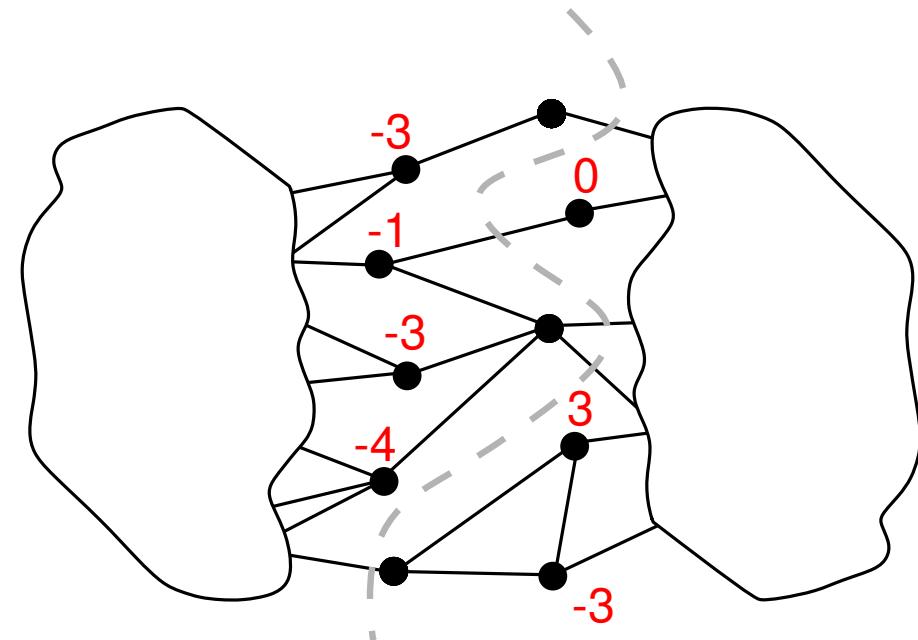
Algorithm 1: FM Local Search

```
while  $\neg done$  do
    find best move
    perform best move
    rollback to best solution
```



Example for Graphs:

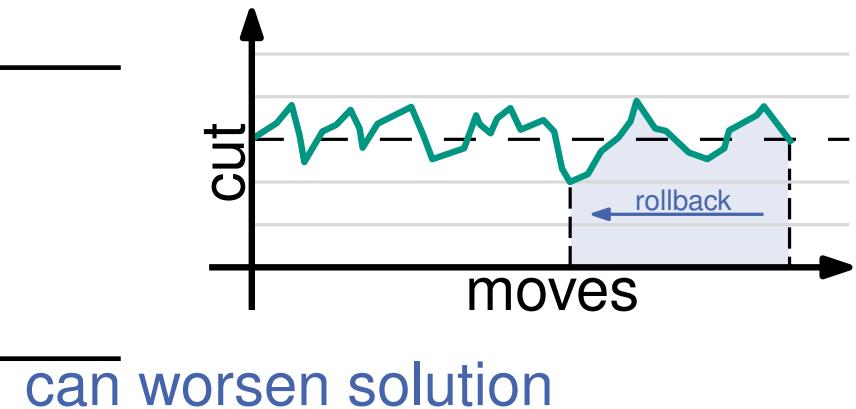
- **recalculate** gain $g(v)$ of neighbors
- move each node at most once
- edge-cut: 7, 6, 5, 5



Fiduccia-Mattheyses Algorithm

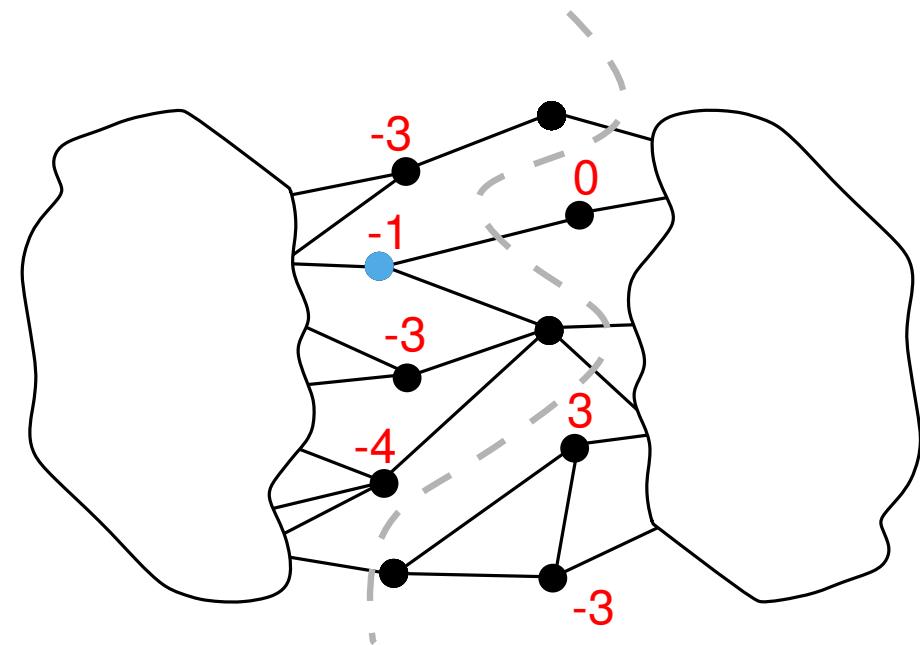
Algorithm 1: FM Local Search

```
while  $\neg done$  do
    find best move
    perform best move
    rollback to best solution
```



Example for Graphs:

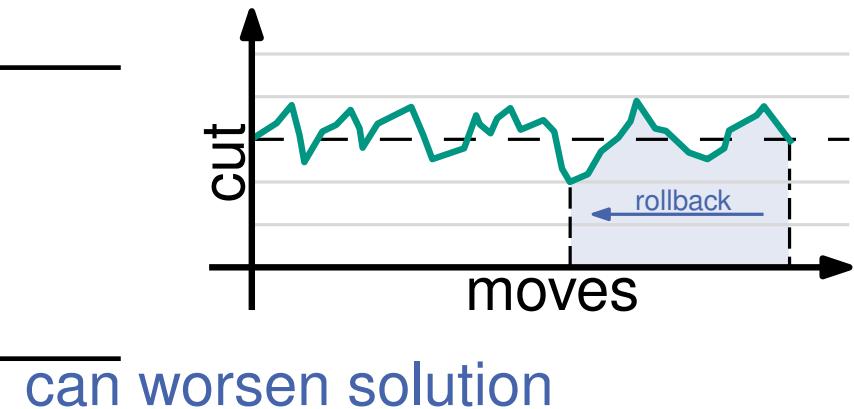
- **recalculate** gain $g(v)$ of neighbors
- move each node at most once
- edge-cut: 7, 6, 5, 5



Fiduccia-Mattheyses Algorithm

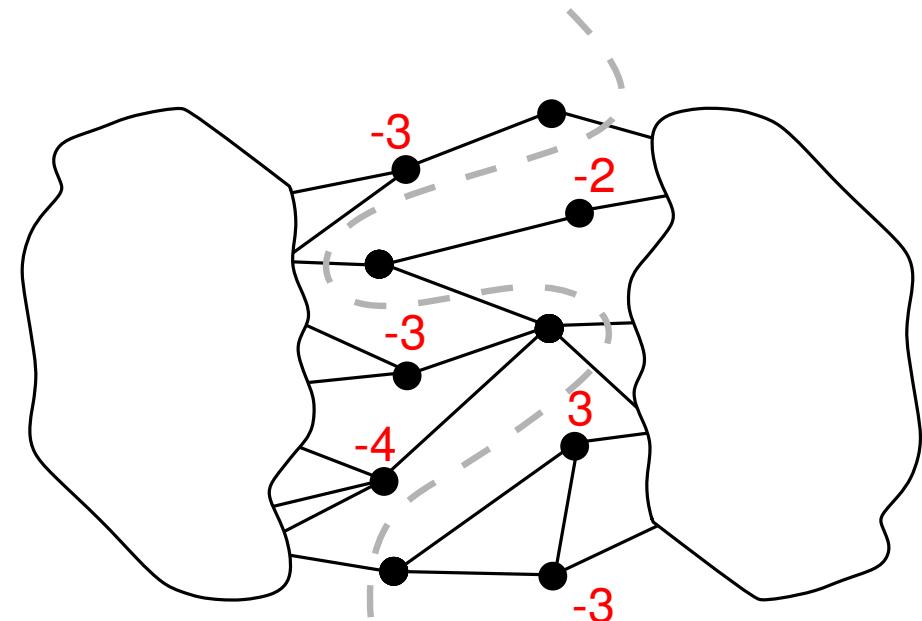
Algorithm 1: FM Local Search

```
while  $\neg done$  do
    find best move
    perform best move
    rollback to best solution
```



Example for Graphs:

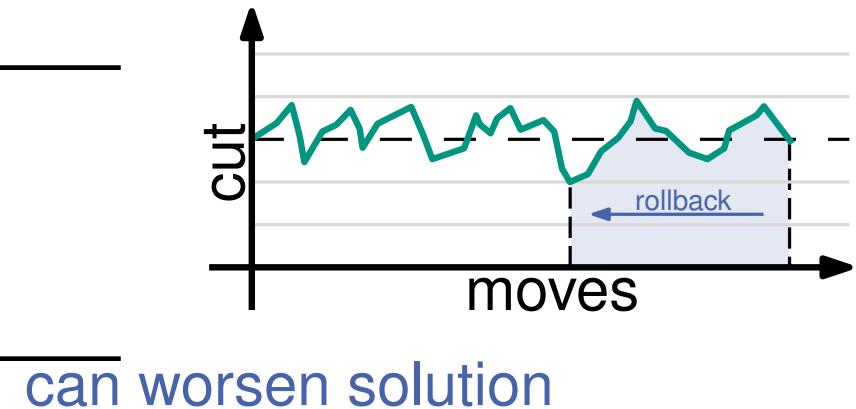
- recalculate gain $g(v)$ of neighbors
- move each node at most once
- edge-cut: 7, 6, 5, 5, 6



Fiduccia-Mattheyses Algorithm

Algorithm 1: FM Local Search

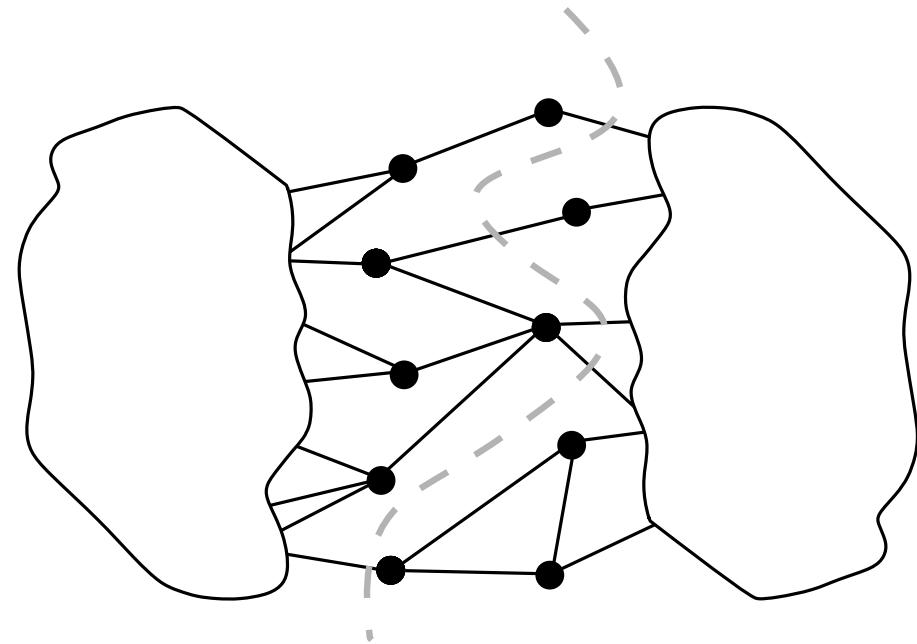
```
while  $\neg done$  do
    find best move
    perform best move
    rollback to best solution
```



Example for Graphs:

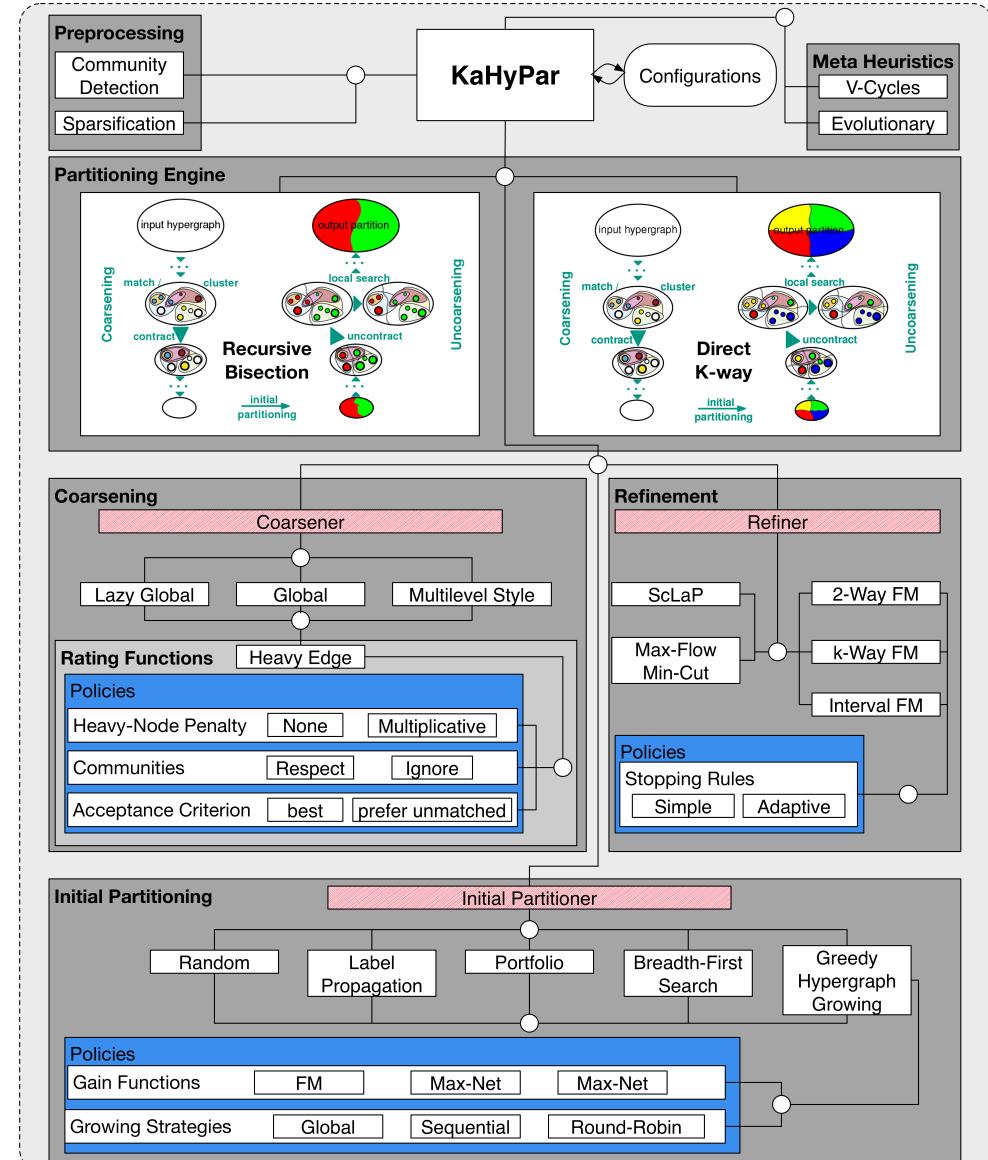
- **recalculate** gain $g(v)$ of neighbors
- move each node at most once
- edge-cut: 7, 6, 5, 5, 6

rollback



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>



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.