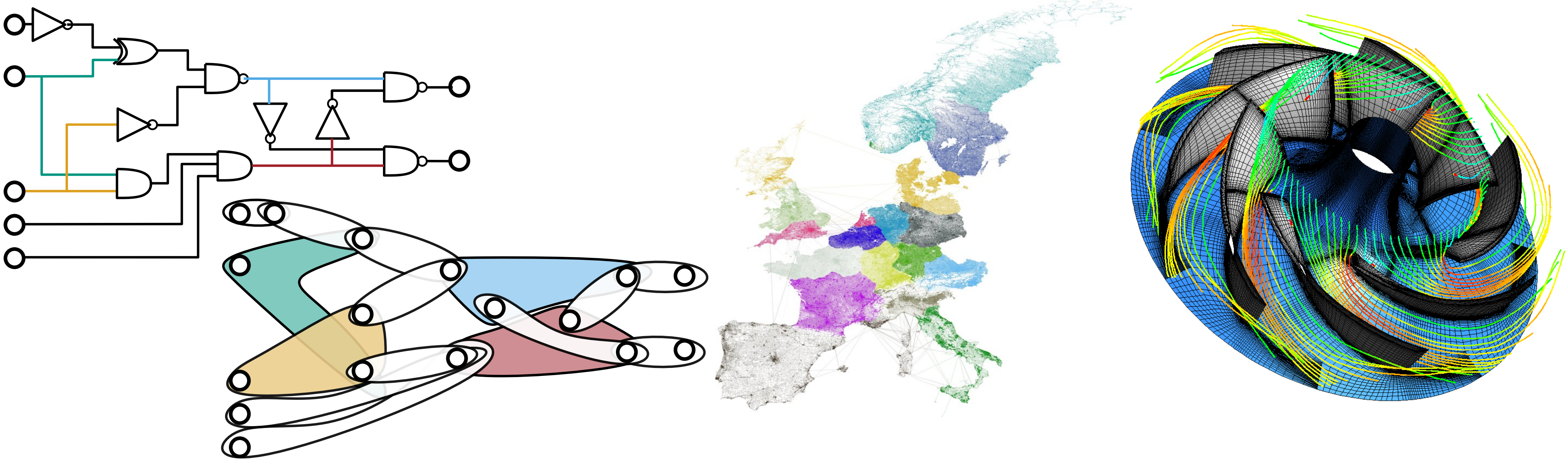
Brief Introduction to Hypergraph Partitioning

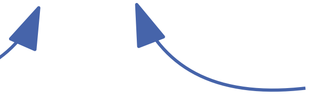
Bioinformatics Programming Practical Kickoff Meeting · April 19, 2018
Sebastian Schlag

INSTITUTE OF THEORETICAL INFORMATICS · ALGORITHMICS GROUP

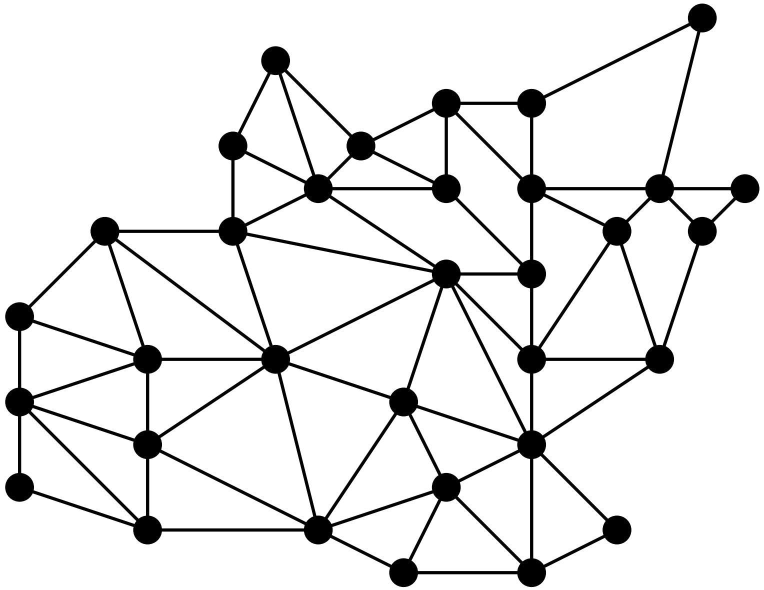


Graphs and Hypergraphs

Graph $G = (V, E)$



- models **relationships** between **objects**
- dyadic (**$\mathbf{2}$-ary**) relationships



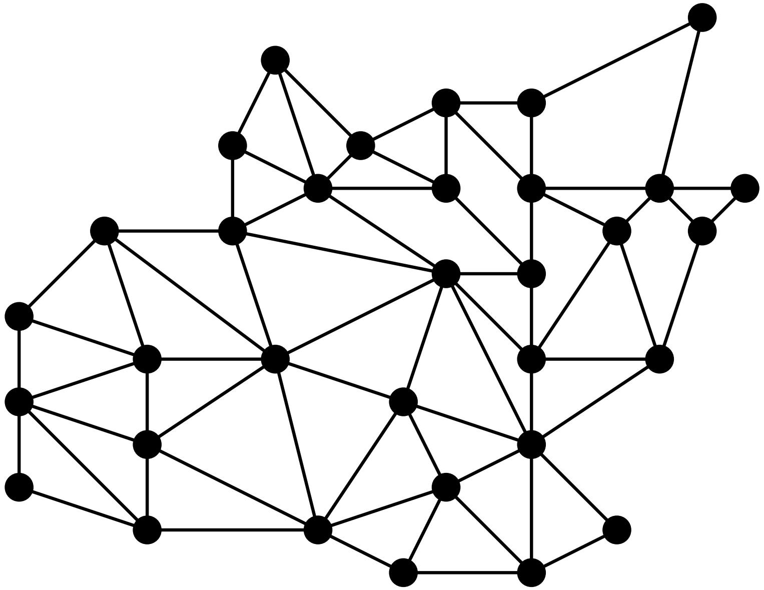
Graphs and Hypergraphs

Graph $G = (V, E)$

vertices → V

edges → E

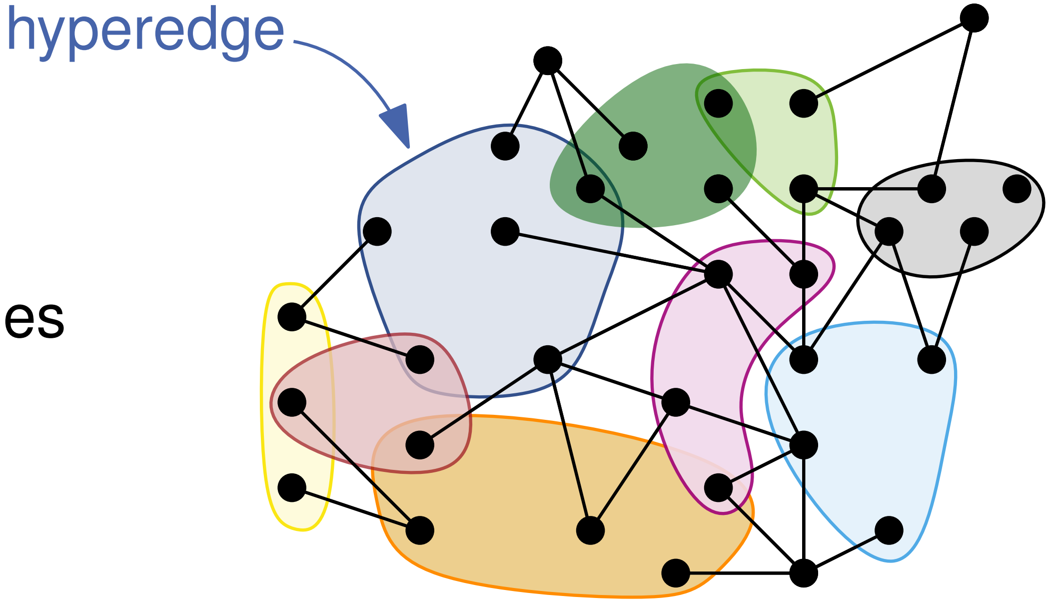
- models **relationships** between **objects**
- dyadic (**$\mathbf{2}$-ary**) relationships



Hypergraph $H = (V, E)$

- generalization of a graph

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



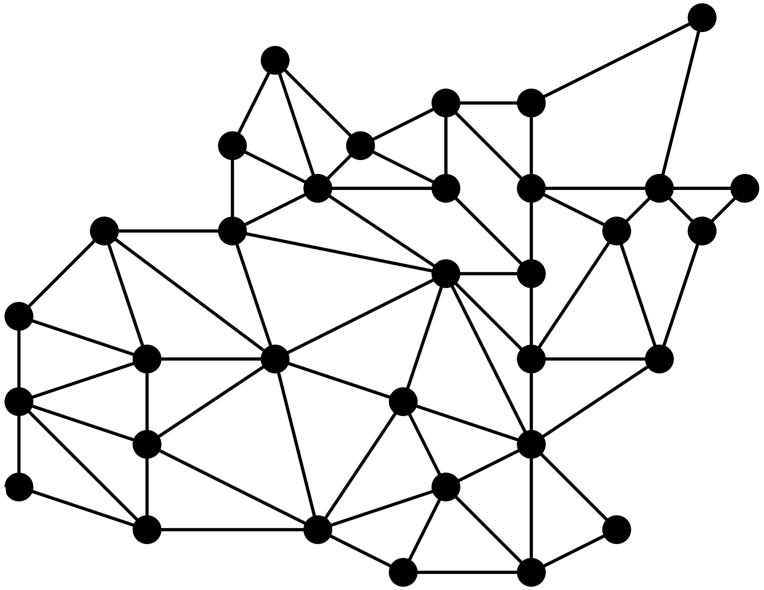
ε-Balanced Graph and Hypergraph Partitioning

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



ε-Balanced Graph and Hypergraph Partitioning

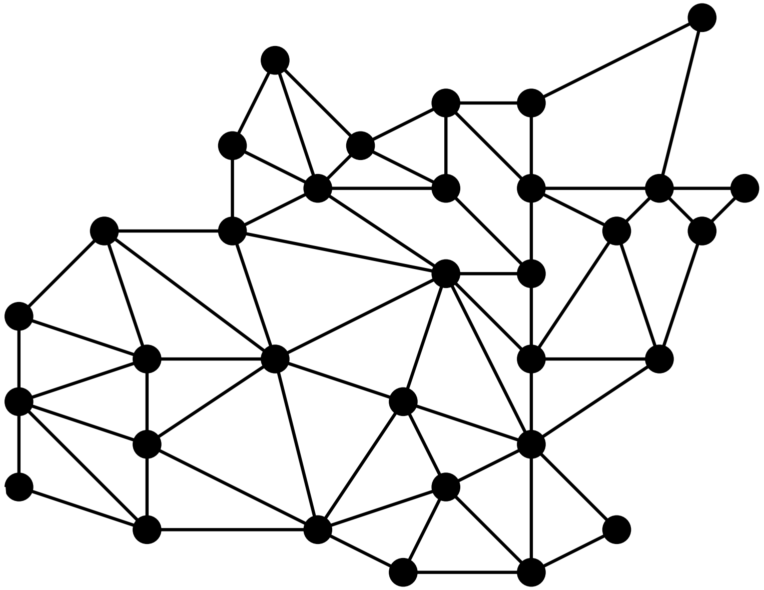
Partition (hyper)graph $G = (V, E, c : V \to \mathrm{R}_{>0}, \omega : E \to \mathrm{R}_{>0})$ into **k** disjoint blocks $V_1, \ldots, V_k$ s.t.

- 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

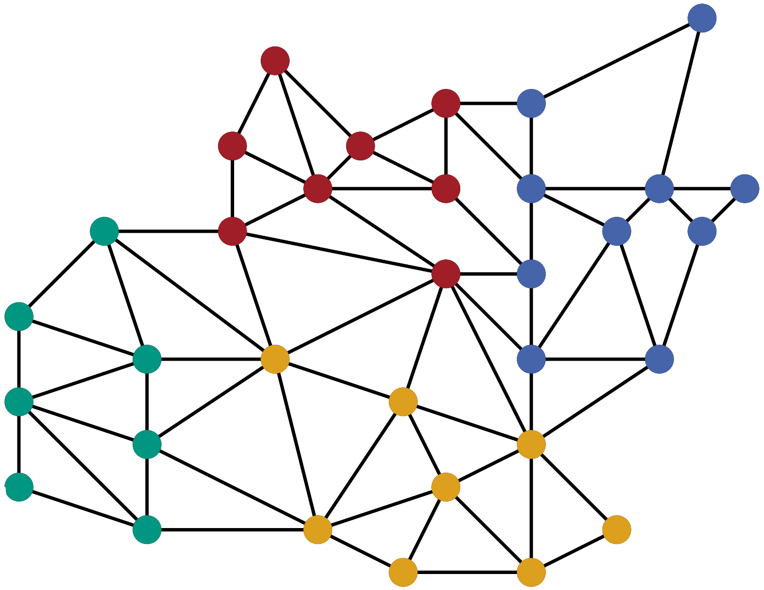
Partition (hyper)graph $G = (V, E, c : V \to \mathrm{R}_{>0}, \omega : E \to \mathrm{R}_{>0})$ into **k** disjoint blocks $V_1, \ldots, V_k$ s.t.

- 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**:

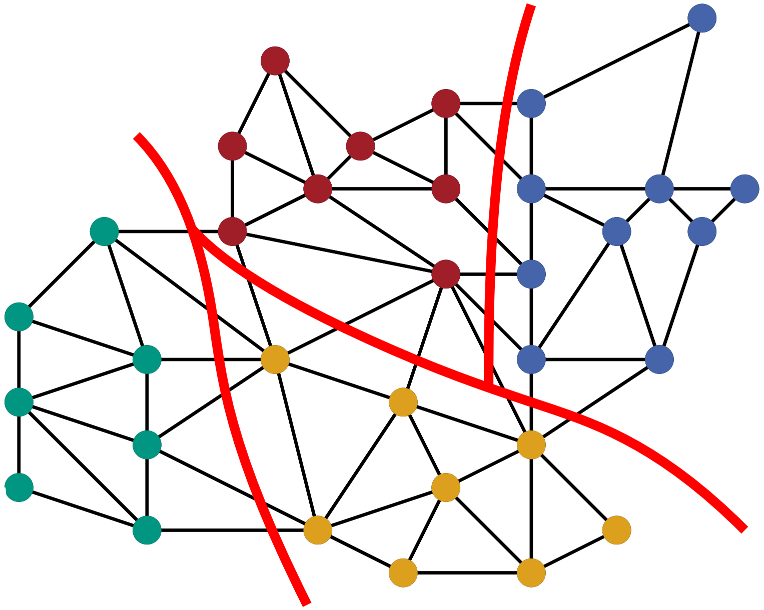
imbalance parameter

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

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

Common Objectives:

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



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**:

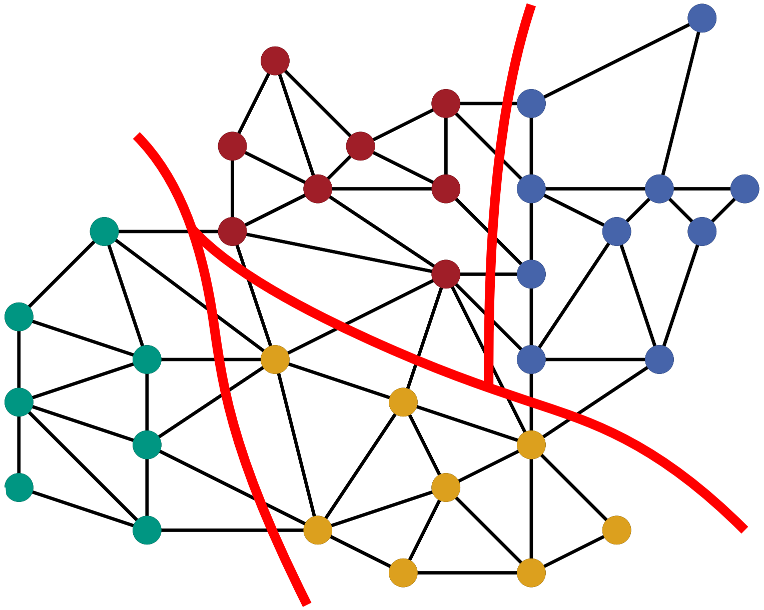
imbalance parameter

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

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

Common Objectives:

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



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 $V_1, \dots, V_k$ s.t.

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

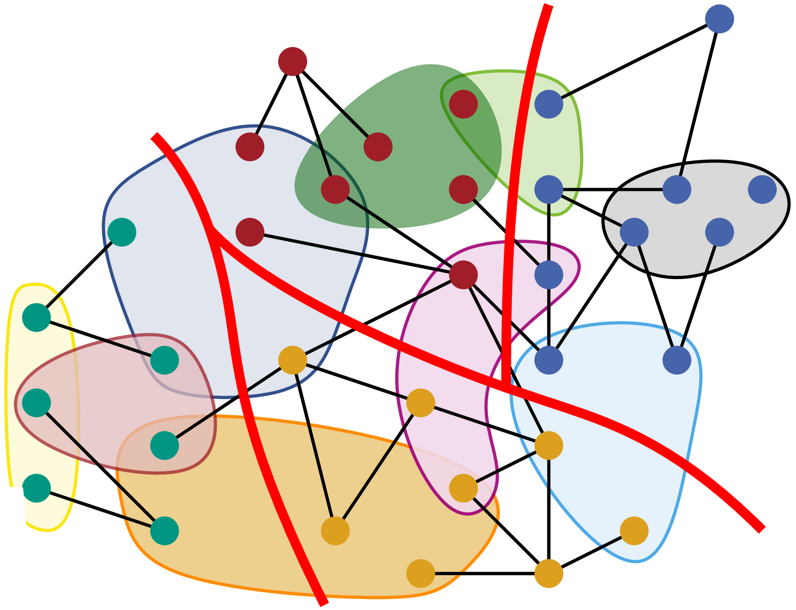
imbalance parameter

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

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

Common Objectives:

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



ε-Balanced Graph and Hypergraph Partitioning

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

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

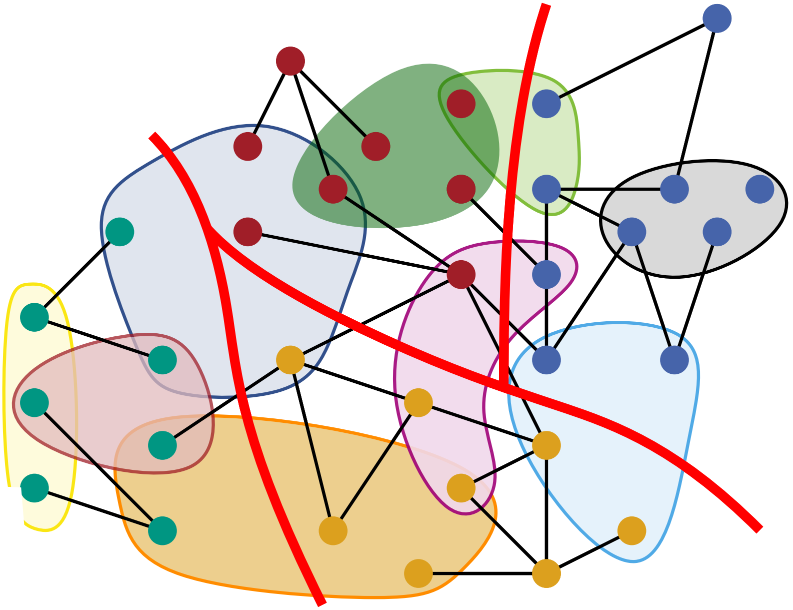
imbalance parameter

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

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

Common Objectives:

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

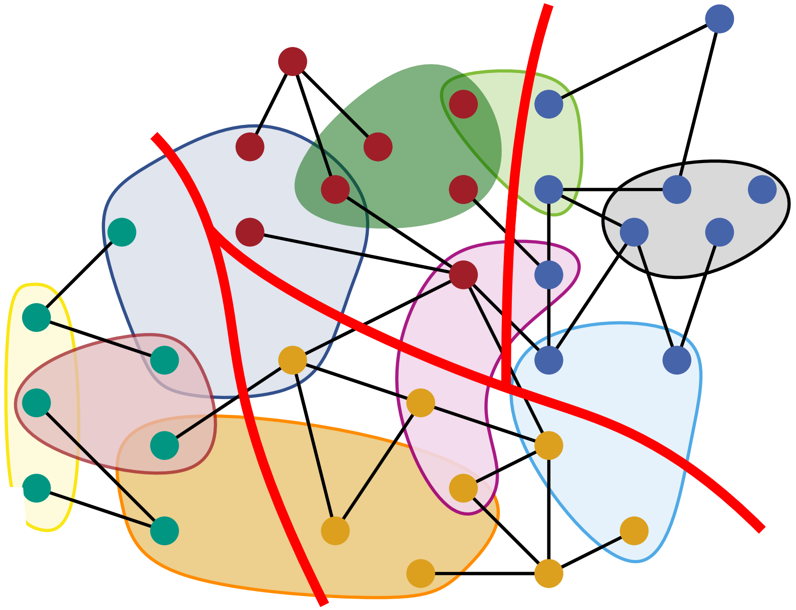
imbalance parameter

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

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

Common Objectives:

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



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**:

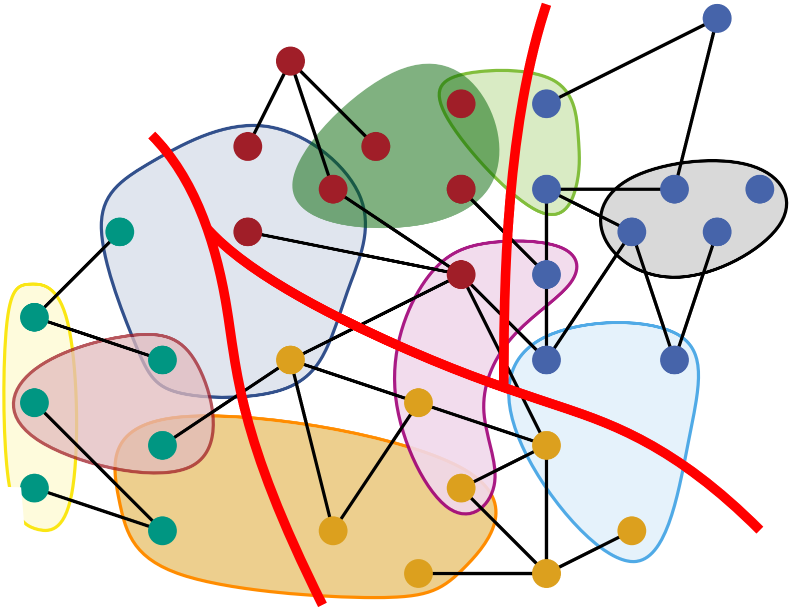
imbalance parameter

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

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

Common Objectives:

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



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

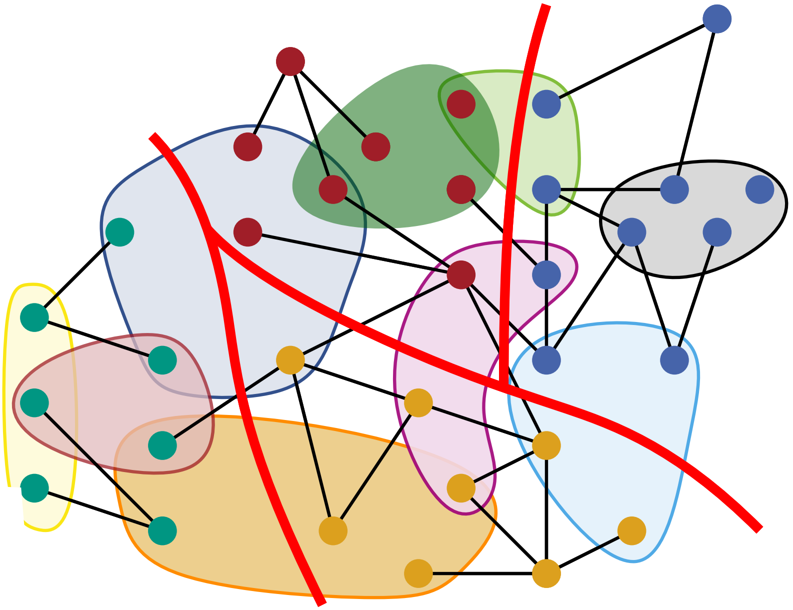
$$c(V_i) \leq (1 + \varepsilon) \left\lceil \frac{c(V)}{k} \right\rceil$$

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

Common Objectives:

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

 # 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

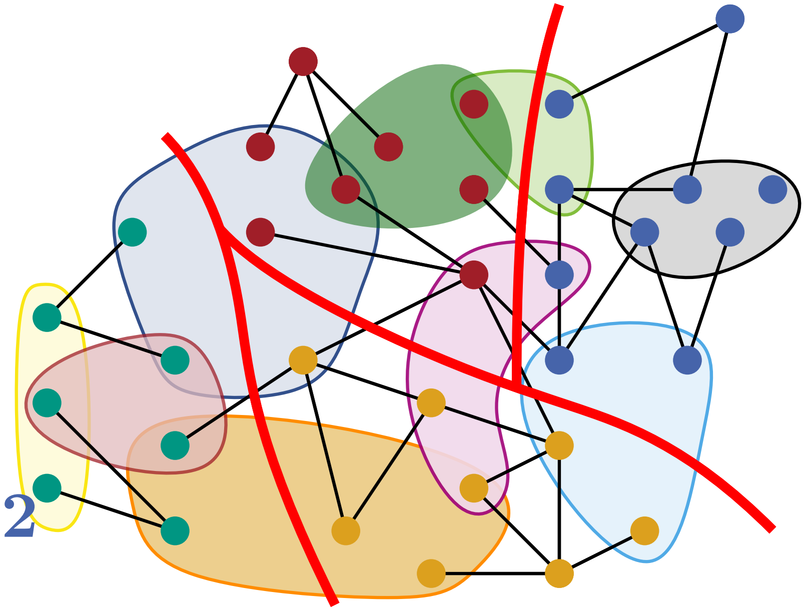
$$c(V_i) \leq (1 + \varepsilon) \left\lceil \frac{c(V)}{k} \right\rceil$$

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

Common Objectives:

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

 # blocks connected by *e*

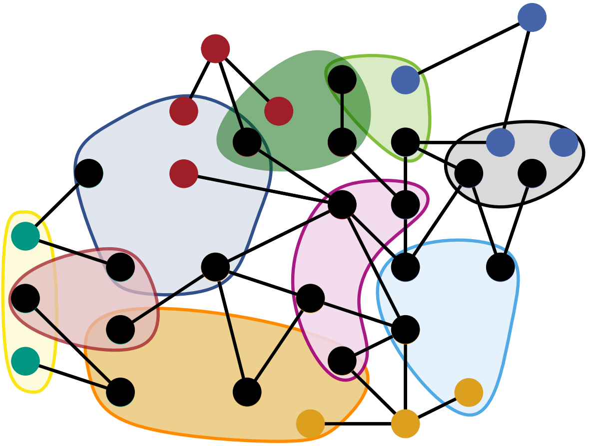


Variants of Standard Hypergraph Partitioning

Possibly relevant/interesting variants:

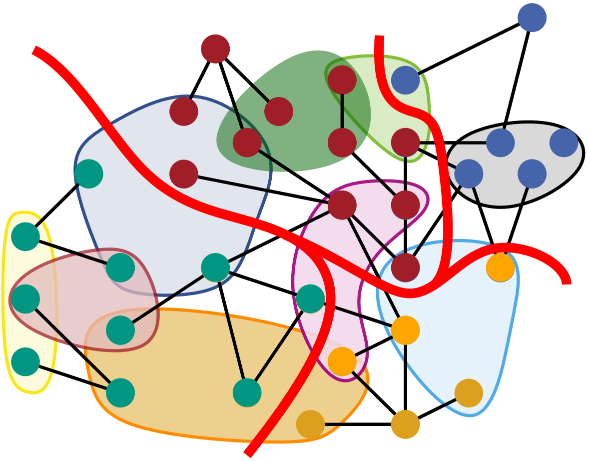
Partitioning with Fixed Vertices:

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



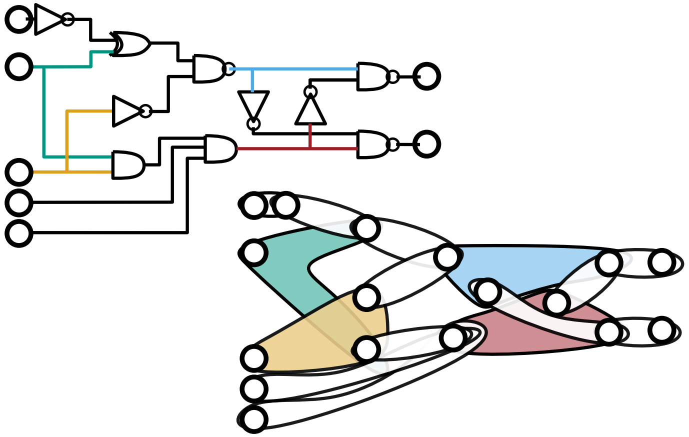
Partitioning with Variable Block Weights:

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



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

Applications



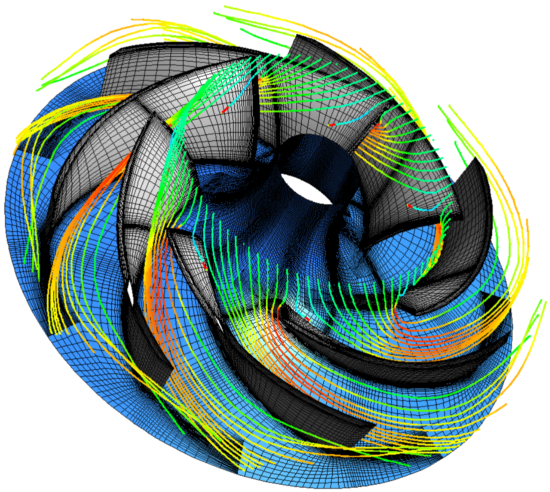
VLSI Design

Warehouse Optimization

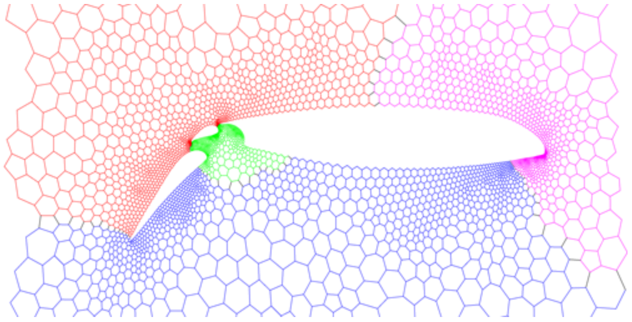
[Martin Grandjean, via Wikimedia Commons]

Complex Networks

Route Planning



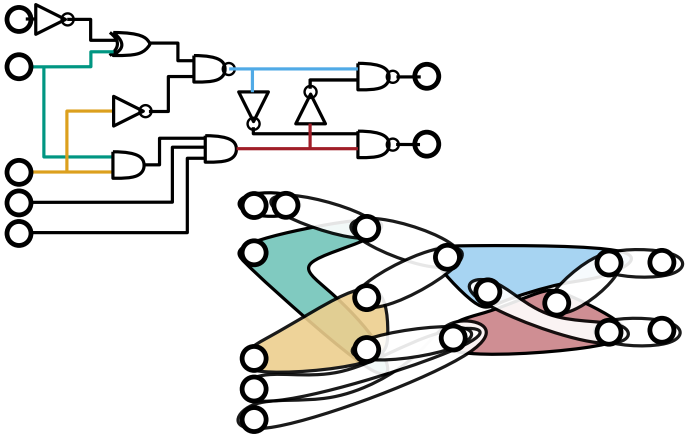
Simulation



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

Scientific Computing

Applications



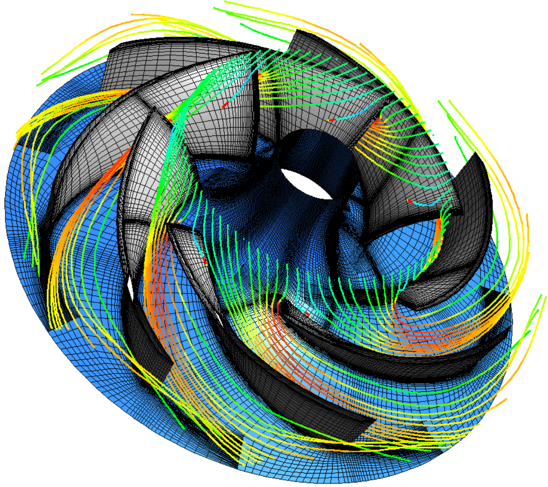
VLSI Design

Warehouse Optimization

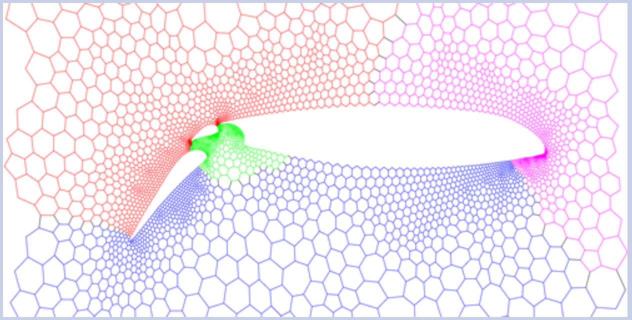
[Martin Grandjean, via Wikimedia Commons]

Complex Networks

Route Planning



Simulation

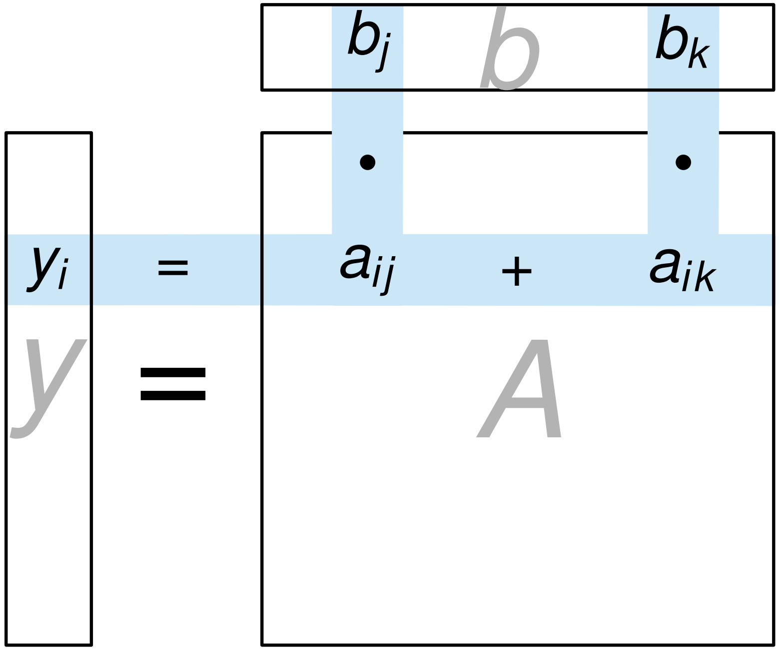


$$\mathbf{R}^{n\times n} \ni Ax = b \in \mathbf{R}^n$$

Scientific Computing

Parallel Sparse-Matrix Vector Product (SpM×V)

$$y = A\,b$$

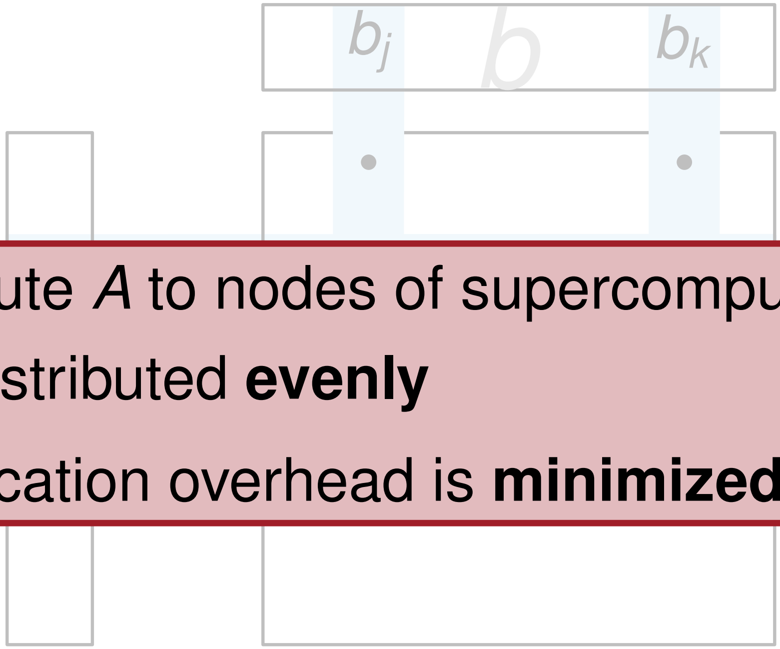


Setting:

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

Parallel Sparse-Matrix Vector Product (SpM×V)

$$y = A\,b$$



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

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

Setting:

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

Naive Approach: Rowwise Decomposition

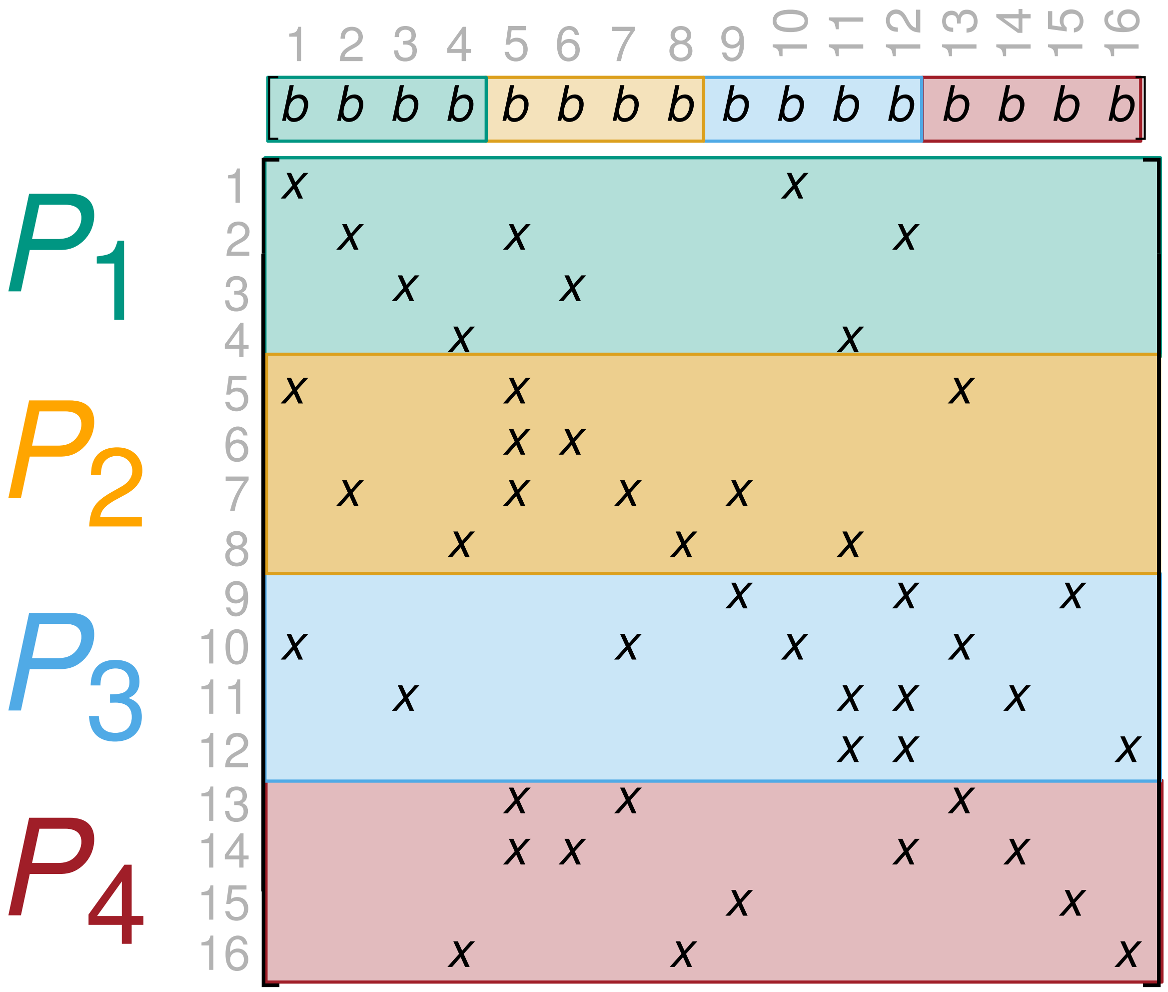
$A \in \mathbb{R}^{16\times 16}$

$$\begin{array}{cccccccccccccccc} 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 & 10 & 11 & 12 & 13 & 14 & 15 & 16 \\ \left[b\right. & b & b & b & b & b & b & b & b & b & b & b & b & b & b & \left.b\right] \end{array}$$

$$\begin{array}{c} 1\\2\\3\\4\\5\\6\\7\\8\\9\\10\\11\\12\\13\\14\\15\\16 \end{array} \begin{bmatrix}
x & & & & & & & & & x & & & & & & \\
 & x & & & x & & & & & & & x & & & & \\
 & & x & & & x & & & & & & & & & & \\
 & & & x & & & & & & & x & & & & & \\
x & & & & x & & & & & & & & x & & & \\
 & & & & x & x & & & & & & & & & & \\
 & x & & & x & & x & & x & & & & & & & \\
 & & & x & & & & x & & & x & & & & & \\
 & & & & & & & & x & & & x & & & x & \\
x & & & & & & x & & & x & & & x & & & \\
 & & x & & & & & & & & x & x & & x & & \\
 & & & & & & & & & & x & x & & & & x \\
 & & & & x & & x & & & & & & x & & & \\
 & & & & x & x & & & & & & x & & x & & \\
 & & & & & & & & x & & & & & & x & \\
 & & & x & & & & x & & & & & & & & x
\end{bmatrix}$$

Naive Approach: Rowwise Decomposition

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

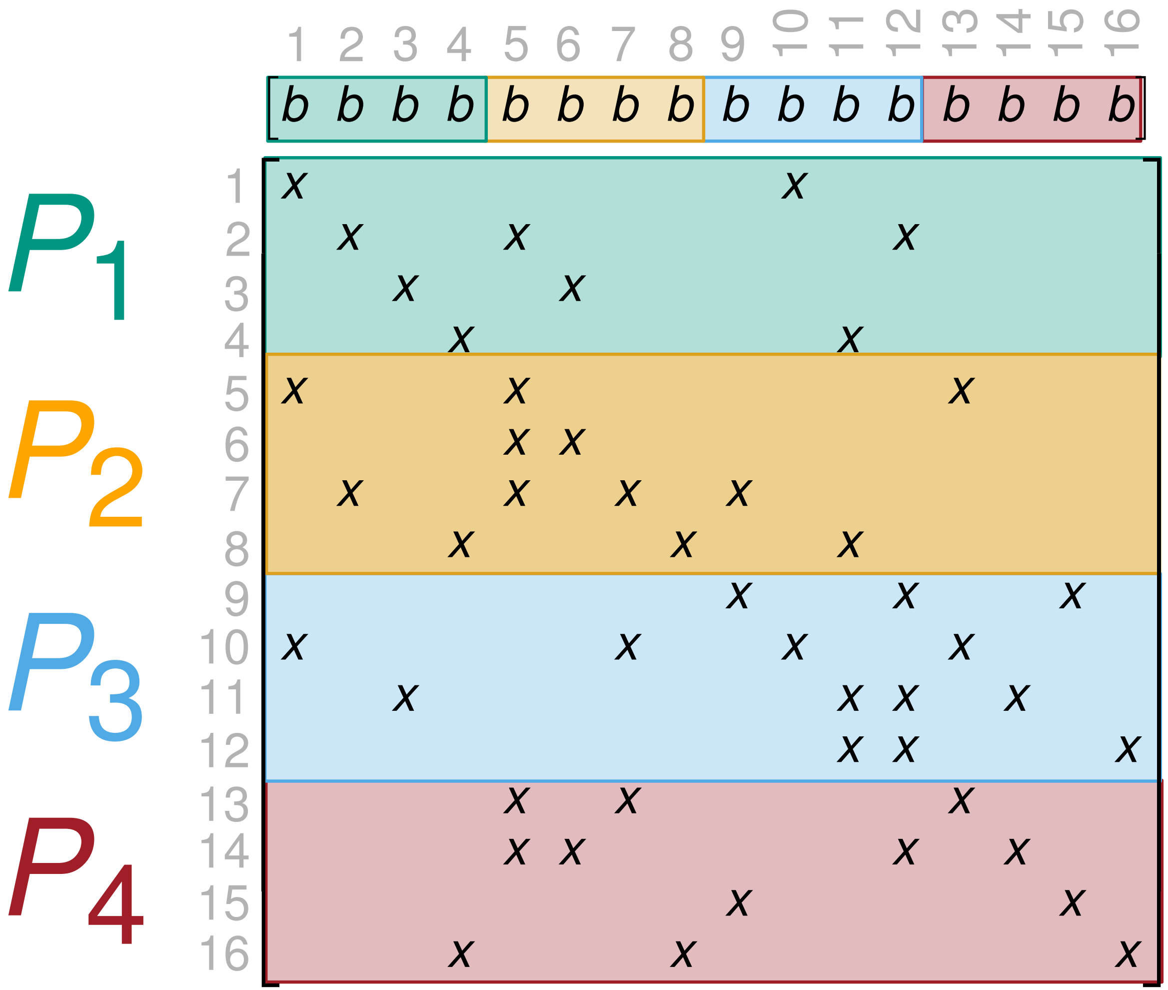


$$\begin{array}{cccccccccccccccc}
1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 & 10 & 11 & 12 & 13 & 14 & 15 & 16 \\
b & b & b & b & b & b & b & b & b & b & b & b & b & b & b & b
\end{array}$$

$$\begin{array}{c|cccccccccccccccc}
 & 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 & 10 & 11 & 12 & 13 & 14 & 15 & 16 \\
\hline
1 & x & & & & & & & & & x & & & & & & \\
2 & & x & & & x & & & & & & & x & & & & \\
3 & & & x & & & x & & & & & & & & & & \\
4 & & & & x & & & & & & & x & & & & & \\
5 & x & & & & x & & & & & & & & x & & & \\
6 & & & & & x & x & & & & & & & & & & \\
7 & & x & & & x & & x & & x & & & & & & & \\
8 & & & & x & & & & x & & & x & & & & & \\
9 & & & & & & & & & x & & & x & & & x & \\
10 & x & & & & & & x & & & x & & & x & & & \\
11 & & & x & & & & & & & & x & x & & x & & \\
12 & & & & & & & & & & & x & x & & & & x \\
13 & & & & & x & & x & & & & & & x & & & \\
14 & & & & & x & x & & & & & & x & & x & & \\
15 & & & & & & & & & x & & & & & & x & \\
16 & & & & x & & & & x & & & & & & & & x
\end{array}$$

Naive Approach: Rowwise Decomposition

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



$$\begin{array}{c|cccccccccccccccc}
 & 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 & 10 & 11 & 12 & 13 & 14 & 15 & 16 \\
 & b & b & b & b & b & b & b & b & b & b & b & b & b & b & b & b \\
\hline
1 & x & & & & & & & & & x & & & & & & \\
2 & & x & & & x & & & & & & & x & & & & \\
3 & & & x & & & x & & & & & & & & & & \\
4 & & & & x & & & & & & & x & & & & & \\
\hline
5 & x & & & & x & & & & & & & & x & & & \\
6 & & & & & x & x & & & & & & & & & & \\
7 & & x & & & x & & x & & x & & & & & & & \\
8 & & & & x & & & & x & & & x & & & & & \\
\hline
9 & & & & & & & & & x & & & x & & & x & \\
10 & x & & & & & & x & & & x & & & x & & & \\
11 & & & x & & & & & & & & x & x & & x & & \\
12 & & & & & & & & & & & x & x & & & & x \\
\hline
13 & & & & & x & & x & & & & & & x & & & \\
14 & & & & & x & x & & & & & & x & & x & & \\
15 & & & & & & & & & x & & & & & & x & \\
16 & & & & x & & & & x & & & & & & & & x
\end{array}$$

Load Balancing?

$\Rightarrow 9$

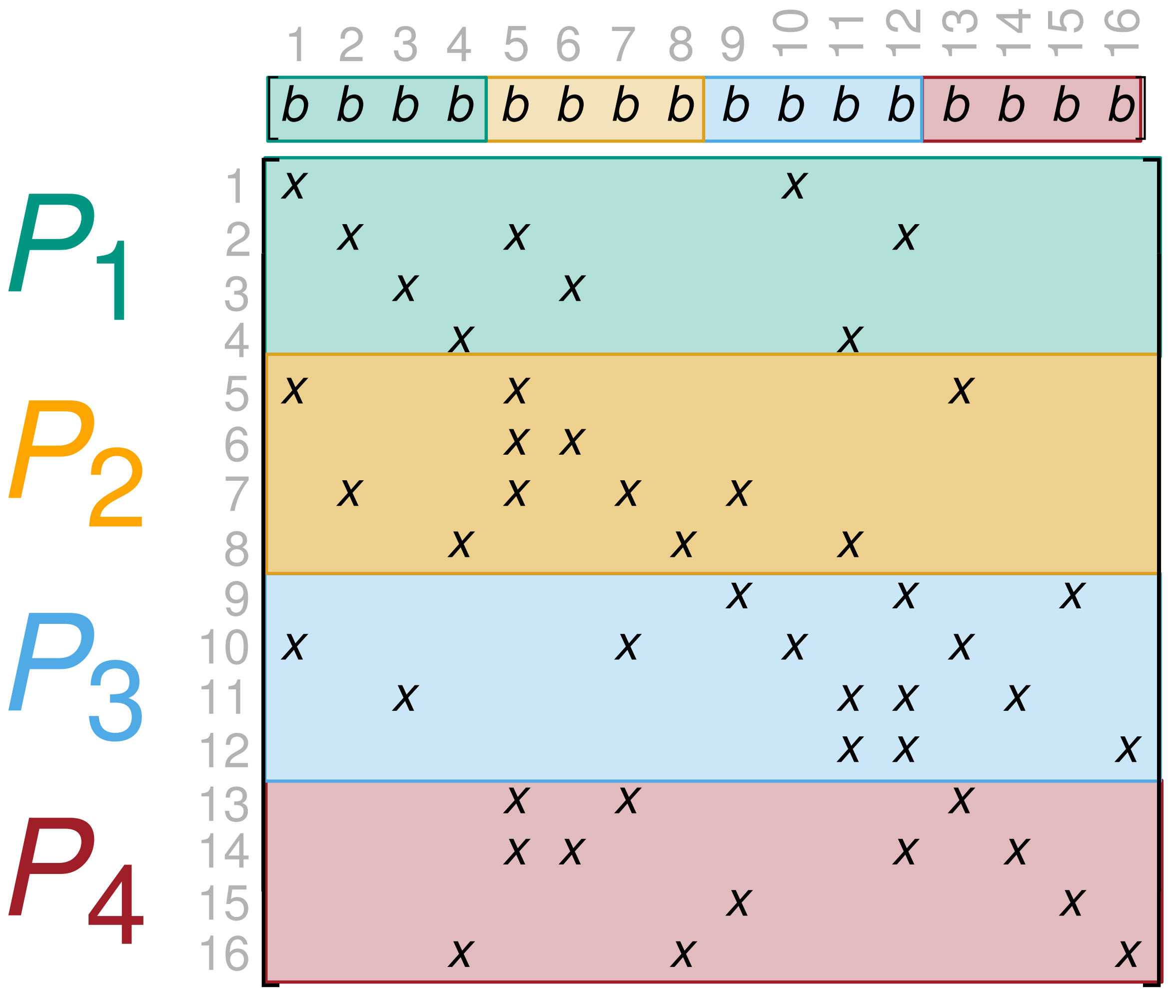
$\Rightarrow 12$

$\Rightarrow 14$

$\Rightarrow 12$

Naive Approach: Rowwise Decomposition

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



$$\begin{array}{cccccccccccccccc} 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 & 10 & 11 & 12 & 13 & 14 & 15 & 16 \\ b & b & b & b & b & b & b & b & b & b & b & b & b & b & b & b \end{array}$$

$$\begin{array}{c|cccccccccccccccc} & 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 & 10 & 11 & 12 & 13 & 14 & 15 & 16 \\ \hline 1 & x & & & & & & & & & x & & & & & & \\ 2 & & x & & & x & & & & & & & x & & & & \\ 3 & & & x & & & x & & & & & & & & & & \\ 4 & & & & x & & & & & & & x & & & & & \\ 5 & x & & & & x & & & & & & & & x & & & \\ 6 & & & & & x & x & & & & & & & & & & \\ 7 & & x & & & x & & x & & x & & & & & & & \\ 8 & & & & x & & & & x & & & x & & & & & \\ 9 & & & & & & & & & x & & & x & & & x & \\ 10 & x & & & & & & x & & & x & & & x & & & \\ 11 & & & x & & & & & & & & x & x & & x & & \\ 12 & & & & & & & & & & & x & x & & & & x \\ 13 & & & & & x & & x & & & & & & x & & & \\ 14 & & & & & x & x & & & & & & x & & x & & \\ 15 & & & & & & & & & x & & & & & & x & \\ 16 & & & & x & & & & x & & & & & & & & x \end{array}$$

Load Balancing?

$\Rightarrow 9$

$\Rightarrow 12$

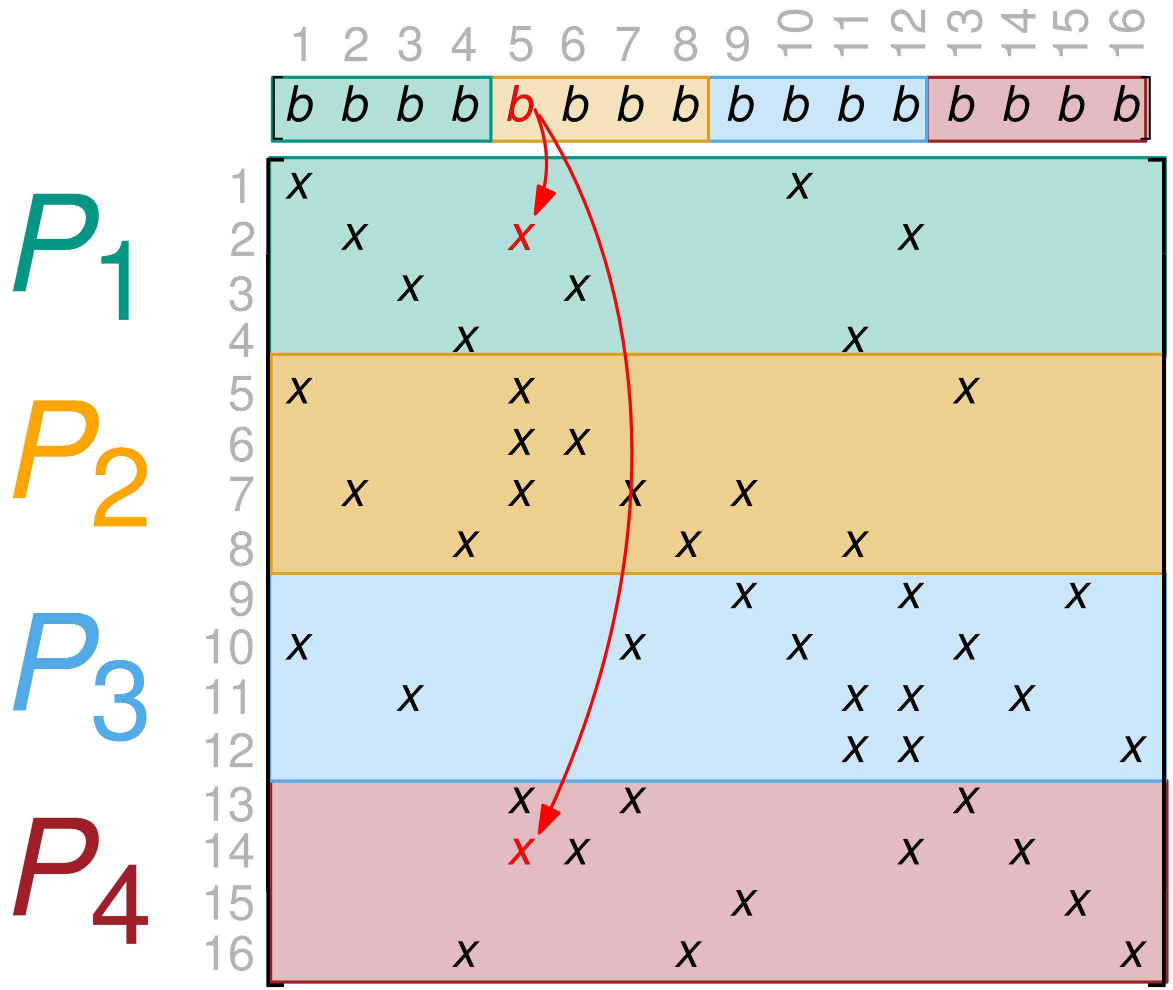
$\Rightarrow 14$

$\Rightarrow 12$

Commuication Volume?

Naive Approach: Rowwise Decomposition

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



$$\begin{array}{c|cccccccccccccccc}
 & 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 & 10 & 11 & 12 & 13 & 14 & 15 & 16 \\
 & b & b & b & b & b & b & b & b & b & b & b & b & b & b & b & b \\
\hline
1 & x & & & & & & & & & x & & & & & & \\
2 & & x & & & x & & & & & & & x & & & & \\
3 & & & x & & & x & & & & & & & & & & \\
4 & & & & x & & & & & & & x & & & & & \\
\hline
5 & x & & & & x & & & & & & & & x & & & \\
6 & & & & & x & x & & & & & & & & & & \\
7 & & x & & & x & & x & & x & & & & & & & \\
8 & & & & x & & & & x & & & x & & & & & \\
\hline
9 & & & & & & & & & x & & & x & & & x & \\
10 & x & & & & & & x & & & x & & & x & & & \\
11 & & & x & & & & & & & & x & x & & x & & \\
12 & & & & & & & & & & & x & x & & & & x \\
\hline
13 & & & & & x & & x & & & & & & x & & & \\
14 & & & & & x & x & & & & & & x & & x & & \\
15 & & & & & & & & & x & & & & & & x & \\
16 & & & & x & & & & x & & & & & & & & x
\end{array}$$

Load Balancing?

$\Rightarrow 9$

$\Rightarrow 12$

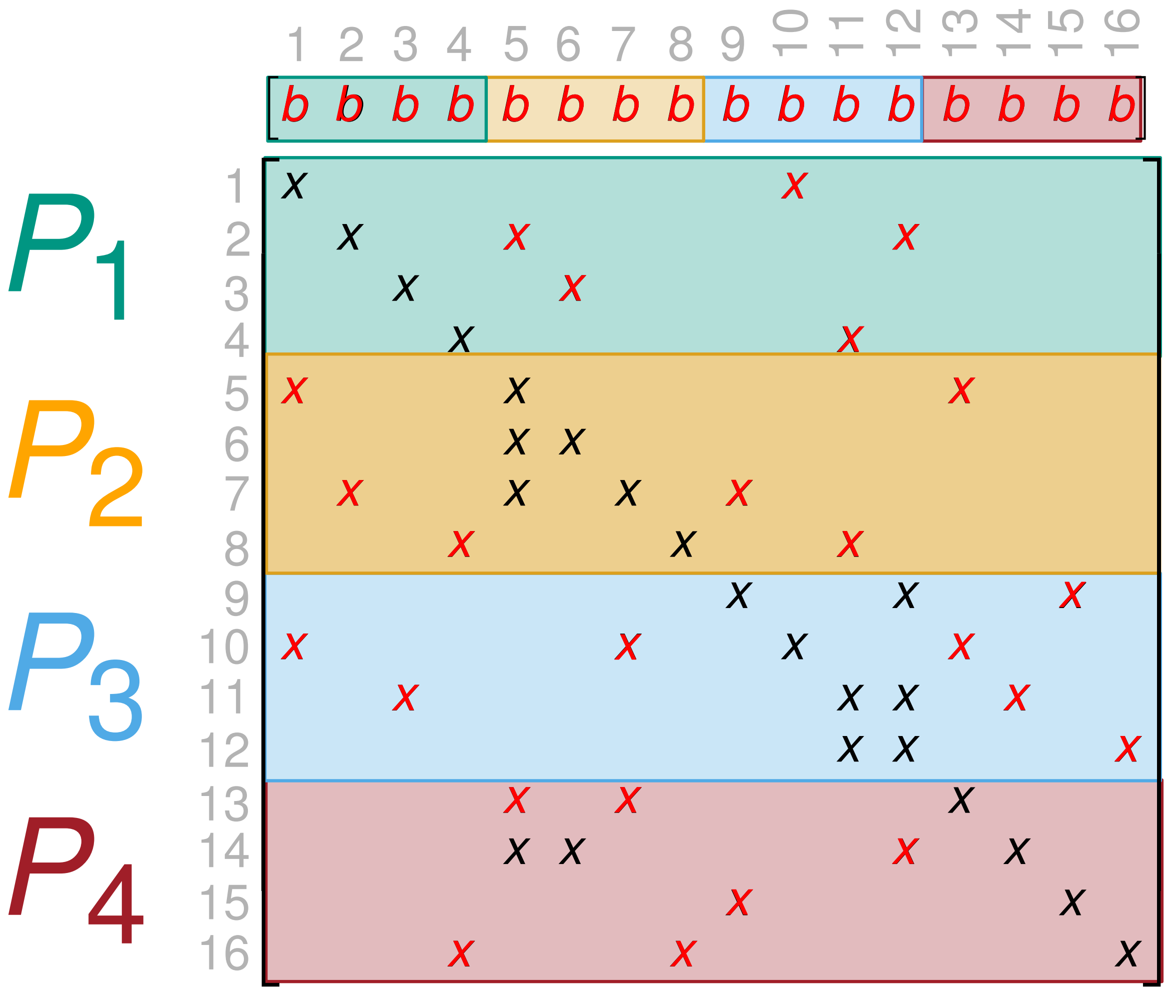
$\Rightarrow 14$

$\Rightarrow 12$

Commuication Volume?

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					
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

Load Balancing?

⇒ 9

⇒ 12

⇒ 14

⇒ 12

Commuication Volume? ⇒ 24 entries!

Naive Approach: Rowwise Decomposition

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

Load Balancing?

$\Rightarrow$ 9

$\Rightarrow$ 12

$\Rightarrow$ 14

$\Rightarrow$ 12

Can we do better?

Commuication Volume? $\Rightarrow$ 24 entries!

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:

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

$$\begin{array}{cccccccccccccccc}
1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 & 10 & 11 & 12 & 13 & 14 & 15 & 16 \\
\end{array}$$

$$\left[\begin{array}{cccccccccccccccc}
b & b & b & b & b & b & b & b & b & b & b & b & b & b & b & b
\end{array}\right]$$

$$\begin{array}{c}
1\\2\\3\\4\\5\\6\\7\\8\\9\\10\\11\\12\\13\\14\\15\\16
\end{array}
\left[\begin{array}{cccccccccccccccc}
x & & & & & & & & & x & & & & & & \\
 & x & & & x & & & & & & & x & & & & \\
 & & x & & & x & & & & & & & & & & \\
 & & & x & & & & & & & x & & & & & \\
x & & & & x & & & & & & & & x & & & \\
 & & & & x & x & & & & & & & & & & \\
 & x & & & x & & x & & x & & & & & & & \\
 & & & x & & & & x & & & x & & & & & \\
 & & & & & & & & x & & & x & & & x & \\
x & & & & & & x & & & x & & & x & & & \\
 & & x & & & & & & & & x & x & & x & & \\
 & & & & & & & & & & x & x & & & & x \\
 & & & & x & & x & & & & & & x & & & \\
 & & & & x & x & & & & & & x & & x & & \\
 & & & & & & & & x & & & & & & x & \\
 & & & x & & & & x & & & & & & & & x
\end{array}\right]$$

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:

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

$$
\begin{array}{r|cccccccccccccccc|}
 & 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 & 10 & 11 & 12 & 13 & 14 & 15 & 16 \\
 & b & b & b & b & b & b & b & b & b & b & b & b & b & b & b & b \\
1 & x & & & & & & & & & x & & & & & & \\
2 & & x & & & x & & & & & & & x & & & & \\
3 & & & x & & & x & & & & & & & & & & \\
4 & & & & x & & & & & & & x & & & & & \\
5 & x & & & & x & & & & & & & & x & & & \\
6 & & & & & x & x & & & & & & & & & & \\
7 & & x & & & x & & x & & x & & & & & & & \\
8 & & & & x & & & & x & & & x & & & & & \\
v_9 \; 9 & & & & & & & & & x & & & x & & & x & \\
10 & x & & & & & & x & & & x & & & x & & & \\
11 & & & x & & & & & & & & x & x & & x & & \\
12 & & & & & & & & & & & x & x & & & & x \\
13 & & & & & x & & x & & & & & & x & & & \\
14 & & & & & x & x & & & & & & x & & x & & \\
15 & & & & & & & & & x & & & & & & x & \\
16 & & & & x & & & & x & & & & & & & & x
\end{array}
$$

$v_i \in V_R$ **:**

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

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:

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

$$
\begin{array}{r|cccccccccccccccc|}
 & 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 & 10 & 11 & 12 & 13 & 14 & 15 & 16 \\
 & b & b & b & b & b & b & b & b & b & b & b & b & b & b & b & b \\
\hline
1 & x & & & & & & & & & x & & & & & & \\
2 & & x & & & x & & & & & & & x & & & & \\
3 & & & x & & & x & & & & & & & & & & \\
4 & & & & x & & & & & & & x & & & & & \\
5 & x & & & & x & & & & & & & & x & & & \\
6 & & & & & x & x & & & & & & & & & & \\
7 & & x & & & x & & x & & x & & & & & & & \\
8 & & & & x & & & & x & & & x & & & & & \\
v_9 \; 9 & & & & & & & & & x & & & x & & & x & \\
10 & x & & & & & & x & & & x & & & x & & & \\
11 & & & x & & & & & & & & x & x & & x & & \\
12 & & & & & & & & & & & x & x & & & & x \\
13 & & & & & x & & x & & & & & & x & & & \\
14 & & & & & x & x & & & & & & x & & x & & \\
15 & & & & & & & & & x & & & & & & x & \\
16 & & & & x & & & & x & & & & & & & & x \\
\hline
 & & & & & e_5 & & & & & & & & & & &
\end{array}
$$

$v_i \in V_R$:

- 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

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:

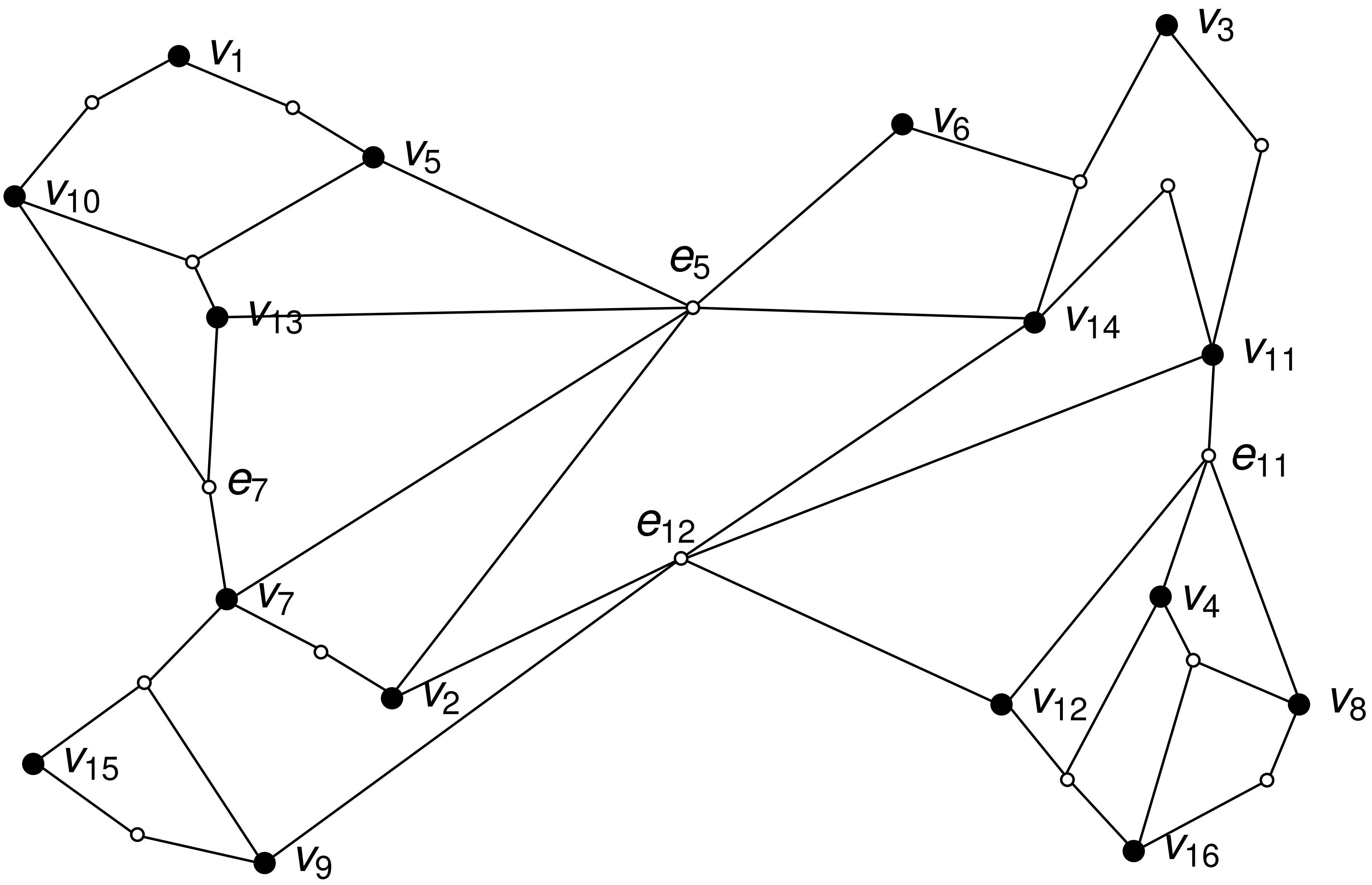
  $\Rightarrow E_C = \{e_1, e_2, \ldots, e_{16}\}$

$$\begin{array}{c|cccccccccccccccc|}
 & 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 & 10 & 11 & 12 & 13 & 14 & 15 & 16 \\
 & b & b & b & b & b & b & b & b & b & b & b & b & b & b & b & b \\
1 & x & & & & & & & & & x & & & & & & \\
2 & & x & & & x & & & & & & & x & & & & \\
3 & & & x & & & x & & & & & & & & & & \\
4 & & & & x & & & & & & & x & & & & & \\
5 & x & & & & x & & & & & & & & x & & & \\
12 & & & & & & & & & & & x & x & & & & x \\
13 & & & & & x & & x & & & & & & x & & & \\
14 & & & & & x & x & & & & & & x & & x & & \\
15 & & & & & & & & & x & & & & & & x & \\
16 & & & & x & & & & x & & & & & & & & x
\end{array}$$

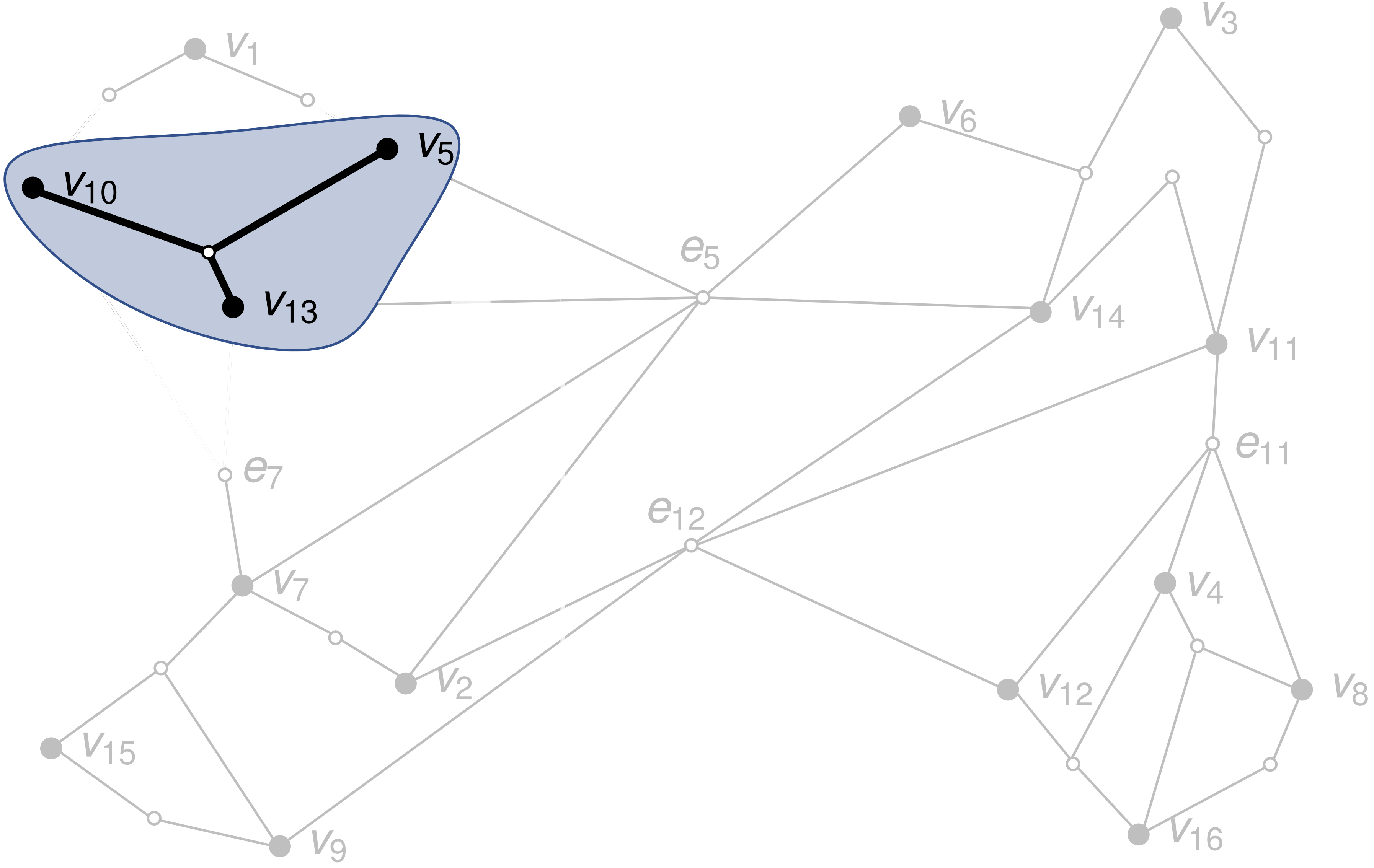
Solution: ε-balanced partition of H

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

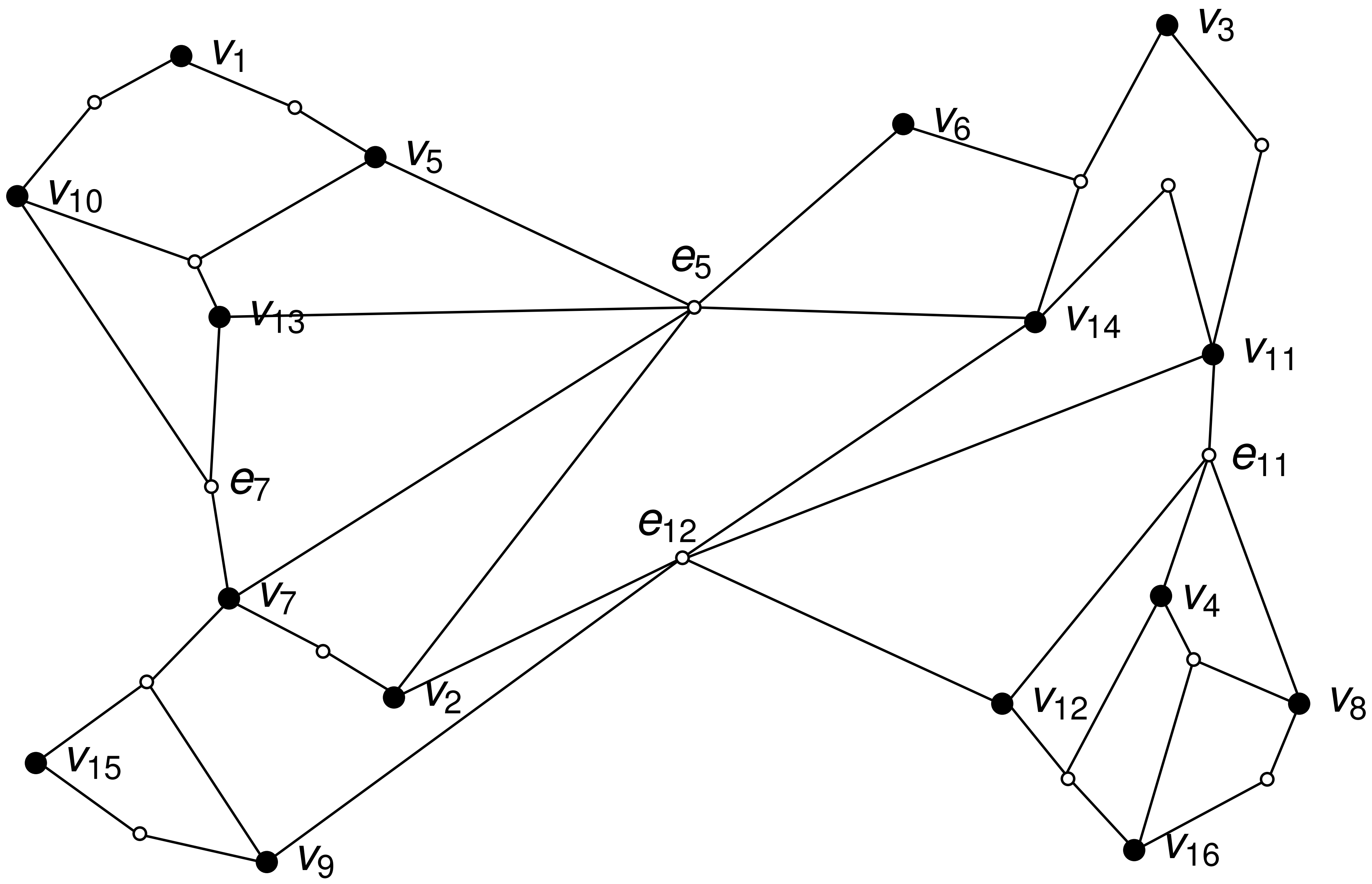
From SpM×V to Hypergraph Partitioning



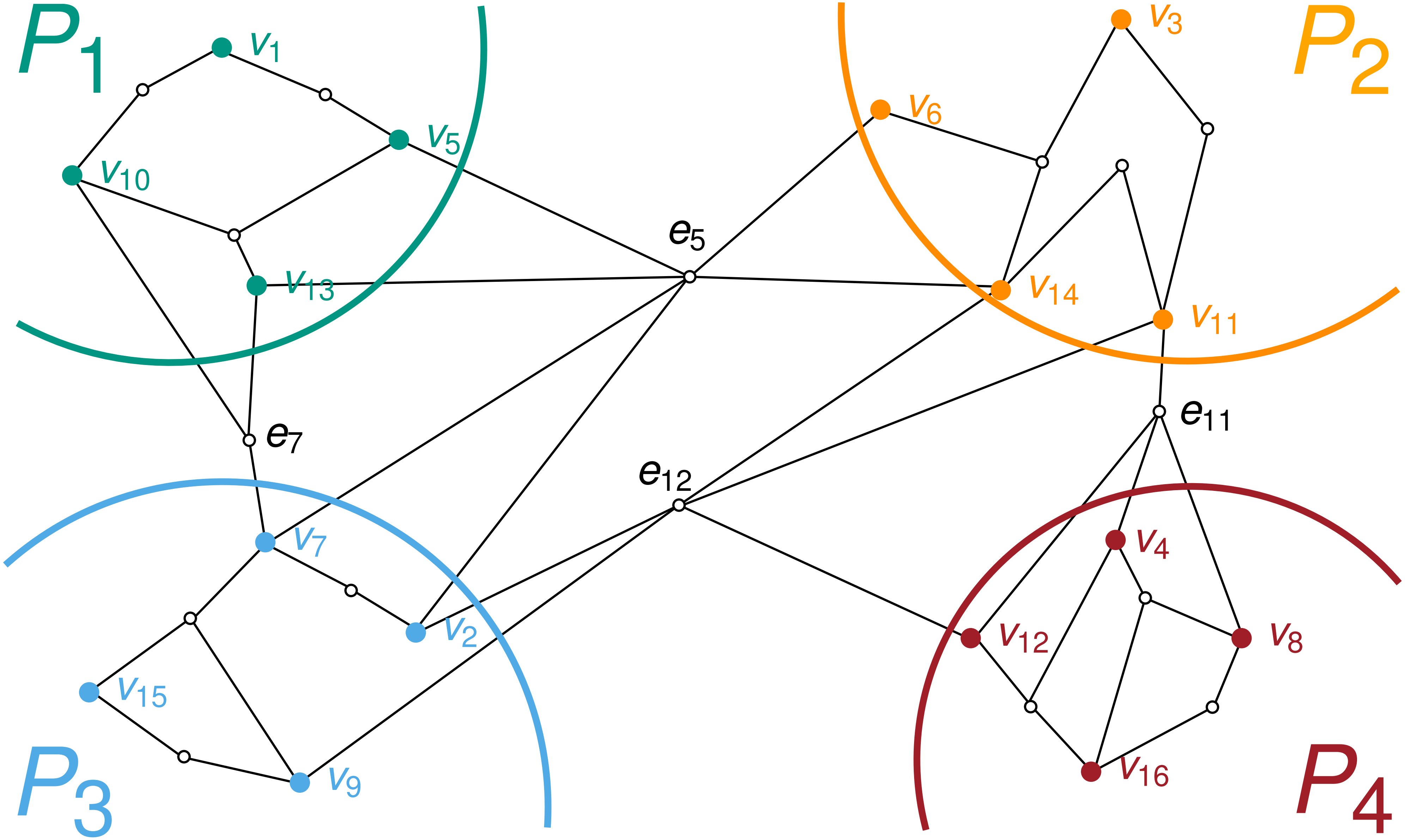
From SpM×V to Hypergraph Partitioning



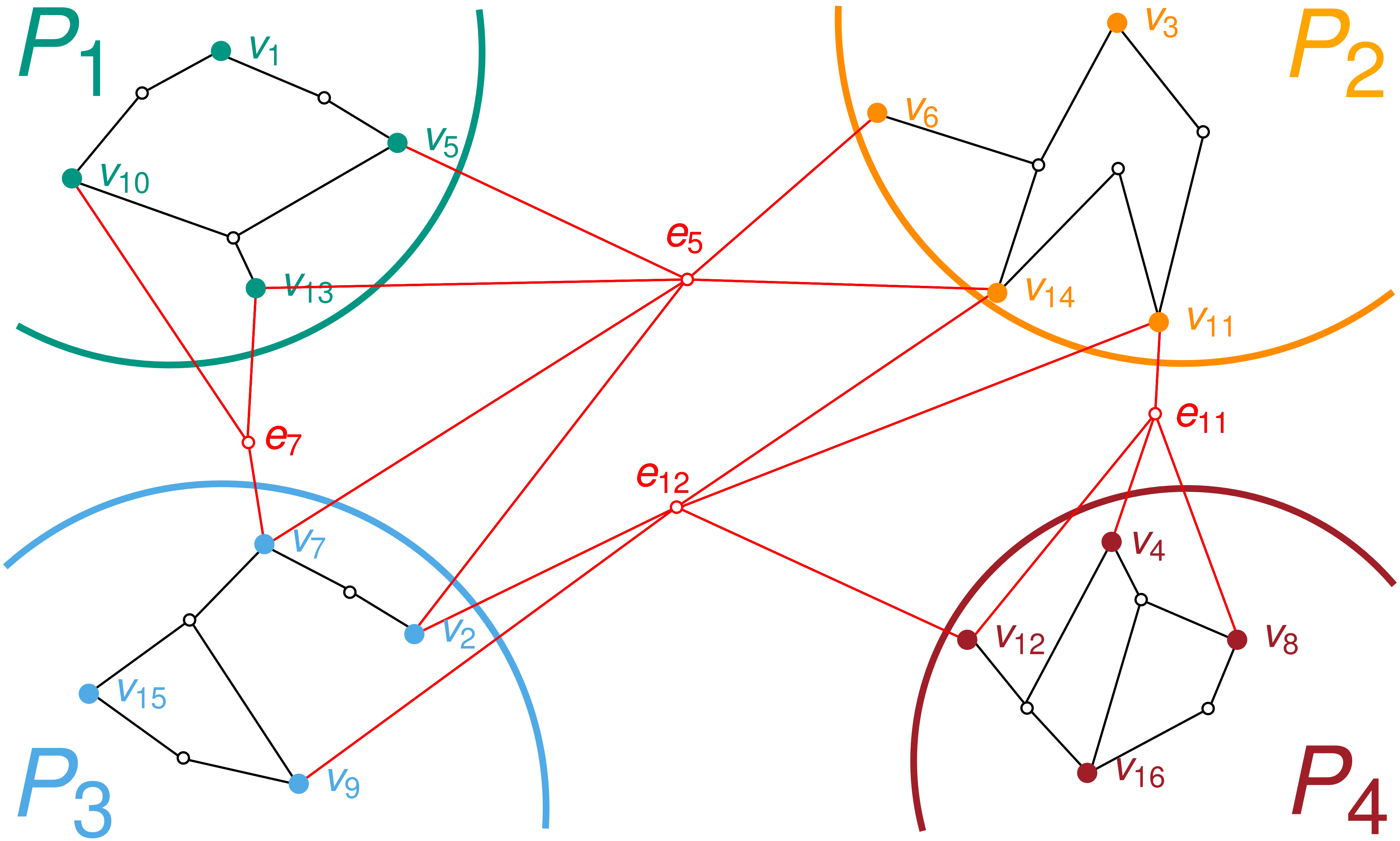
From SpM$\times$V to Hypergraph Partitioning



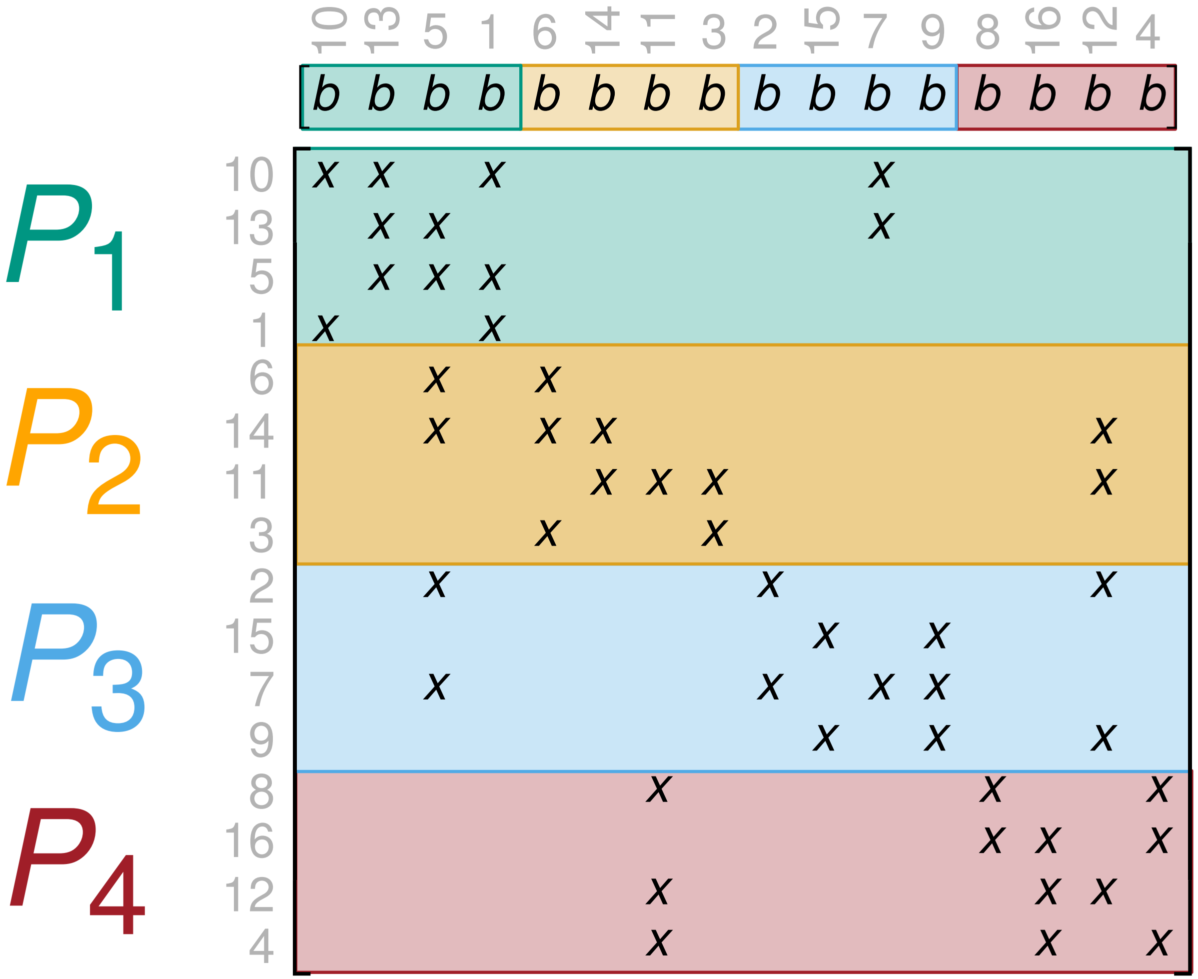
From SpM×V to Hypergraph Partitioning



From SpM × V to Hypergraph Partitioning

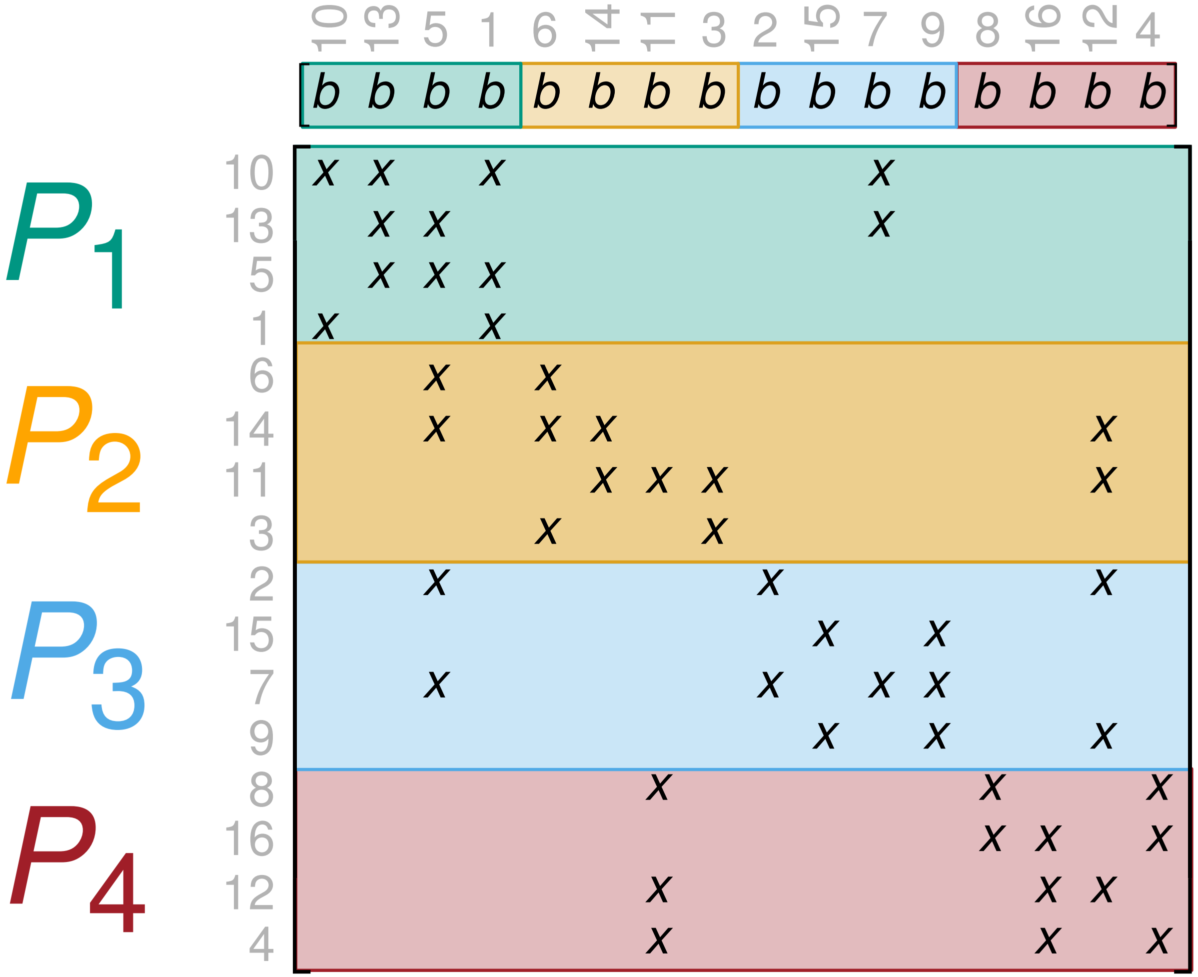


From Hypergraph Partitioning to SpM×V



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

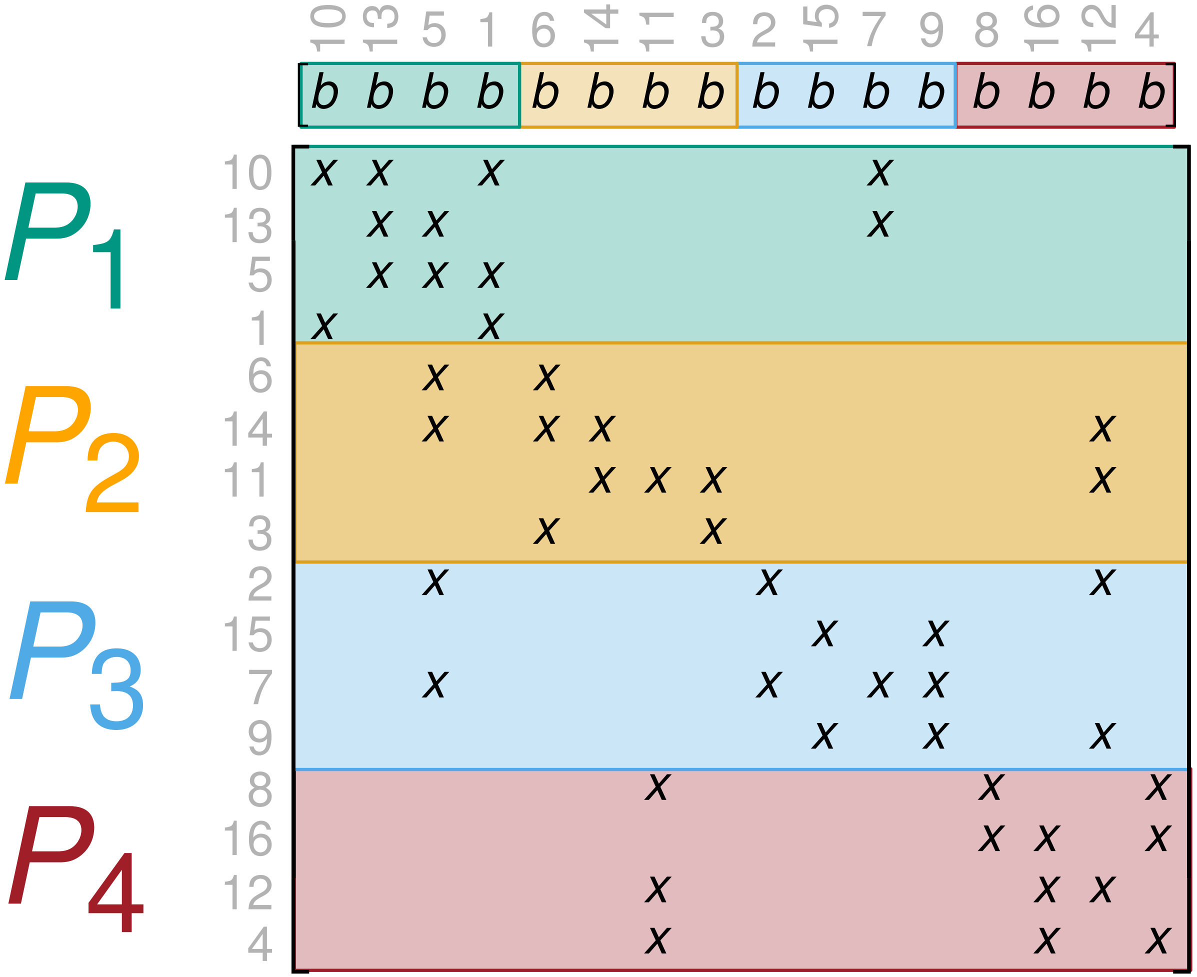
From Hypergraph Partitioning to SpM×V



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

Load Balancing?

From Hypergraph Partitioning to SpM×V



	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	
3					x			x								
2			x						x						x	
15										x		x				
7			x						x		x	x				
9										x		x			x	
8							x						x			x
16													x	x		x
12							x							x	x	
4							x							x		x

Load Balancing?

⇒ 12

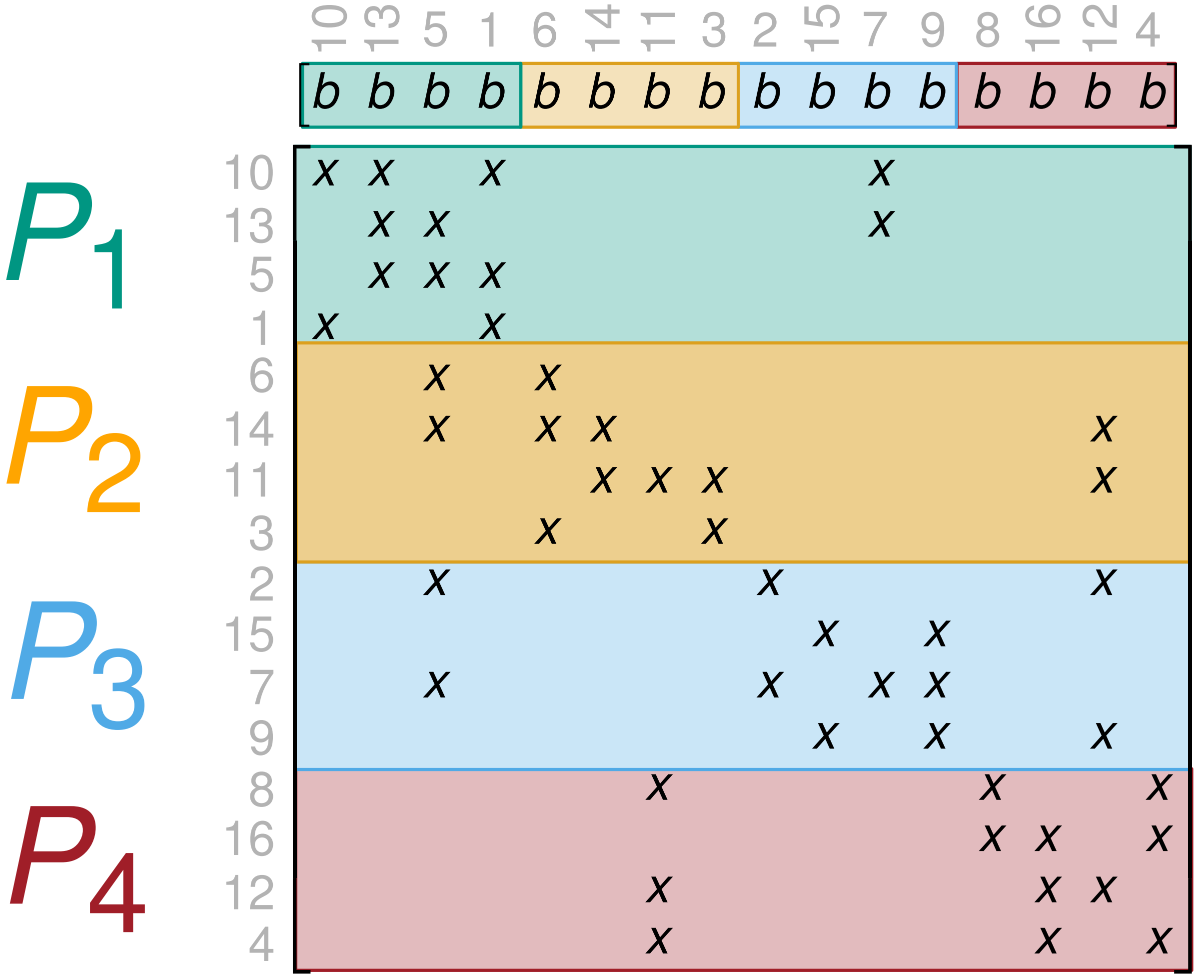
⇒ 12

⇒ 12

⇒ 12

From Hypergraph Partitioning to SpM×V

Where are the cut-hyperedges?



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

Load Balancing?

⇒ 12

⇒ 12

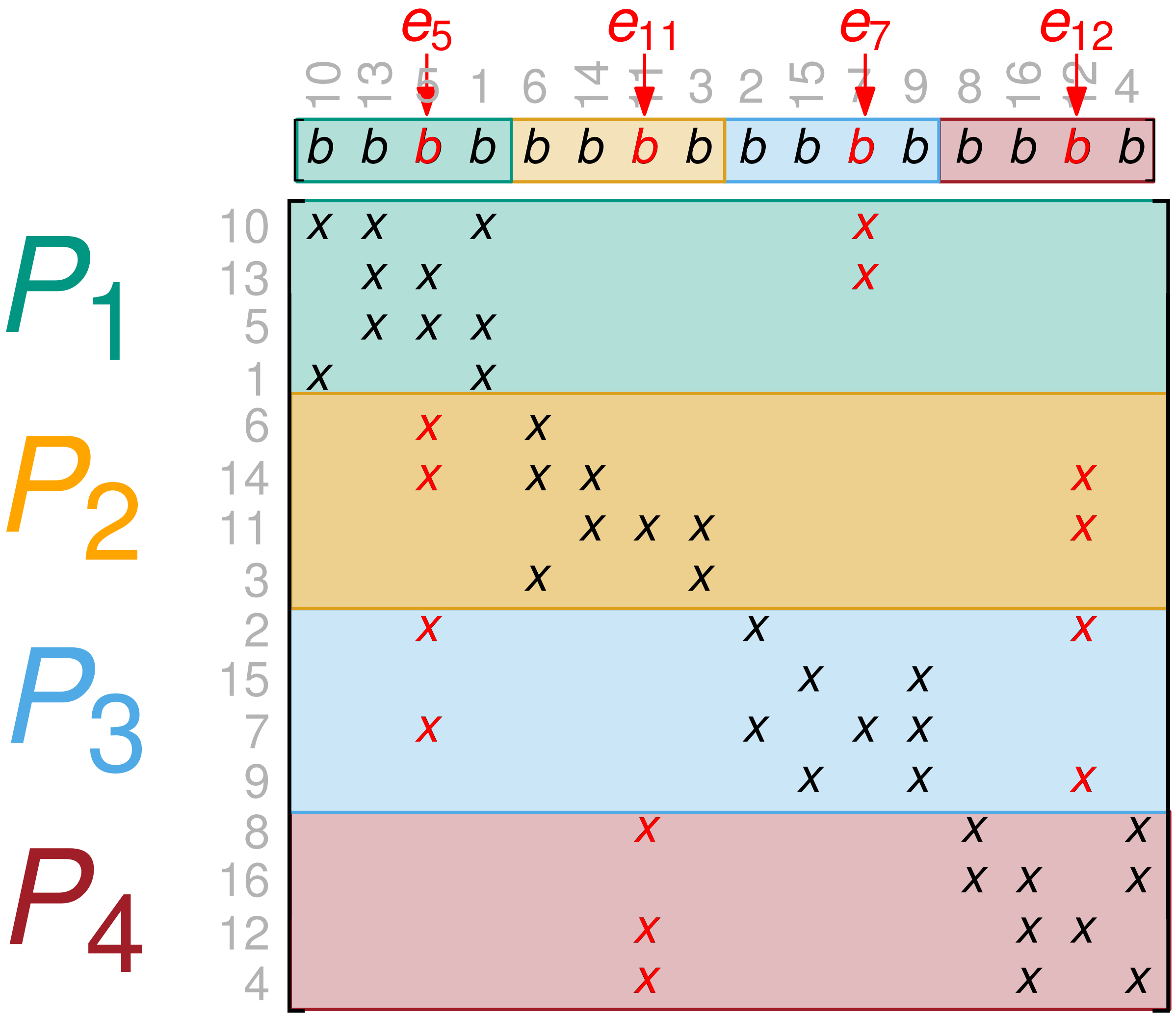
⇒ 12

⇒ 12

Commuication Volume?

From Hypergraph Partitioning to SpM×V

Where are the cut-hyperedges?



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

Load Balancing?

⇒ 12

⇒ 12

⇒ 12

⇒ 12

Commuication Volume? ⇒ 6 entries!

How does Hypergraph Partitioning work?

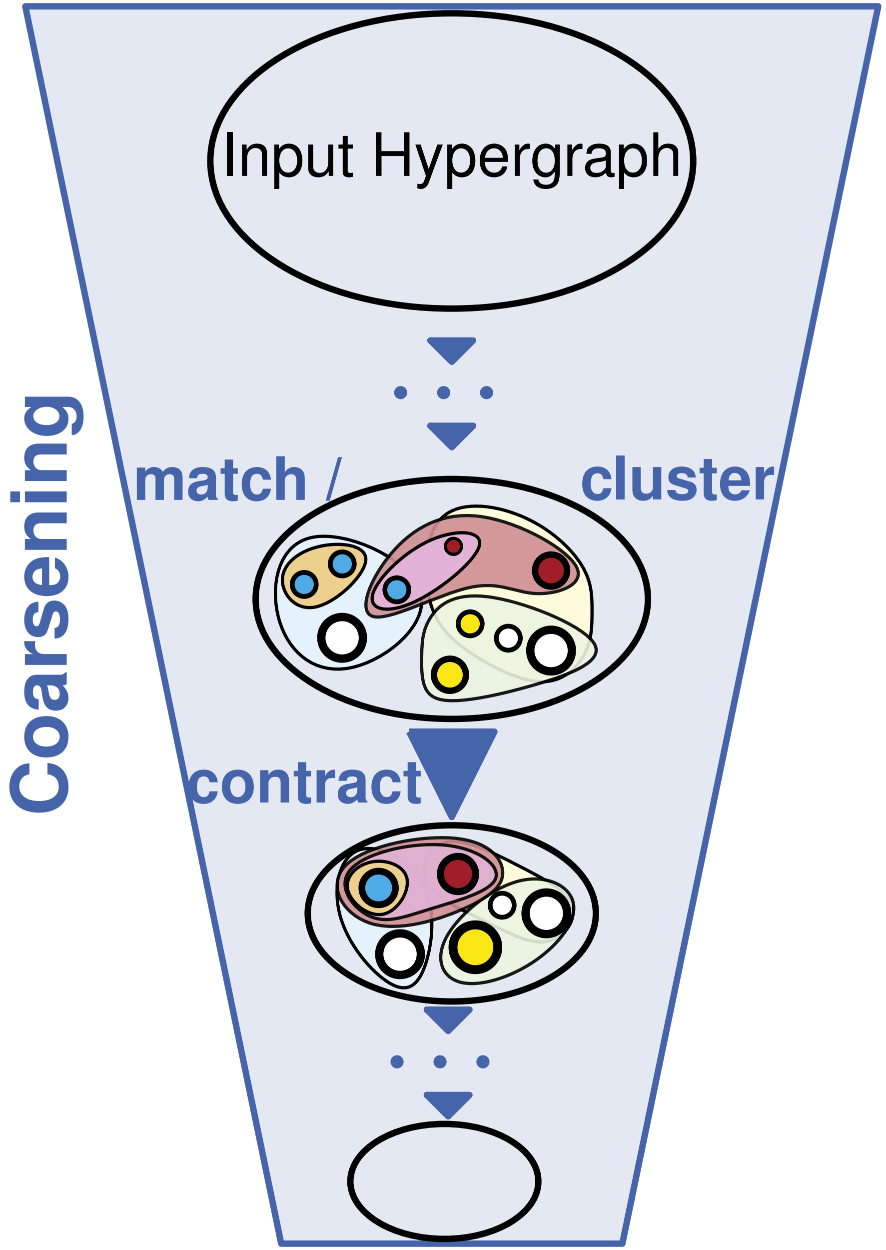
How does

Bad News:

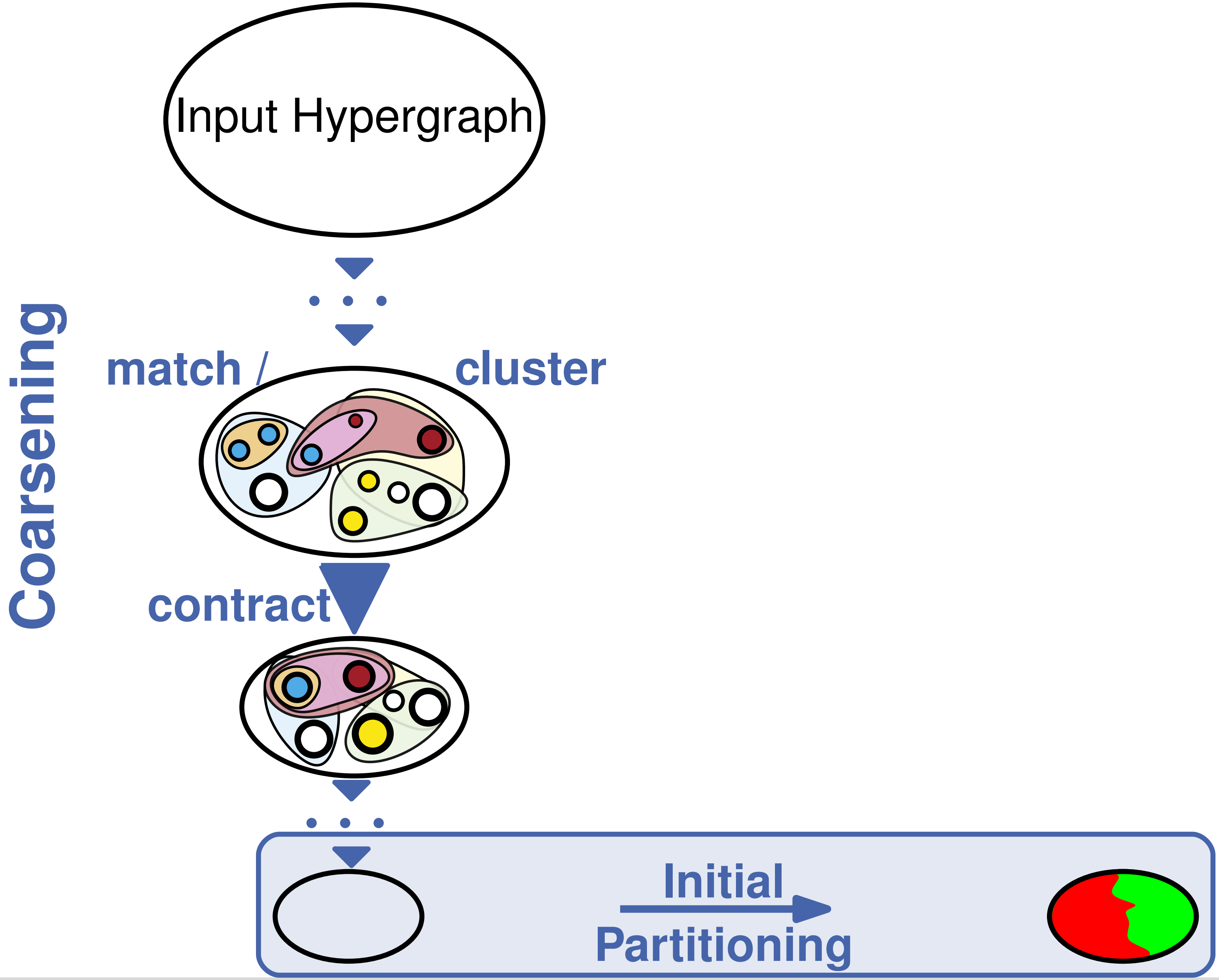
- hypergraph partitioning is NP-hard
- even finding good approximate solutions for graphs is NP-hard

work?

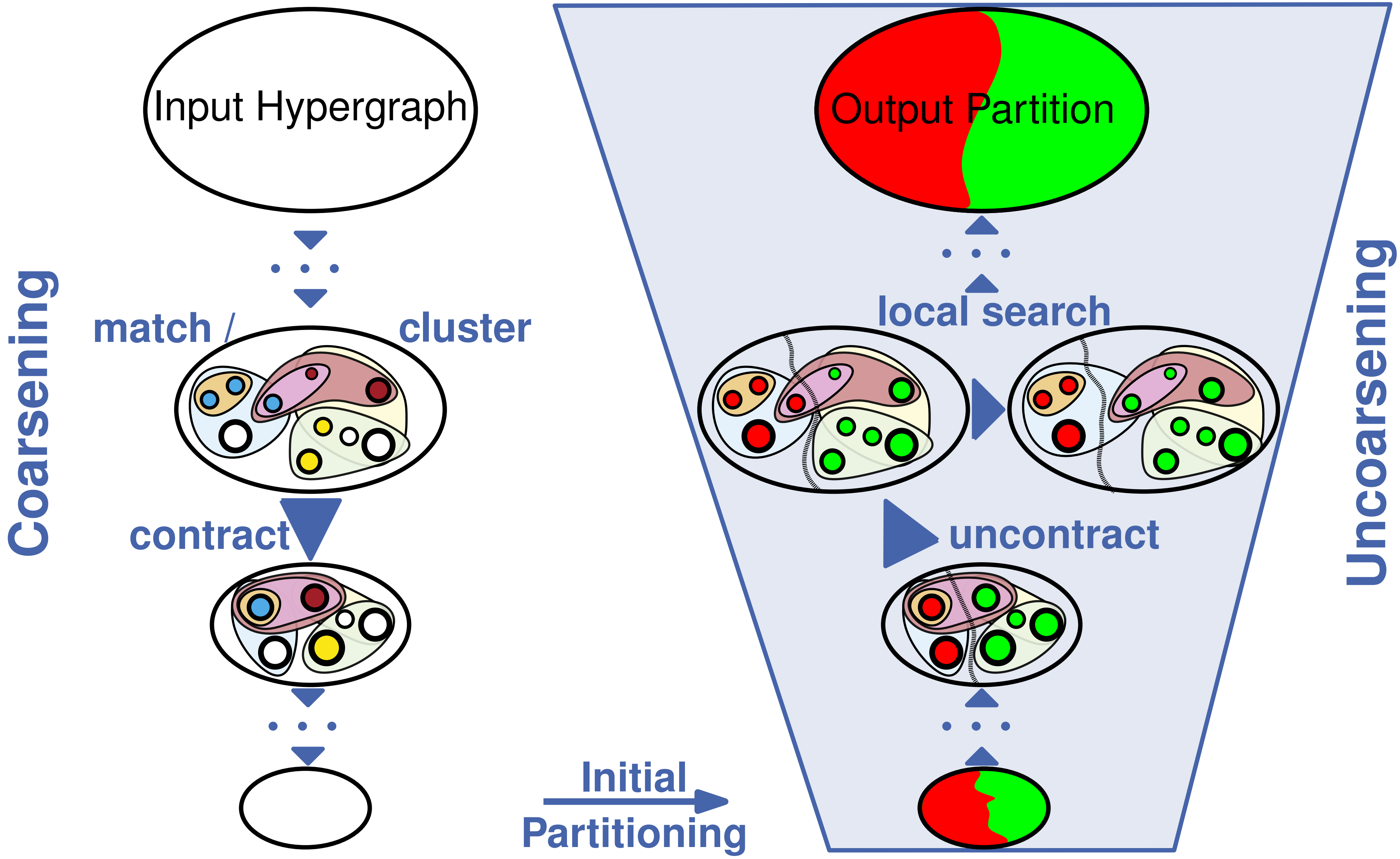
Successful Heuristic: Multilevel Paradigm



Successful Heuristic: Multilevel Paradigm



Successful Heuristic: Multilevel Paradigm



Coarsening

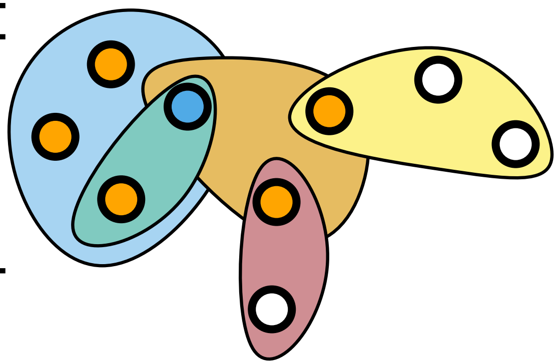
Clustering-based Coarsening

Common Strategy: **avoid** global decisions ⇝ **local**, greedy algorithms

Objective: identify highly connected vertices

using...

```
foreach vertex $v$ do
    cluster[$v$] := $\operatorname{argmax}_{\text{neighbor } u}$ rating($v$,$u$)
```



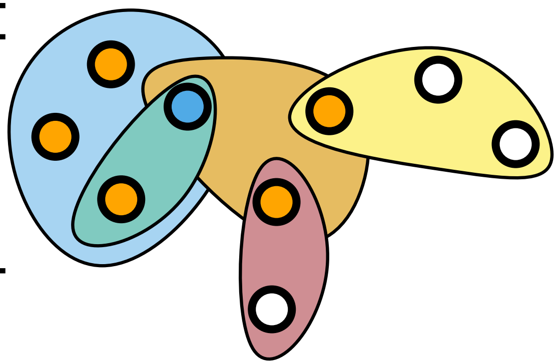
Clustering-based Coarsening

Common Strategy: avoid global decisions ⇝ **local**, greedy algorithms

Objective: identify highly connected vertices

using...

```
foreach vertex v do
    cluster[v] := $\operatorname*{argmax}_{\text{neighbor } u}$ rating(v,u)
```



Main Design Goals: [Karypis,Kumar 99]

1: reduce **size** of nets ⇝ easier local search

2: reduce **number** of nets ⇝ easier initial partitioning

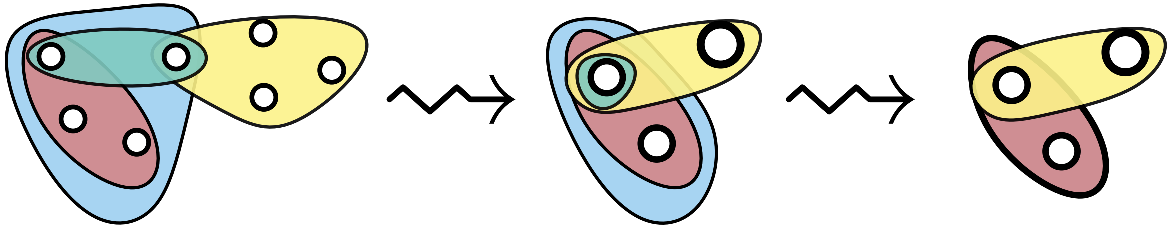
3: maintain **structural similarity** ⇝ good coarse solutions

Clustering-based Coarsening

Main Design Goals:

1: reduce **size** of nets ⇝ easier local search

2: reduce **number** of nets ⇝ easier initial partitioning

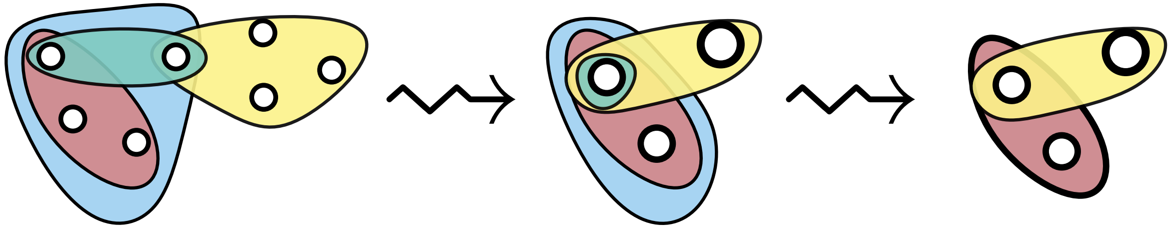


Clustering-based Coarsening

Main Design Goals:

1: reduce **size** of nets ⇝ easier local search

2: reduce **number** of nets ⇝ easier initial partitioning



⟹ hypergraph-tailored rating functions:

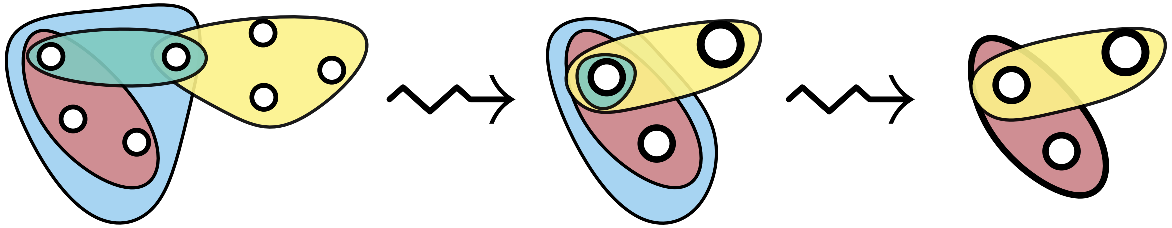
$$r(u,v) := \sum_{\substack{\text{net } e \\ \text{containing } u,v}} \frac{\omega(e)}{|e|-1}$$

Clustering-based Coarsening

Main Design Goals:

1: reduce **size** of nets ⇝ easier local search

2: reduce **number** of nets ⇝ easier initial partitioning



⟹ hypergraph-tailored rating functions:

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

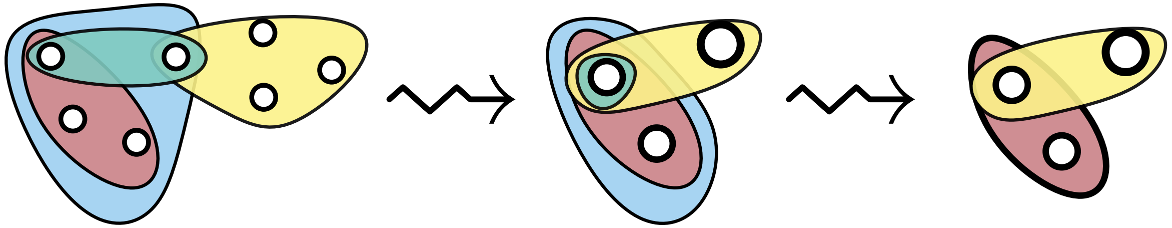
large number ...

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:

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

$\longleftarrow$ of heavy nets ...

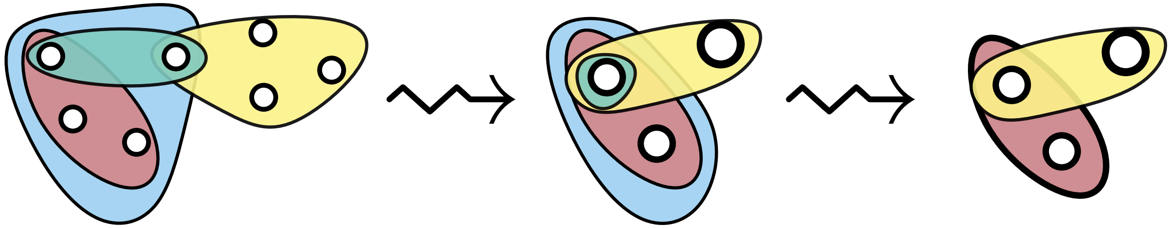
large number ... $\longrightarrow$

Clustering-based Coarsening

Main Design Goals:

1: reduce **size** of nets ⇝ easier local search

2: reduce **number** of nets ⇝ easier initial partitioning



⟹ hypergraph-tailored rating functions:

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

of heavy nets ...

... with small size

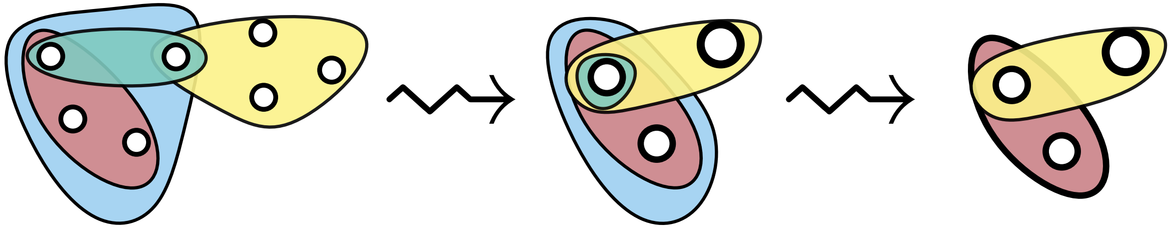
large number ...

Clustering-based Coarsening

Main Design Goals:

1: reduce **size** of nets ⇝ easier local search ✓

2: reduce **number** of nets ⇝ easier initial partitioning ✓



⟹ hypergraph-tailored rating functions:

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

of heavy nets ...

... with small size

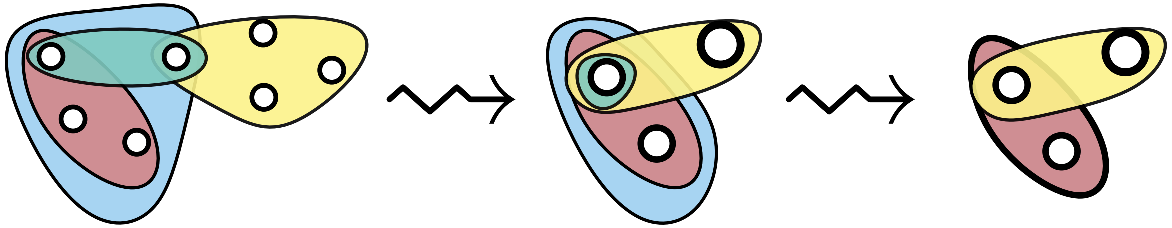
large number ...

Clustering-based Coarsening

Main Design Goals:

1: reduce **size** of nets ⇝ easier local search ✓

2: reduce **number** of nets ⇝ easier initial partitioning ✓



⇛ hypergraph-tailored rating functions:

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

of heavy nets ...

... with small size

large number ...

3: maintain **structural similarity** ⇝ good coarse solutions

⇛ prefer clustering over matching

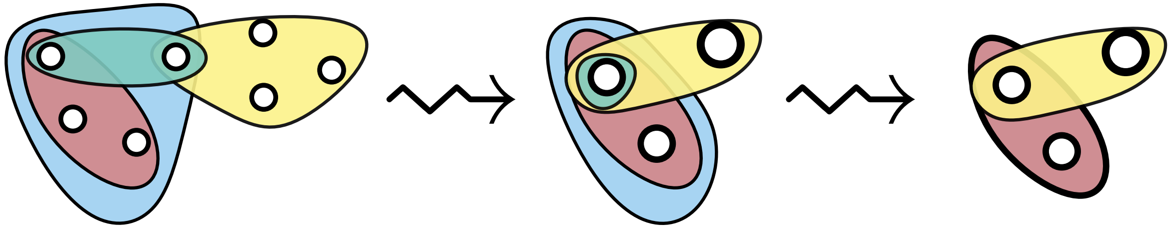
⇛ ensure ∼balanced vertex weights

Clustering-based Coarsening

Main Design Goals:

1: reduce **size** of nets ⇝ easier local search ✓

2: reduce **number** of nets ⇝ easier initial partitioning ✓



⟹ hypergraph-tailored rating functions:

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

of heavy nets ...

... with small size

large number ...

3: maintain **structural similarity** ⇝ good coarse solutions

⟹ prefer clustering over matching

⟹ ensure ∼balanced vertex weights

enough?

What could possibly go wrong?

... a lot:

input

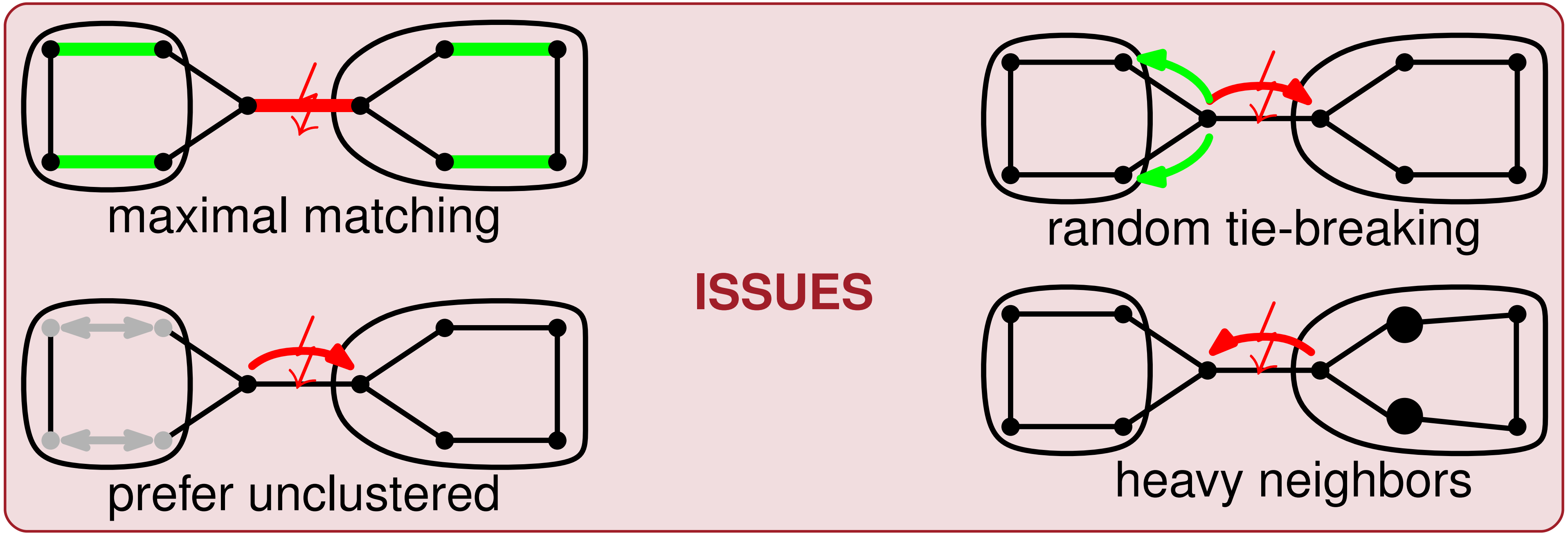
obscured clusters

What could possibly go wrong?

... a lot:

input

obscured clusters



maximal matching

random tie-breaking

ISSUES

prefer unclustered

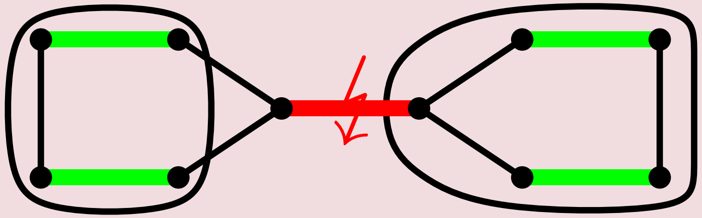
heavy neighbors

What could possibly go wrong?

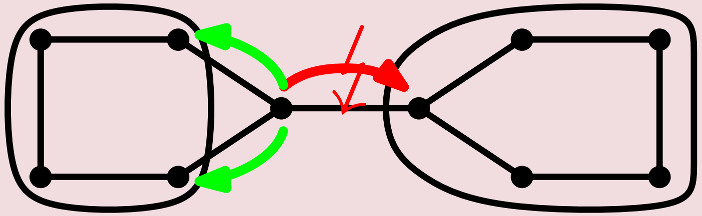
... a lot:

input

obscured clusters

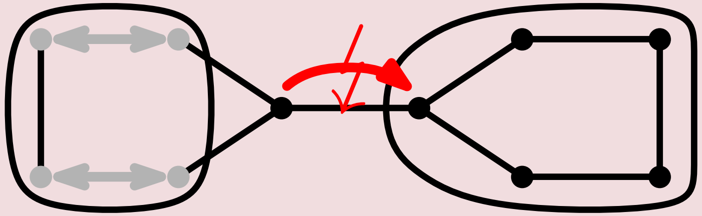


maximal matching

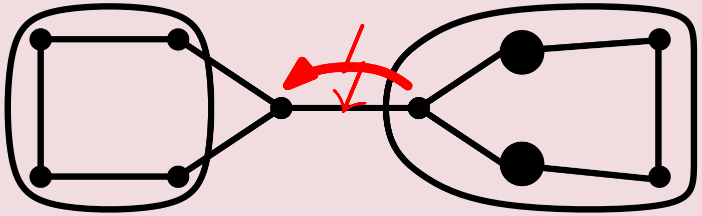


random tie-breaking

ISSUES



prefer unclustered



heavy neighbors

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

Community-aware Coarsening

input

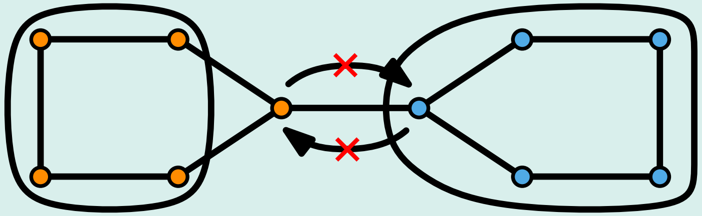
obscured clusters

Community-aware Coarsening

input

obscured clusters

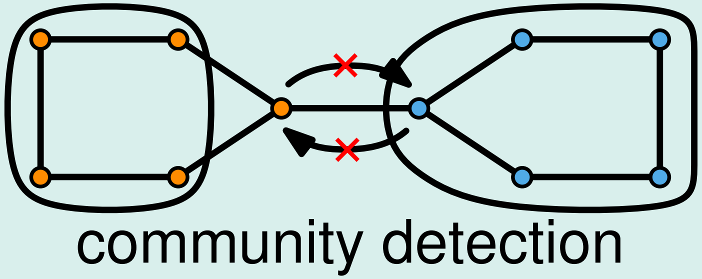
SOLUTION



community detection

Community-aware Coarsening

SOLUTION



Framework:

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

Initial Partitioning

Initial Partitioning

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

⇒ try all algorithms multiple times

⇒ select partition with **best** cut & **lowest** imbalance as initial partition

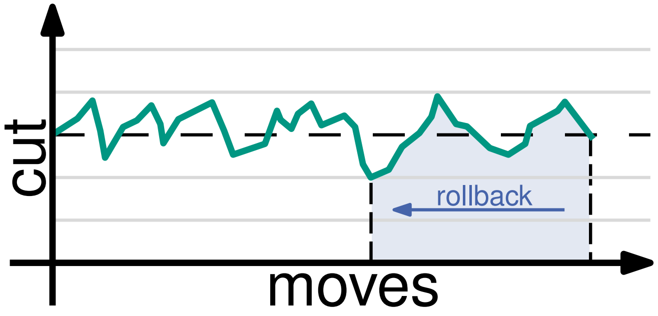
initial partition

Local Search

Fiduccia-Mattheyses Algorithm

Algorithm 1: FM Local Search

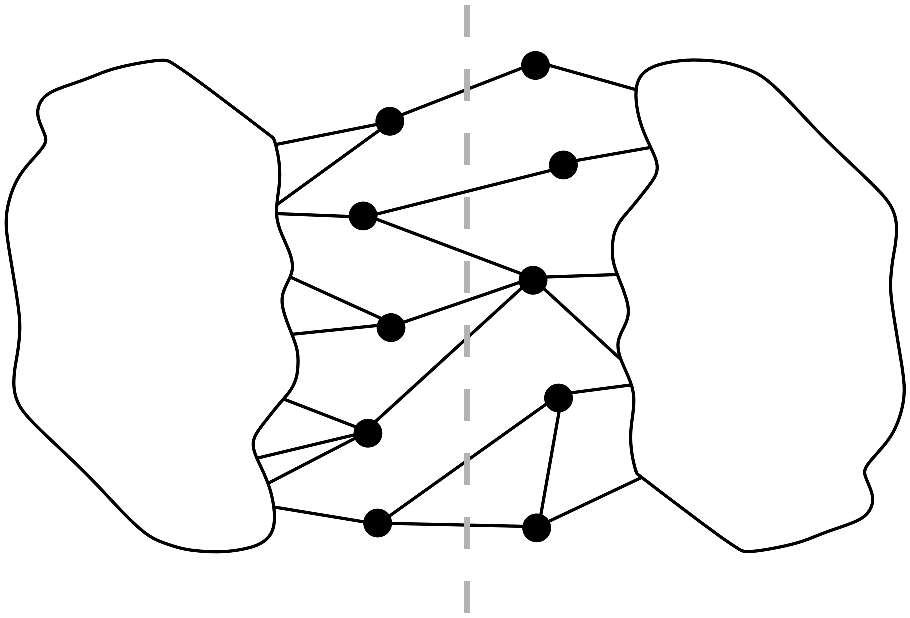
```
while ¬ done do
    find best move
    perform best move
rollback to best solution
```



can worsen solution

Example for Graphs:

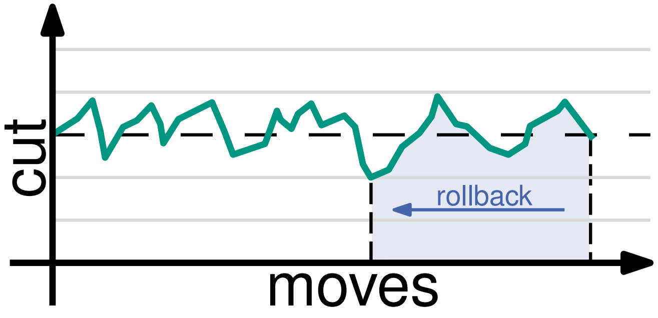
- compute gain $g(v) = d_{\text{ext}}(v) - d_{\text{int}}(v)$
- alternate between blocks
- edge-cut: **7**



Fiduccia-Mattheyses Algorithm

Algorithm 1: FM Local Search

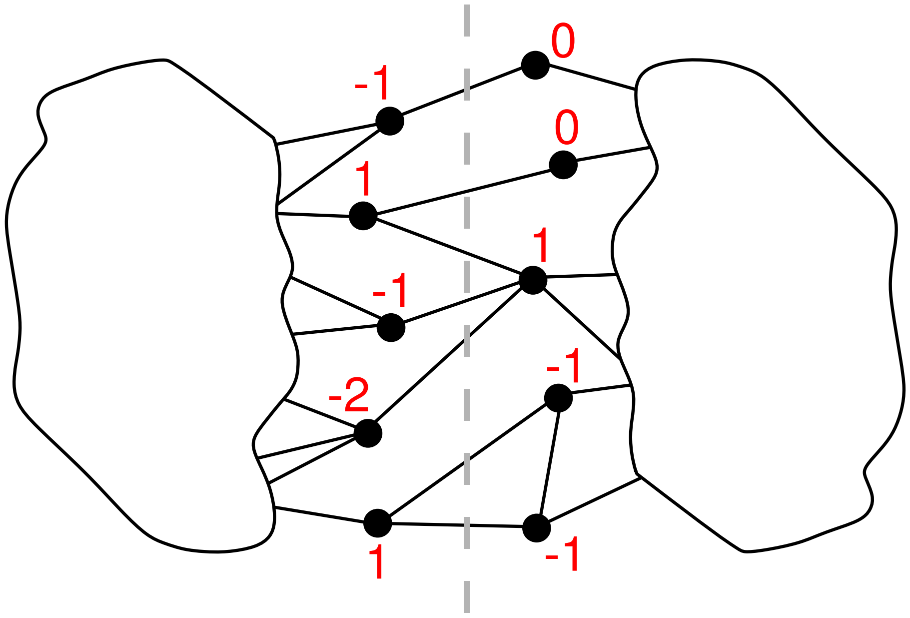
```
while ¬ $done$ do
    find best move
    perform best move
rollback to best solution
```



can worsen solution

Example for Graphs:

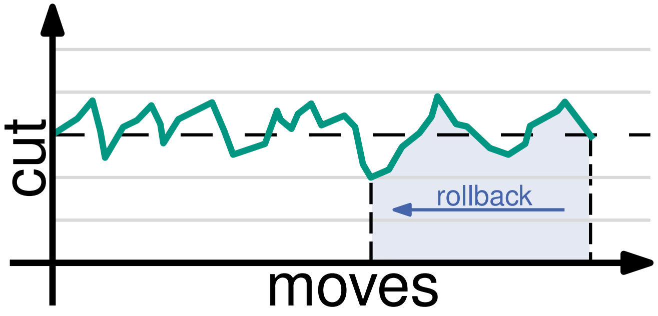
- compute gain $g(v) = d_{\text{ext}}(v) - d_{\text{int}}(v)$
- alternate between blocks
- edge-cut: **7**



Fiduccia-Mattheyses Algorithm

Algorithm 1: FM Local Search

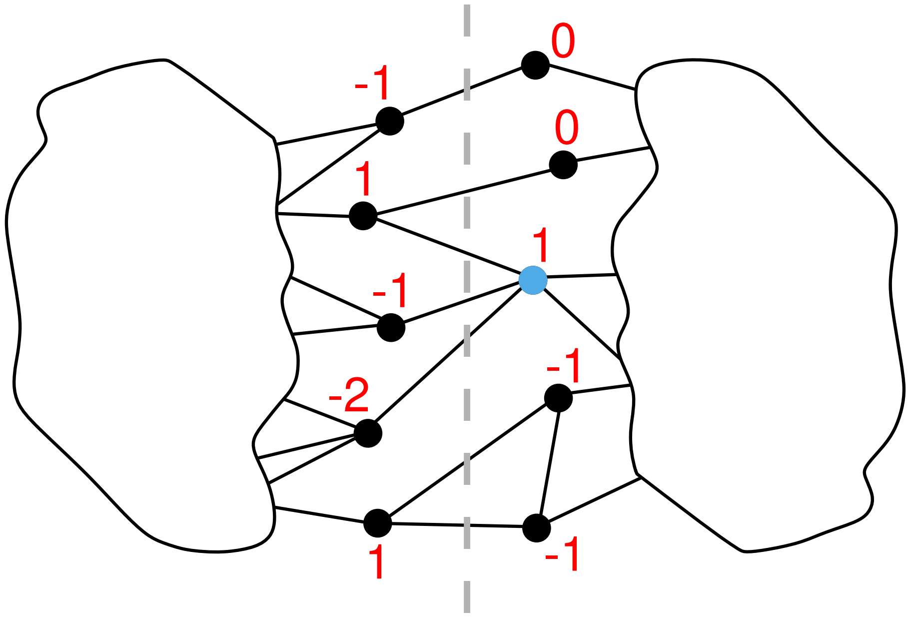
```
while ¬ done do
    find best move
    perform best move
rollback to best solution
```



can worsen solution

Example for Graphs:

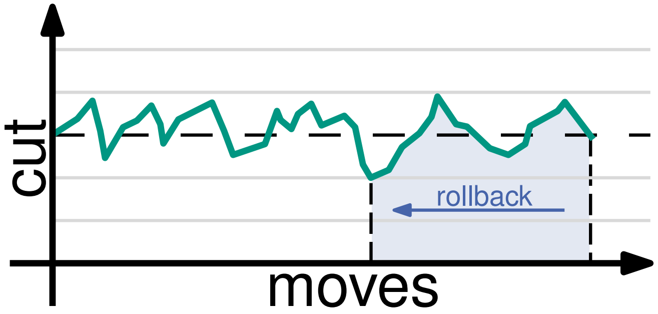
- compute gain $g(v) = d_{\text{ext}}(v) - d_{\text{int}}(v)$
- alternate between blocks
- edge-cut: **7**



Fiducci-Mattheyses Algorithm

Algorithm 1: FM Local Search

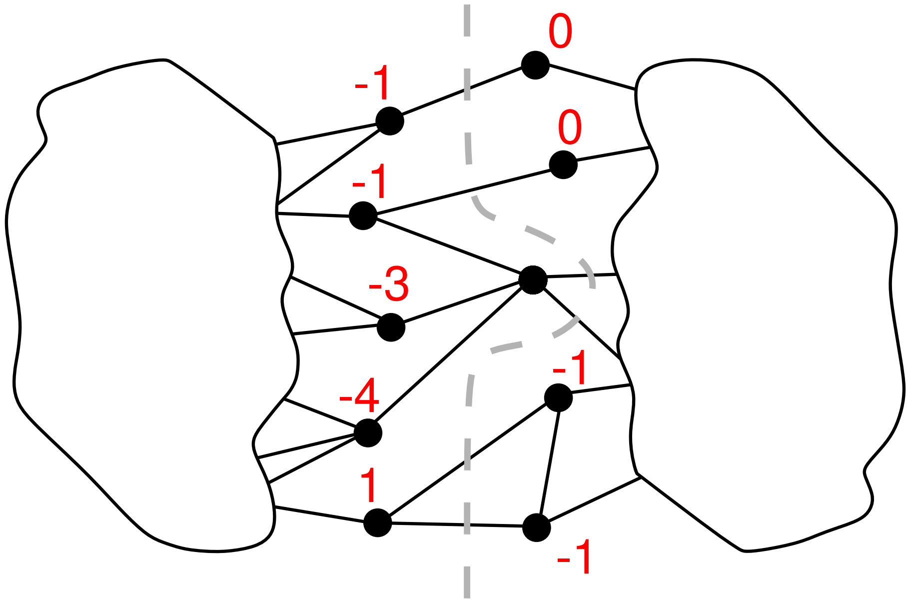
```
while ¬ done do
    find best move
    perform best move
rollback to best solution
```



can worsen solution

Example for Graphs:

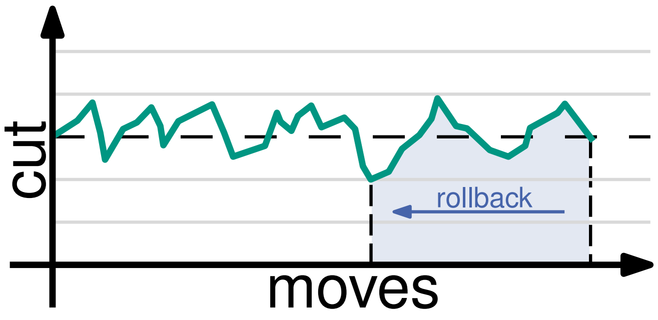
- recalculate gain $g(v)$ of neighbors
- move each node at most once
- edge-cut: **7**, **6**



Fiduccia-Mattheyses Algorithm

Algorithm 1: FM Local Search

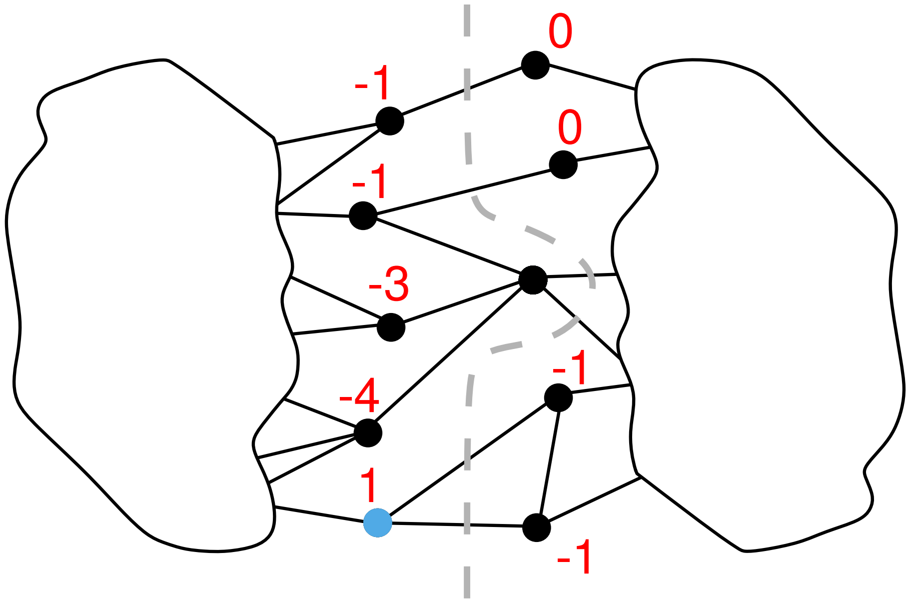
```
while ¬ done do
    find best move
    perform best move
rollback to best solution
```



can worsen solution

Example for Graphs:

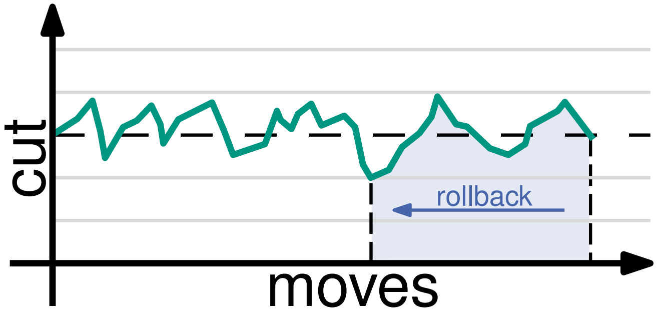
- recalculate gain $g(v)$ of neighbors
- move each node at most once
- edge-cut: **7**, **6**



Fiduccia-Mattheyses Algorithm

Algorithm 1: FM Local Search

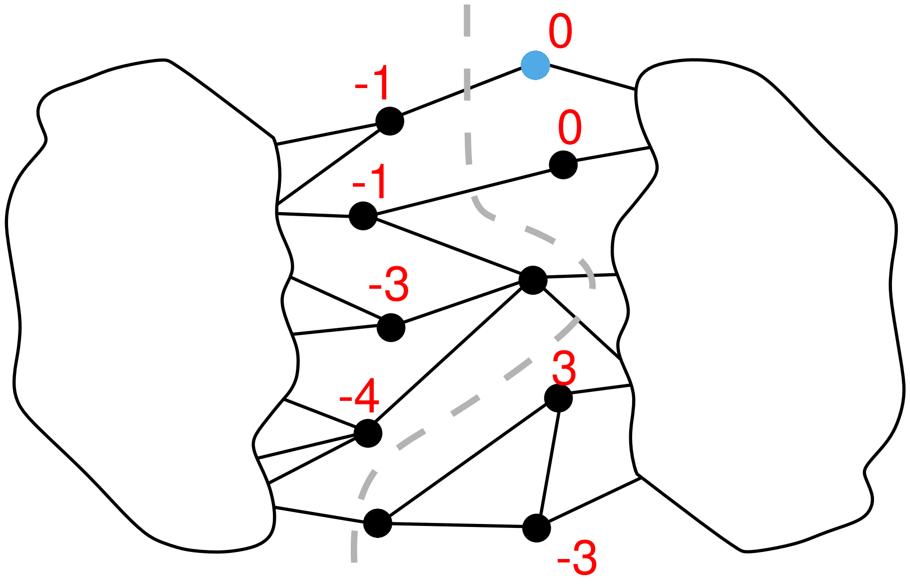
```
while ¬ done do
    find best move
    perform best move
rollback to best solution
```



can worsen solution

Example for Graphs:

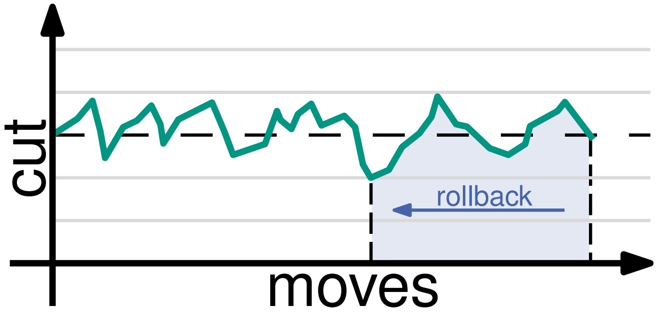
- recalculate gain $g(v)$ of neighbors
- move each node at most once
- edge-cut: **7**, **6**,**5**



Fiduccia-Mattheyses Algorithm

Algorithm 1: FM Local Search

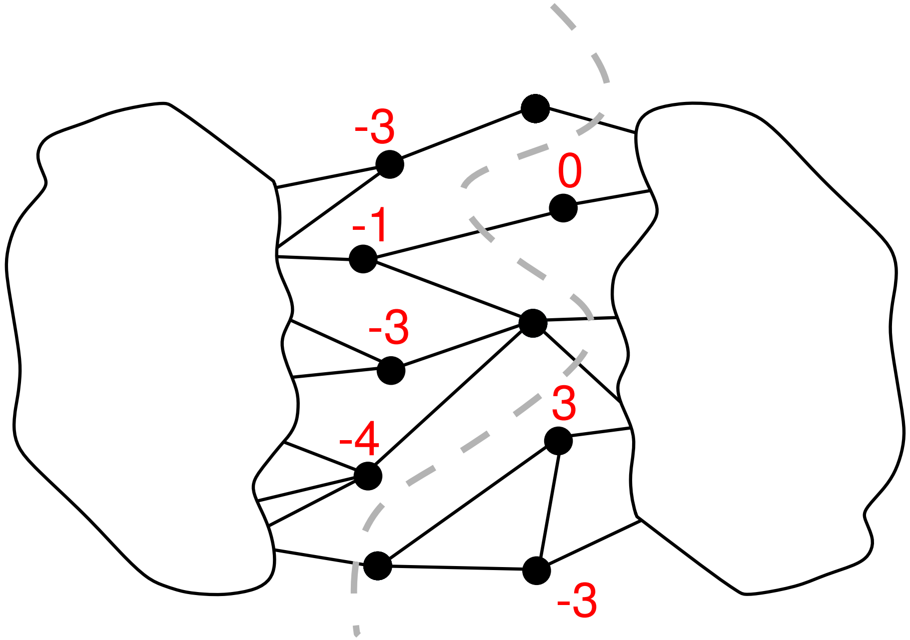
```
while ¬ done do
    find best move
    perform best move
rollback to best solution
```



can worsen solution

Example for Graphs:

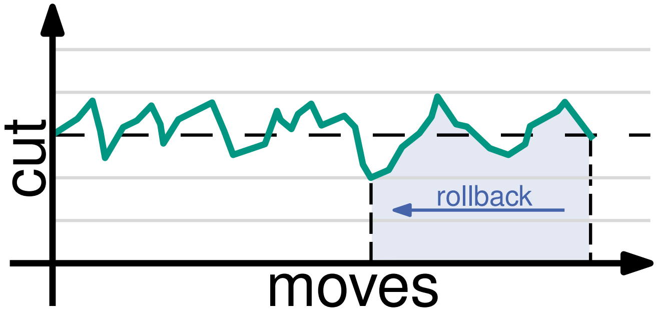
- recalculate gain $g(v)$ of neighbors
- move each node at most once
- edge-cut: **7**, **6**,**5**,**5**



Fiduccia-Mattheyses Algorithm

Algorithm 1: FM Local Search

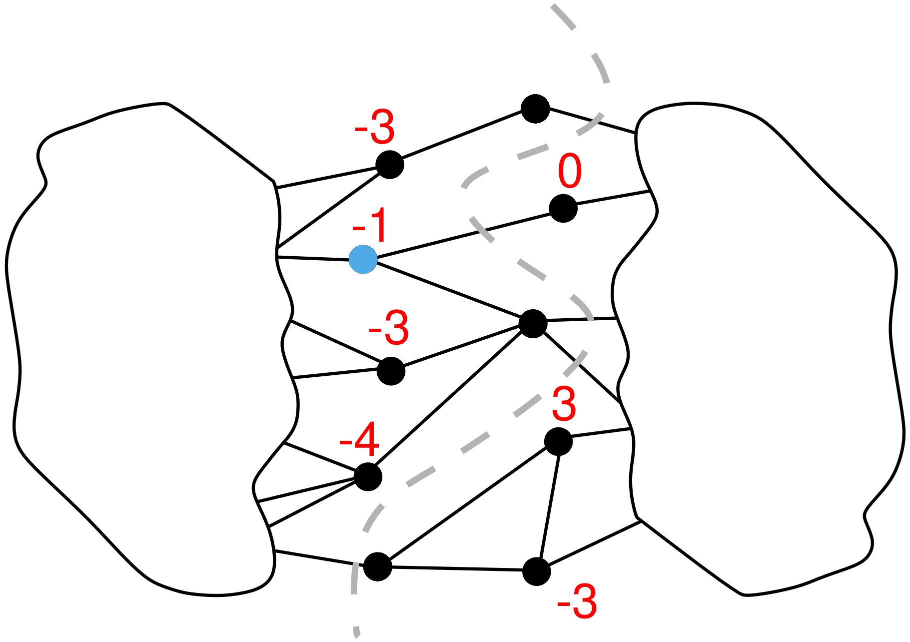
```
while ¬ done do
    find best move
    perform best move
rollback to best solution
```



can worsen solution

Example for Graphs:

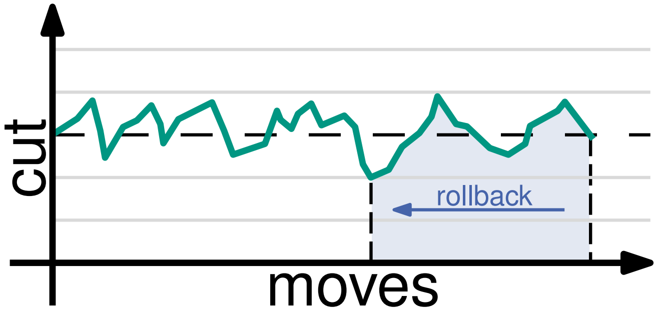
- recalculate gain $g(v)$ of neighbors
- move each node at most once
- edge-cut: **7, 6,5,5**



Fiduccia-Mattheyses Algorithm

Algorithm 1: FM Local Search

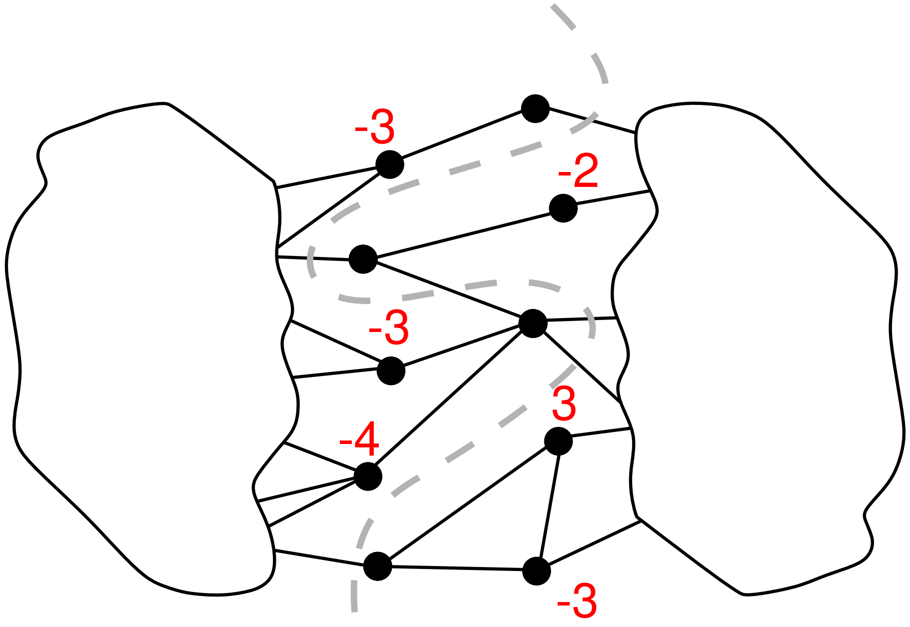
```
while ¬ done do
    find best move
    perform best move
rollback to best solution
```



can worsen solution

Example for Graphs:

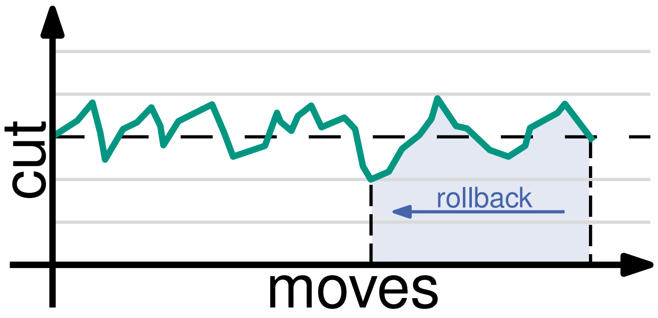
- recalculate gain $g(v)$ of neighbors
- move each node at most once
- edge-cut: **7**, **6**,**5**,**5**,**6**



Fiduccia-Mattheyses Algorithm

Algorithm 1: FM Local Search

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

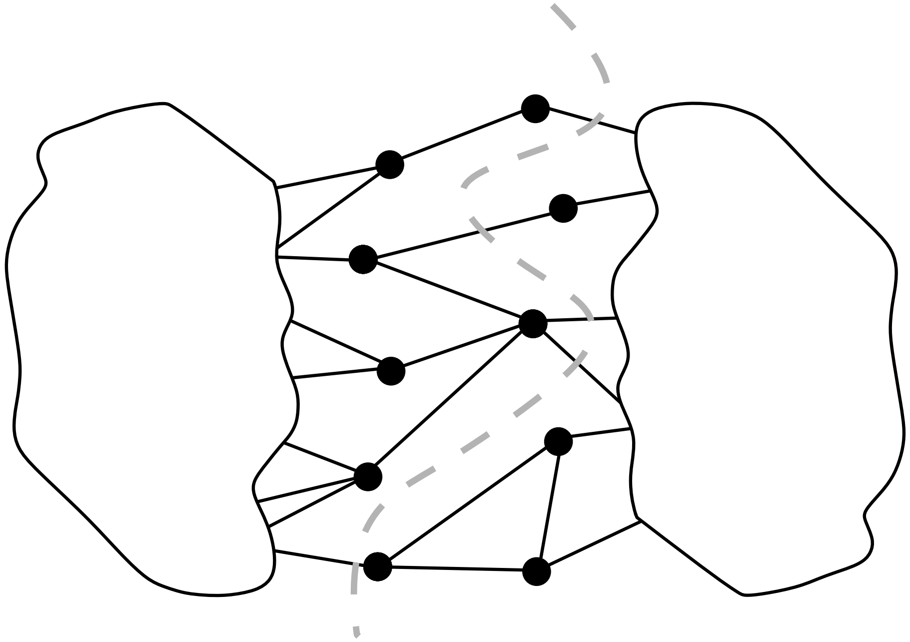


can worsen solution

Example for Graphs:

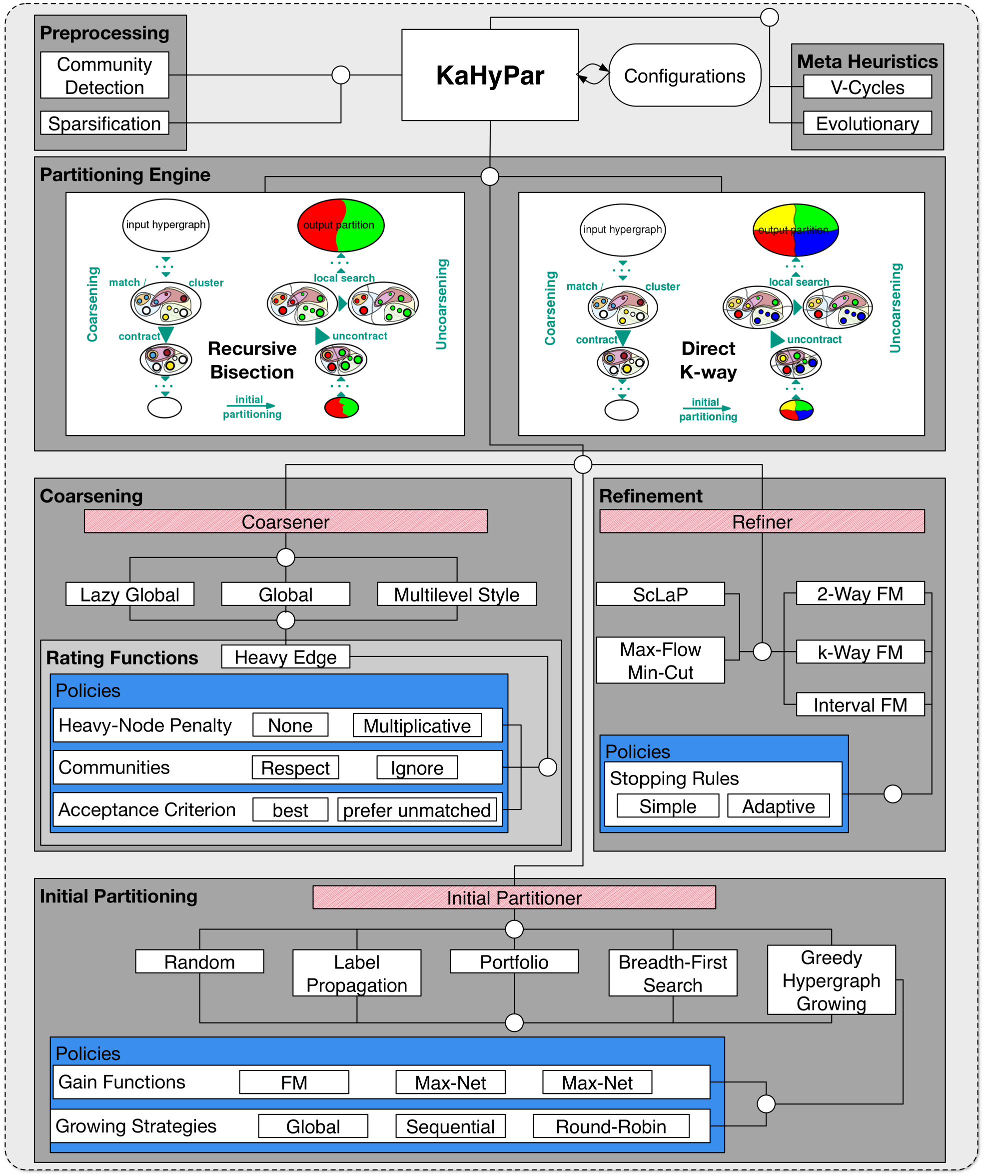
- recalculate gain $g(v)$ of neighbors
- move each node at most once
- edge-cut: **7**, **6**,**5**,**5**,**6**

rollback



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`



Preprocessing
Community Detection
Sparsification

KaHyPar
Configurations

Meta Heuristics
V-Cycles
Evolutionary

Partitioning Engine
Recursive Bisection
Direct K-way

Coarsening
Coarsener
Lazy Global
Global
Multilevel Style

Rating Functions
Heavy Edge

Policies

Heavy-Node Penalty	None	Multiplicative
Communities	Respect	Ignore
Acceptance Criterion	best	prefer unmatched

Refinement
Refiner
ScLaP
Max-Flow Min-Cut
2-Way FM
k-Way FM
Interval FM

Policies
Stopping Rules
Simple
Adaptive

Initial Partitioning
Initial Partitioner
Random
Label Propagation
Portfolio
Breadth-First Search
Greedy Hypergraph Growing

Policies

Gain Functions	FM	Max-Net	Max-Net
Growing Strategies	Global	Sequential	Round-Robin

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.