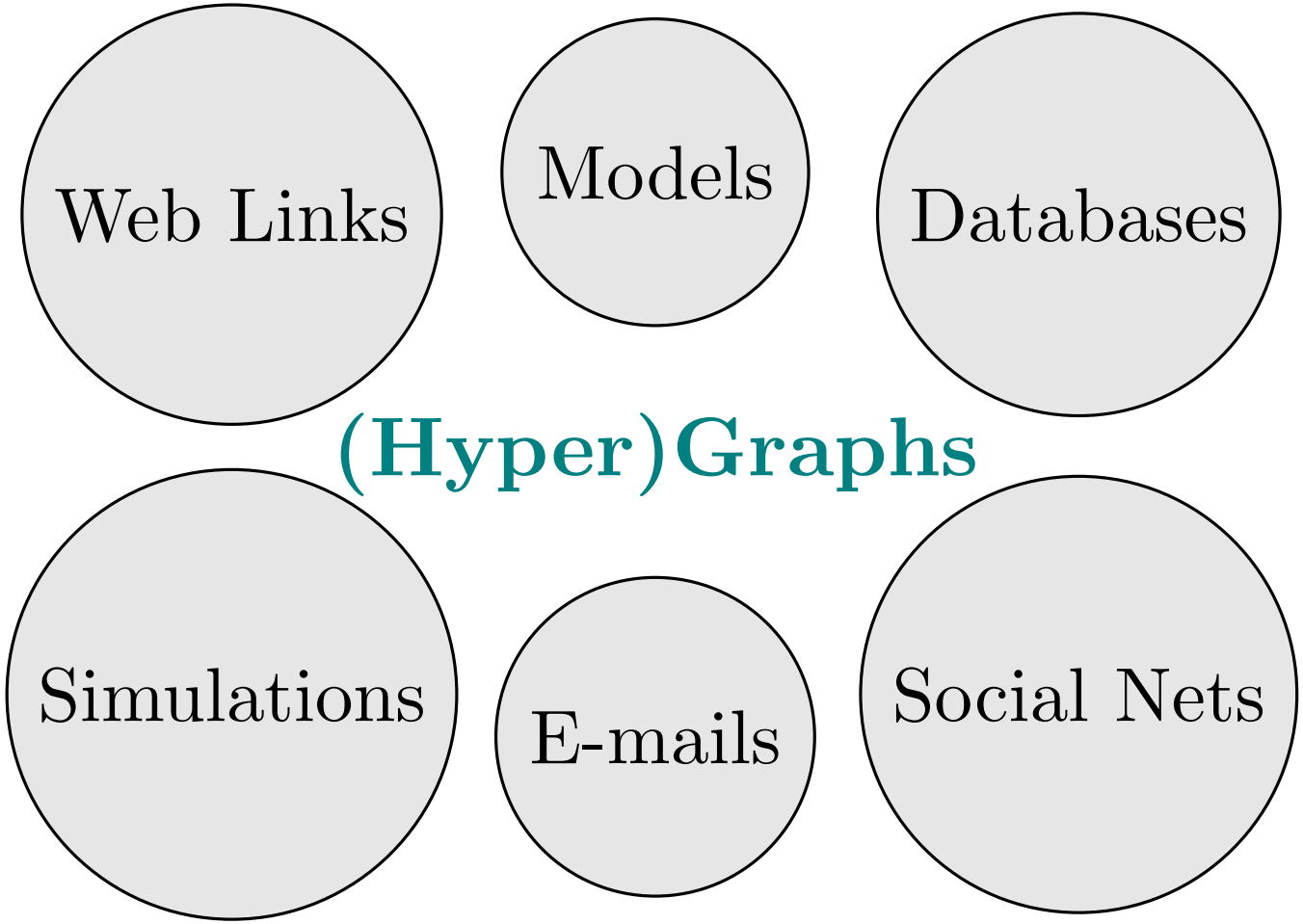
# FREIGHT: Fast Streaming Hypergraph Partitioning

Kamal Eyubov, Heidelberg University
**Marcelo Fonseca Faraj, Heidelberg University**
Christian Schulz, Heidelberg University

# Introduction

## Ubiquity of (Hyper)Graphs



# Introduction

## Processing Large-Scale (Hyper)Graphs

Split workload among processors

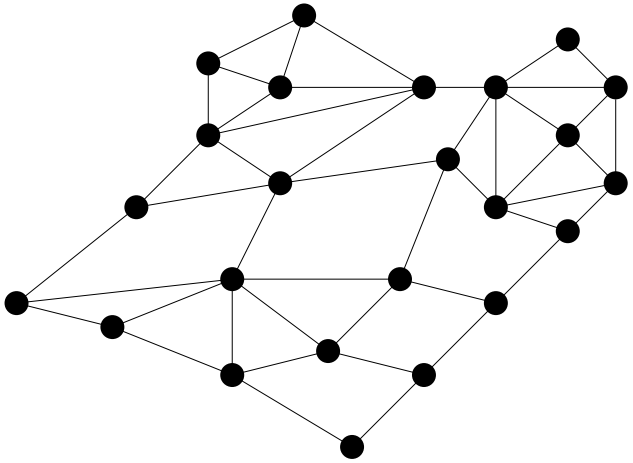
- In roughly equal blocks
- Cutting as little as possible

# Graphs and Hypergraphs

Let $n = |V|, m = |E|$

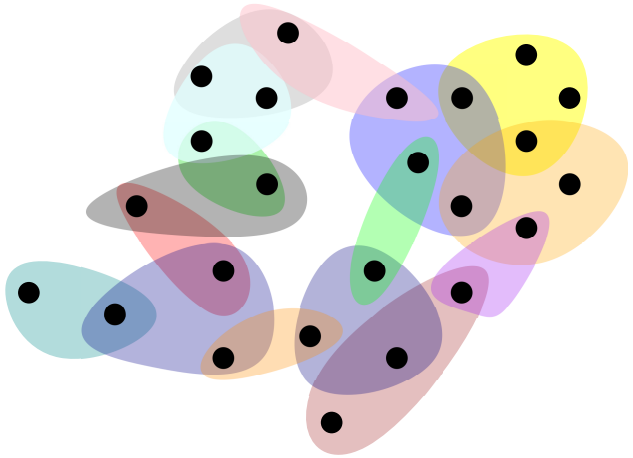
Graph $G = (V, E)$

- Edges connect two nodes
- Memory required to store it: $O(n + m)$



Hypergraphs $H = (V, E)$

- Nets connect two or more nodes
- We say a net $e$ has $|e|$ pins
- Memory required to store it: $O\left(n + \sum_{e \in E} |e|\right)$

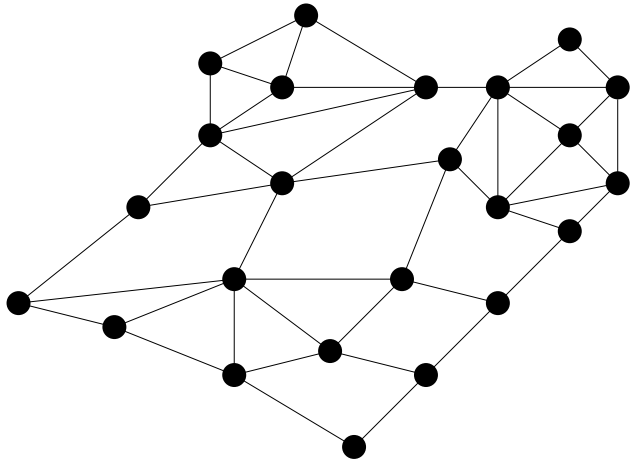


# Graphs and Hypergraphs

Let $n = |V|, m = |E|, c : V \to \mathbf{R}_{>0}, \omega : E \to \mathbf{R}_{>0}$

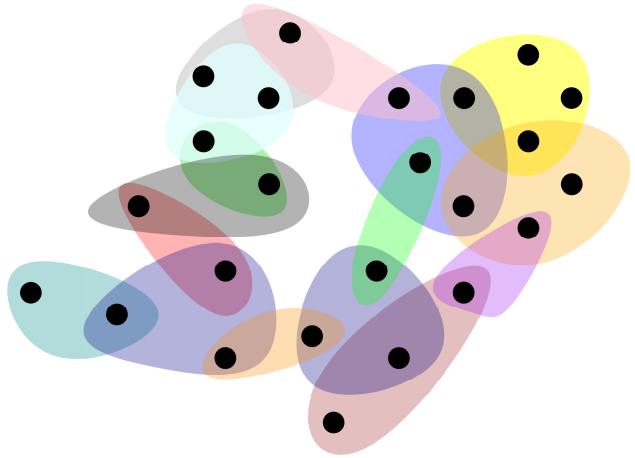
Graph $G = (V, E)$

- Edges connect two nodes
- Memory required to store it: $O(n + m)$



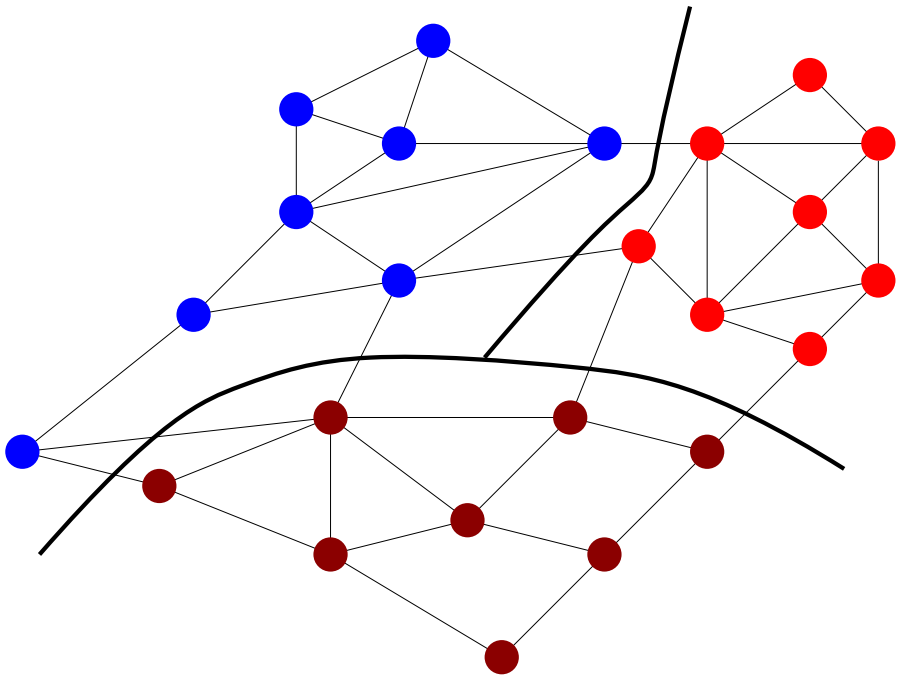
Hypergraphs $H = (V, E)$

- Nets connect two or more nodes
- We say a net $e$ has $|e|$ pins
- Memory required to store it: $O\left(n + \sum_{e \in E} |e|\right)$



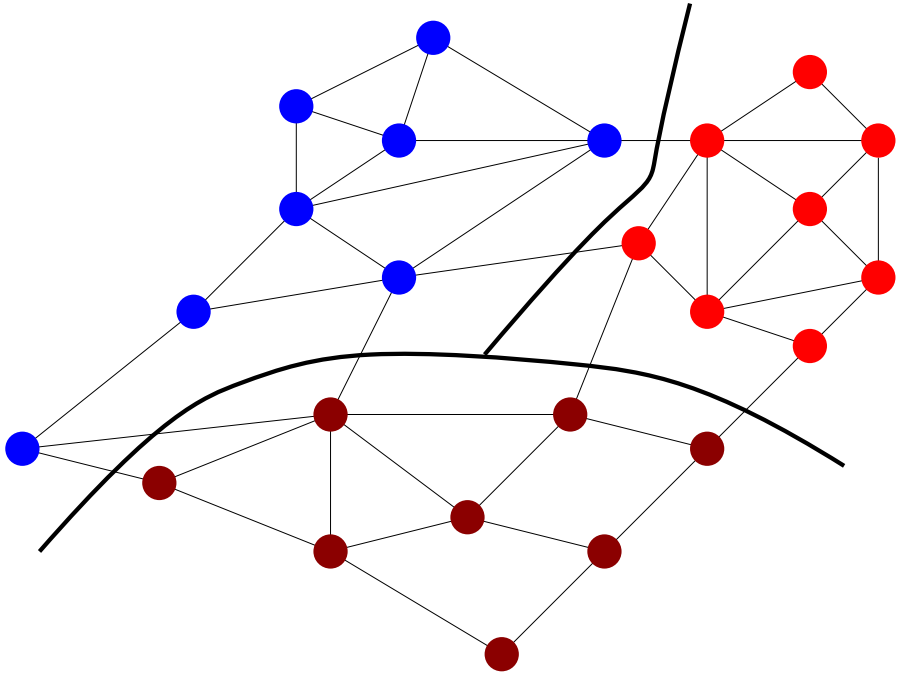
# Graph Partitioning (GP)

- Fixed amount k of disjoint blocks
- $\epsilon$-Balanced blocks: Cardinality of each block $\leq \frac{1+\epsilon}{k} n$
- Objective: minimizeedge-cut: $|Cut|$



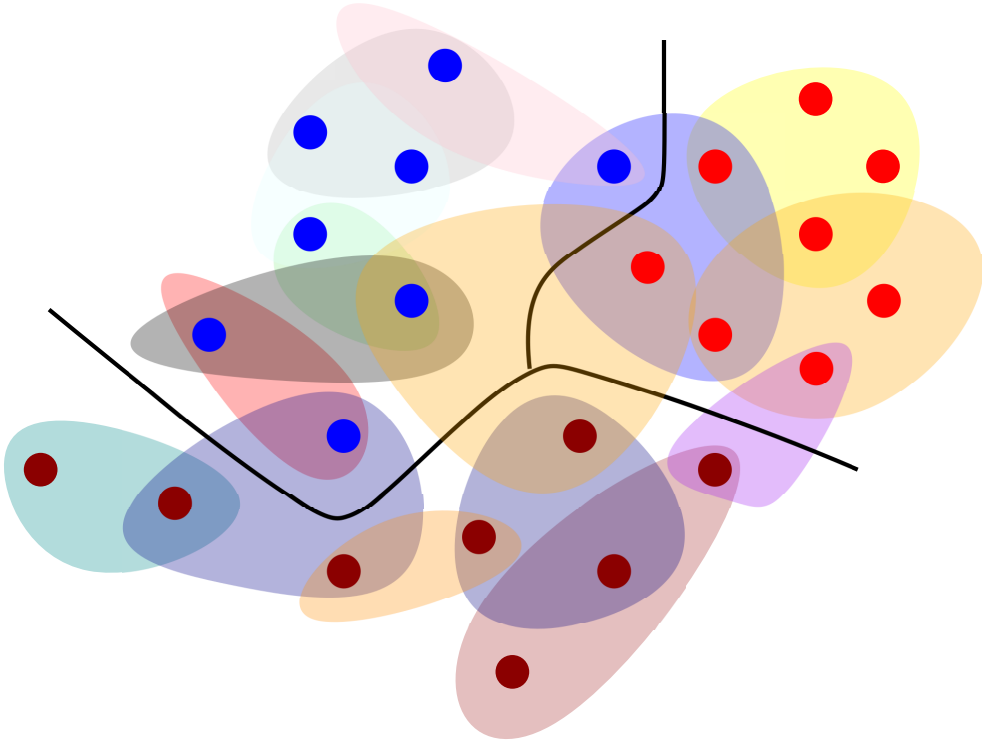
# Graph Partitioning (GP)

- Fixed amount k of disjoint blocks
- $\epsilon$-Balanced blocks: Weight of each block $\leq \frac{1+\epsilon}{k} c(V)$
- Objective: minimizeedge-cut: $\sum_{e \in \text{Cut}} \omega(e)$



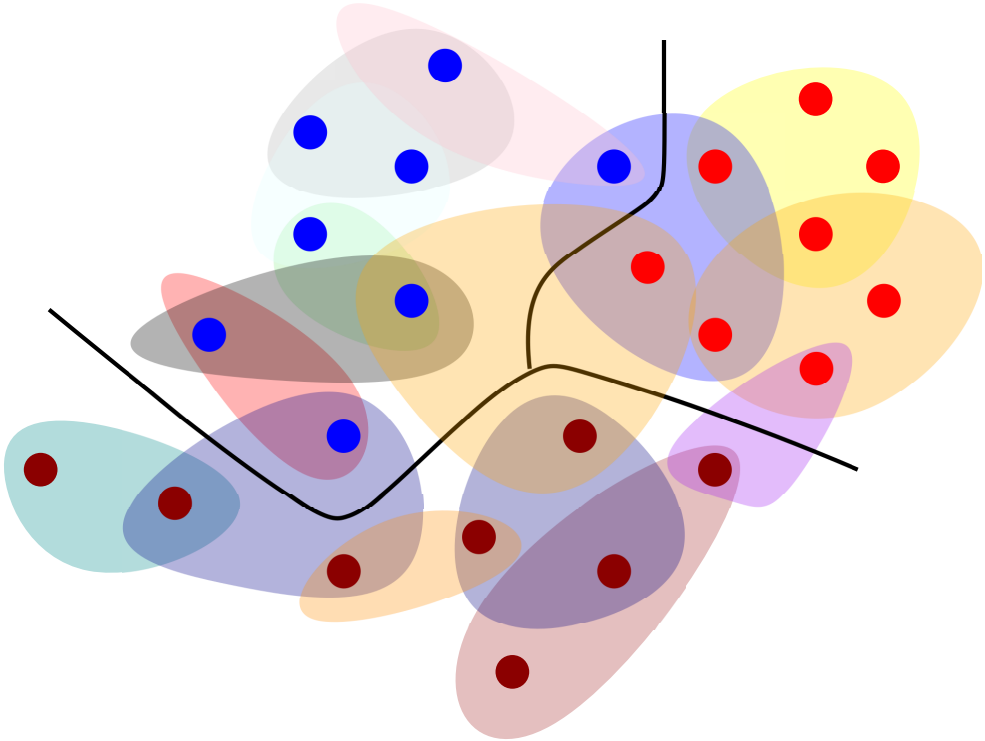
# Hypergraph Partitioning (HGP)

- Fixed amount k of disjoint blocks
- Cardinality of each block $\leq \frac{1+\epsilon}{k} n$
- Common objectives minimized: ($\lambda$ = # of blocks connected by $e$)
  - Cut-Net: $|Cut|$
  - Connectivity: $\sum_{e \in \text{Cut}} (\lambda - 1)$



# Hypergraph Partitioning (HGP)

- Fixed amount k of disjoint blocks
- Weight of each block $\leq \frac{1+\epsilon}{k} c(V)$
- Common objectives minimized: ($\lambda$ = # of blocks connected by $e$)
  - Cut-Net: $\sum_{e \in \text{Cut}} \omega(e)$
  - Connectivity: $\sum_{e \in \text{Cut}} (\lambda - 1)\omega(e)$

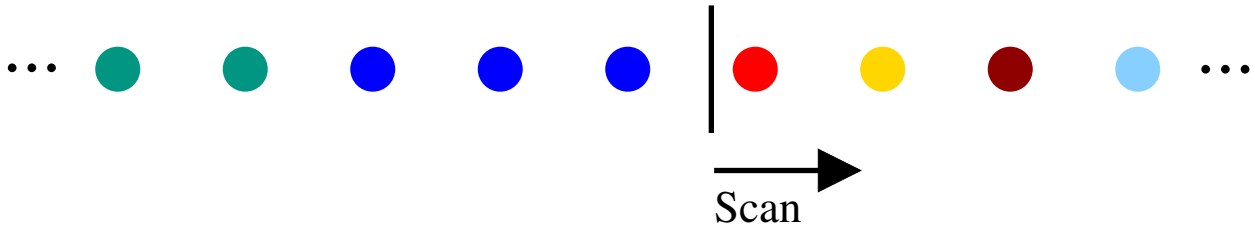


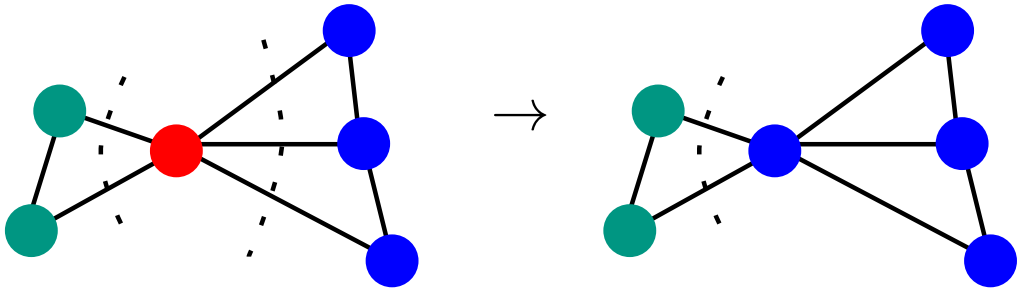
# Streaming (H)GP

Computational model: one-pass partitioning

- Nodes arrive one at a time with their (hyper)edges
- Each node is assigned on the fly
- Assignments to blocks are permanent
- 

| Memory Budged in | Streaming | In-Memory |
|---|---|---|
| Graphs | $\mathcal{O}(n+k)$ | $\mathcal{O}(n+m+k)$ |
| Hypergraphs | $\mathcal{O}(mk)$ | $\mathcal{O}\left(n+\sum_{e\in E}\lvert e\rvert+k\right)$ |



# The Case for Streaming (H)GP

Many real-world use cases

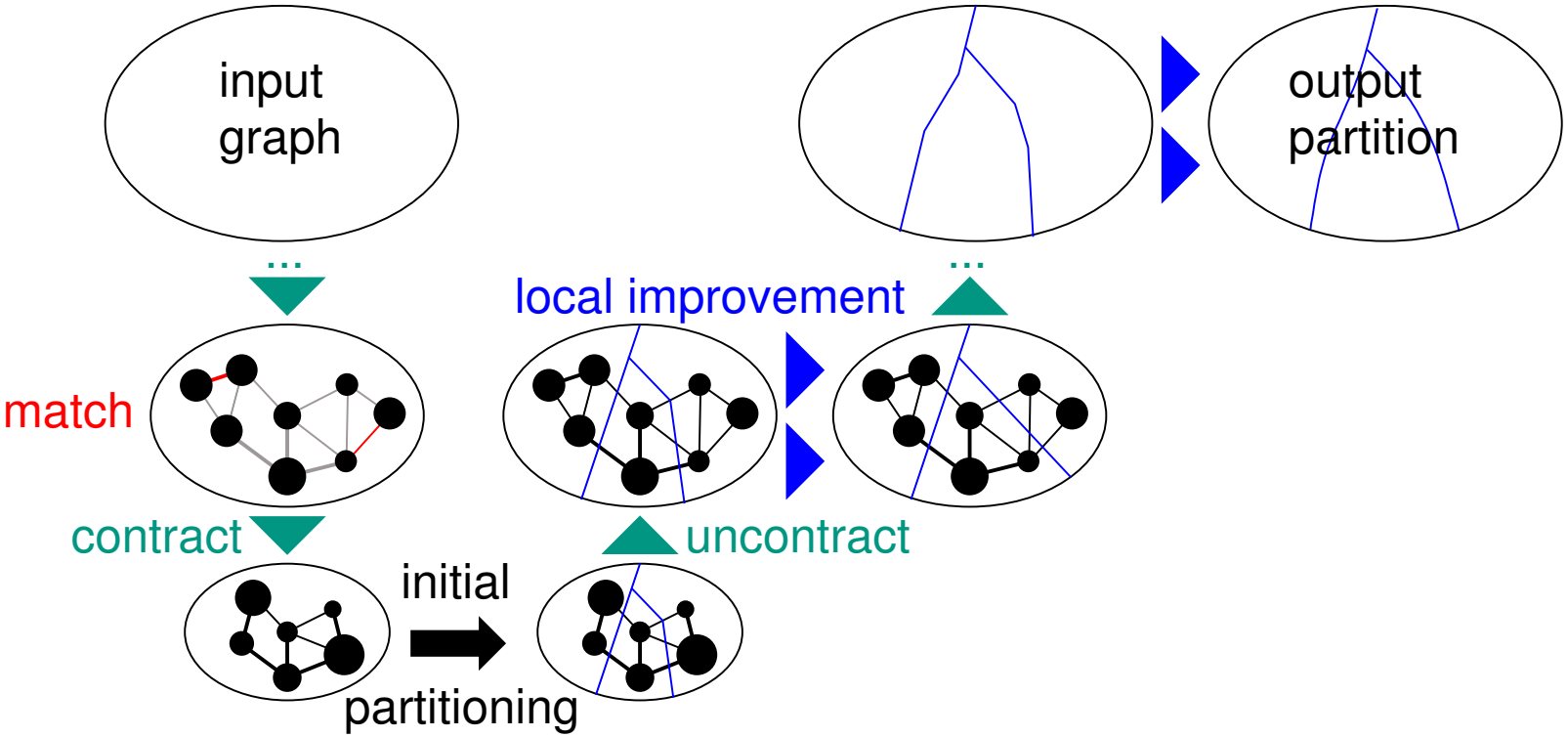
- (Hyper)Graph systems based on load-compute-store
  - Spark
  - Giraph
- Systems with (hyper)graph-as-a-stream computations
  - Flink
- Prepartitioning step for distributed (hyper)graph partitioning
- Partitioning huge (hyper)graphs using modest machines

# Related Work

# Related Work

## In-Memory GP

Multilevel Graph Partitioning: THE Traditional Approach

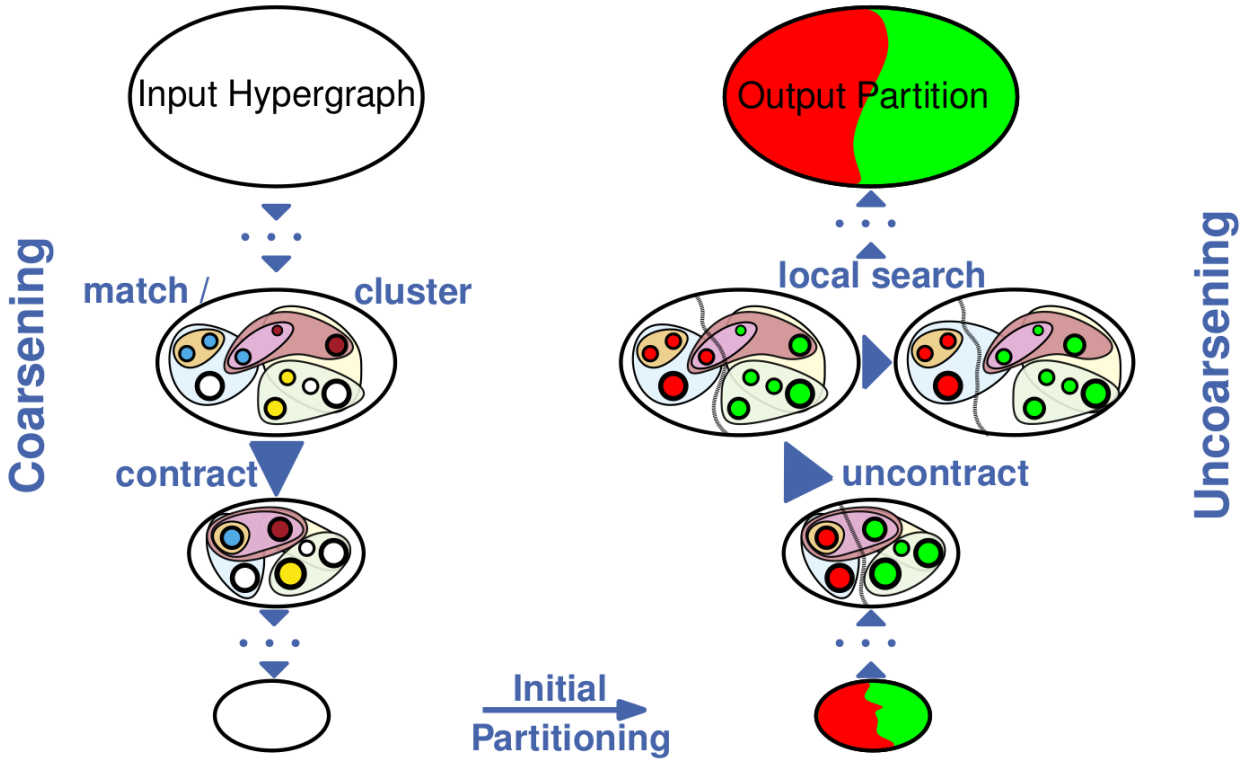



Some in existing systems:
Chaco, (mt)-Metis, Scotch, Jostle, DiBaP, (mt-)KaHIP, KaMinPar, …

# Related Work

## In-Memory HGP

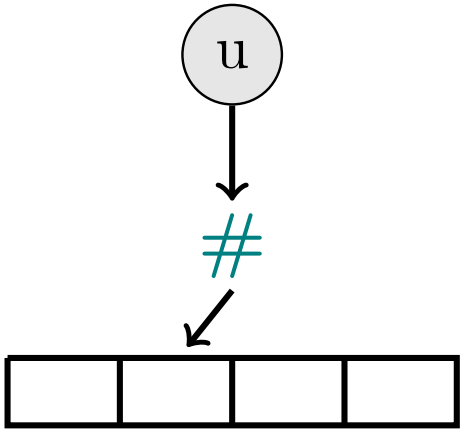
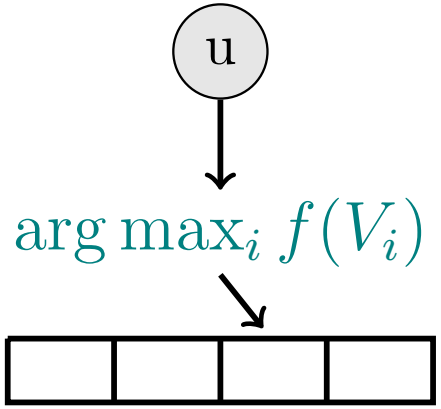
Multilevel Hypergraph Partitioning: THE Traditional Approach




Some in existing systems:
(mt-)KaHyPar, hMetis, PaToH, …

# Related Work

## Streaming GP

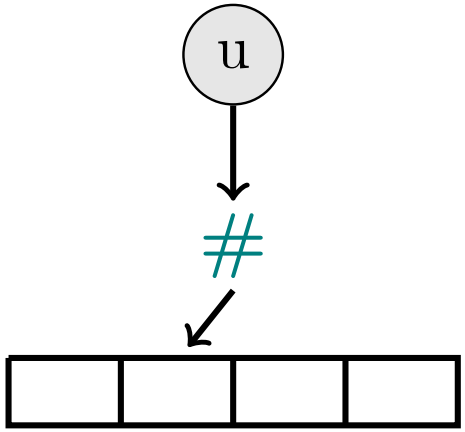
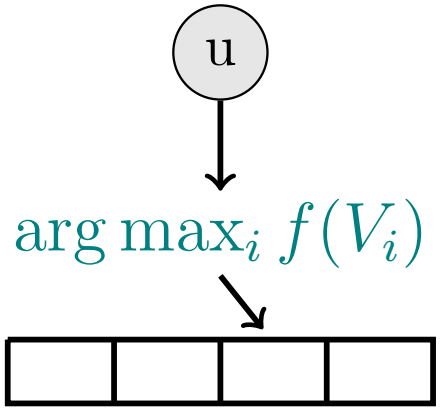
| Algorithm | Hashing | Fennel |
|---|---|---|
| Scheme |   |   |

$$f(V_i) = |V_i \cap N(u)| - \alpha * \gamma * |V_i|^{\gamma-1}$$

Intuition of $f(V_i)$: consider cut and balance on the fly

# Related Work

## Streaming GP

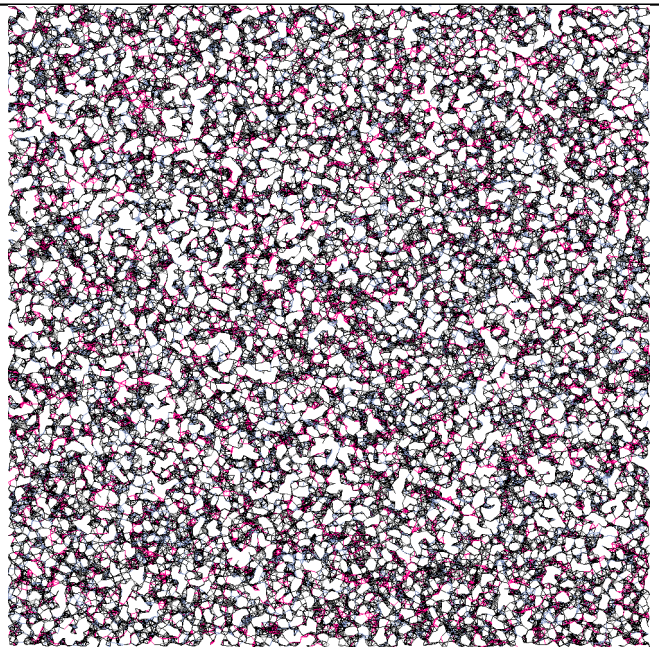
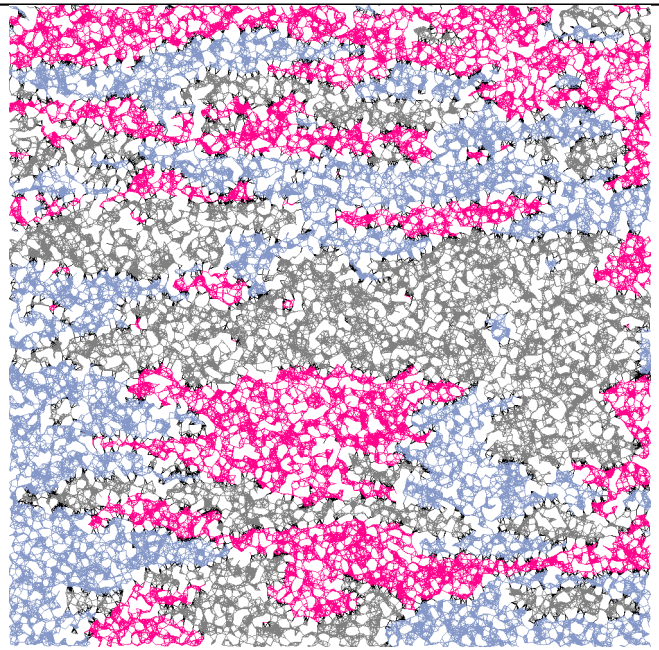
| Algorithm | Hashing | Fennel |
|---|---|---|
| Scheme |   |   |
| Memory | $O(k)$ | $O(n+k)$ |
| Running Time | $O(n)$ | $O(nk+m)$ |

$$f(V_i) = |V_i \cap N(u)| - \alpha * \gamma * |V_i|^{\gamma-1}$$

Intuition of $f(V_i)$: consider cut and balance on the fly

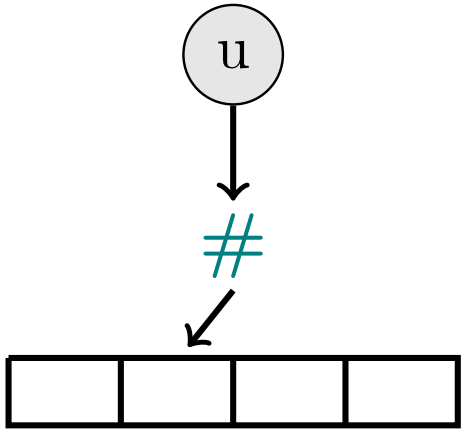
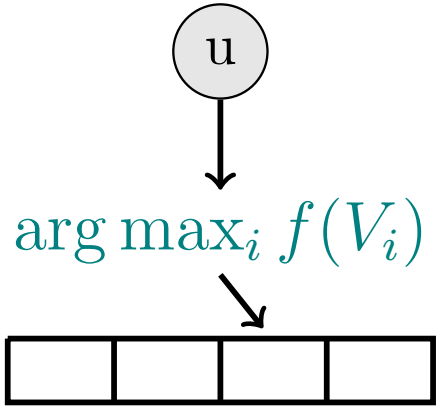
# Related Work

## Streaming GP

| Algorithm | Hashing | Fennel |
|---|---|---|
| Quality |  |  |
| Memory | $O(k)$ | $O(n+k)$ |
| Running Time | $O(n)$ | $O(nk+m)$ |

# Related Work

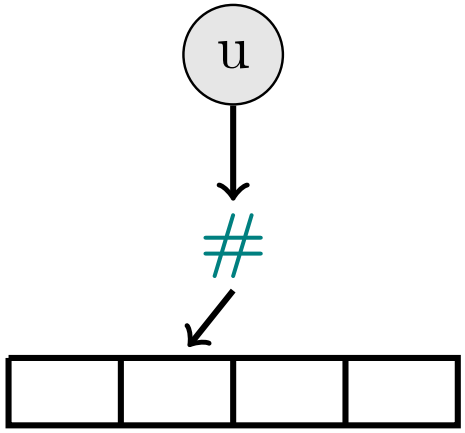
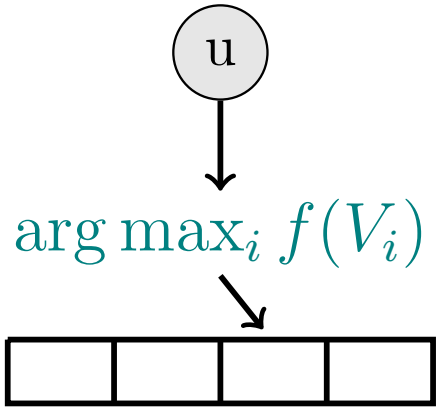
## Streaming HGP

| Algorithm | Hashing | Min-Max |
|---|---|---|
| Scheme |   |   |

$$f(V_i) = \sum_{e \in Nets(u)} |V_i \cap e \neq \emptyset|$$

# Related Work

## Streaming HGP

| Algorithm | Hashing | Min-Max |
|---|---|---|
| Scheme |   |   |
| Memory | $O(k)$ | $O(mk)$ |
| Running Time | $O(n)$ | $O(n + k \sum_{e \in E} \lvert e \rvert)$ |

$$f(V_i) = \sum_{e \in Nets(u)} |V_i \cap e \neq \emptyset|$$

# Related Work

## Streaming HGP

Taşyaran et al. (2021) propose **improved versions** of Min-Max

Faster implementation with same quality: (same complexity)

- Min-Max-N2P: maps nets → blocks rather than blocks → nets

Lightweight versions of Min-Max-N2P w/ worse quality:

- Min-Max-L$\ell$: uses upper-bound $\ell$ for memory consumption per net,
- Min-Max-BF: uses Bloom filters for membership queries
- Min-Max-MH: uses hashing to replace some connectivity info

Buffered streaming algorithms based on Min-Max-N2P:

- REF: buffers every incoming vertex but only reassigns those that may improve connectivity
- REF_RLX: buffers all nodes and reassigns all nodes in the buffer
- REF_RLX_SV: only buffers nodes with small net sets and reassigns all nodes in the buffer.

# Related Work

## (H)GP

You are referred to [1] for extensive references.

[1] Çatalyürek et al. (2023). More Recent Advances in (Hyper)Graph Partitioning. *ACM Computing Surveys*.

# FREIGHT: Fast stREamInG Hypergraph parTitioning

# FREIGHT

## Mathematical Definition

FREIGHT is inspired in Fennel.

- Fennel (for graphs)

$$d = \underset{i,\ |V_i|<L_{\max}}{\operatorname{argmax}} \left\{ |V_i \cap N(v)| - \alpha * \gamma * |V_i|^{\gamma-1} \right\} \tag{1}$$

- FREIGHT (for hypergraphs):

$$d = \underset{i,\ |V_i|<L_{\max}}{\operatorname{argmax}} \left\{ |I^i_{obj}(v)| - \alpha * \gamma * |V_i|^{\gamma-1} \right\} \tag{2}$$

# FREIGHT

## Mathematical Definition

FREIGHT is inspired in Fennel.

- Fennel (for graphs)

$$d = \underset{i,\ |V_i|<L_{\max}}{\operatorname{argmax}} \left\{ |V_i \cap N(v)| - \alpha * \gamma * |V_i|^{\gamma-1} \right\} \tag{1}$$

- FREIGHT (for hypergraphs):

$$d = \underset{i,\ |V_i|<L_{\max}}{\operatorname{argmax}} \left\{ |I^i_{obj}(v)| - \alpha * \gamma * |V_i|^{\gamma-1} \right\} \tag{2}$$

Factor to penalize block imbalance is identical

# FREIGHT

## Mathematical Definition

FREIGHT is inspired in Fennel.

- Fennel (for graphs)

$$d = \underset{i,\ |V_i|<L_{\max}}{\operatorname{argmax}} \left\{ |V_i \cap N(v)| - \alpha * \gamma * |V_i|^{\gamma-1} \right\} \tag{1}$$

- FREIGHT (for hypergraphs):

$$d = \underset{i,\ |V_i|<L_{\max}}{\operatorname{argmax}} \left\{ |I^i_{obj}(v)| - \alpha * \gamma * |V_i|^{\gamma-1} \right\} \tag{2}$$

Factor to penalize block imbalance is identical

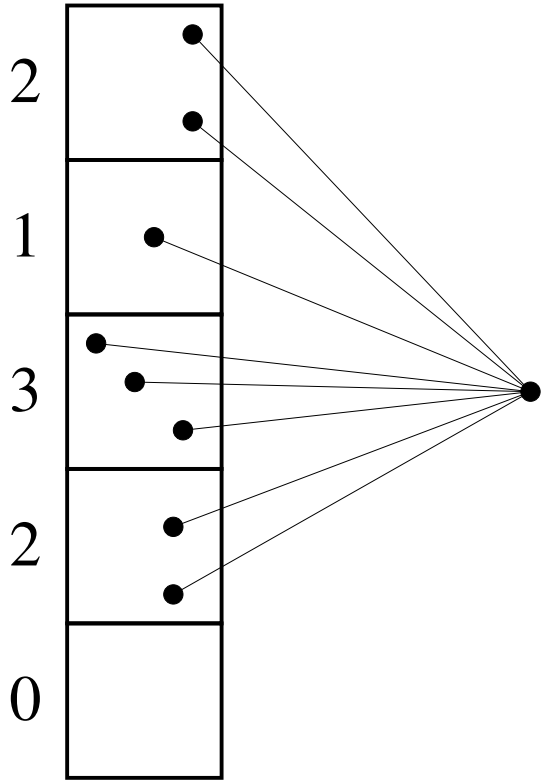
Factor to minimize edge-cut is adapted to minimize the intended metric: cut-net or connectivity.

# FREIGHT

## Mathematical Definition

- Fennel (for graphs):

$$d = \underset{i,\ |V_i|<L_{\max}}{\operatorname{argmax}} \left\{ |V_i \cap N(v)| - \alpha * \gamma * |V_i|^{\gamma-1} \right\} \tag{1}$$



# FREIGHT

## Mathematical Definition

- FREIGHT (for hypergraphs):

$$d = \underset{i,\ |V_i| < L_{\max}}{\operatorname{argmax}} \left\{ |I^i_{obj}(v)| - \alpha * \gamma * |V_i|^{\gamma - 1} \right\} \tag{2}$$

Keep $d_e$: block to which latest pin of net $e$ was assigned.

Connectivity: $I^i_{obj}(v) = I^i_{con}(v) = \{e \in Nets(v) : d_e = i\}$

Cut-Net: $I^i_{obj}(v) = I^i_{cut}(v) = I^i_{con}(v) \setminus E'$

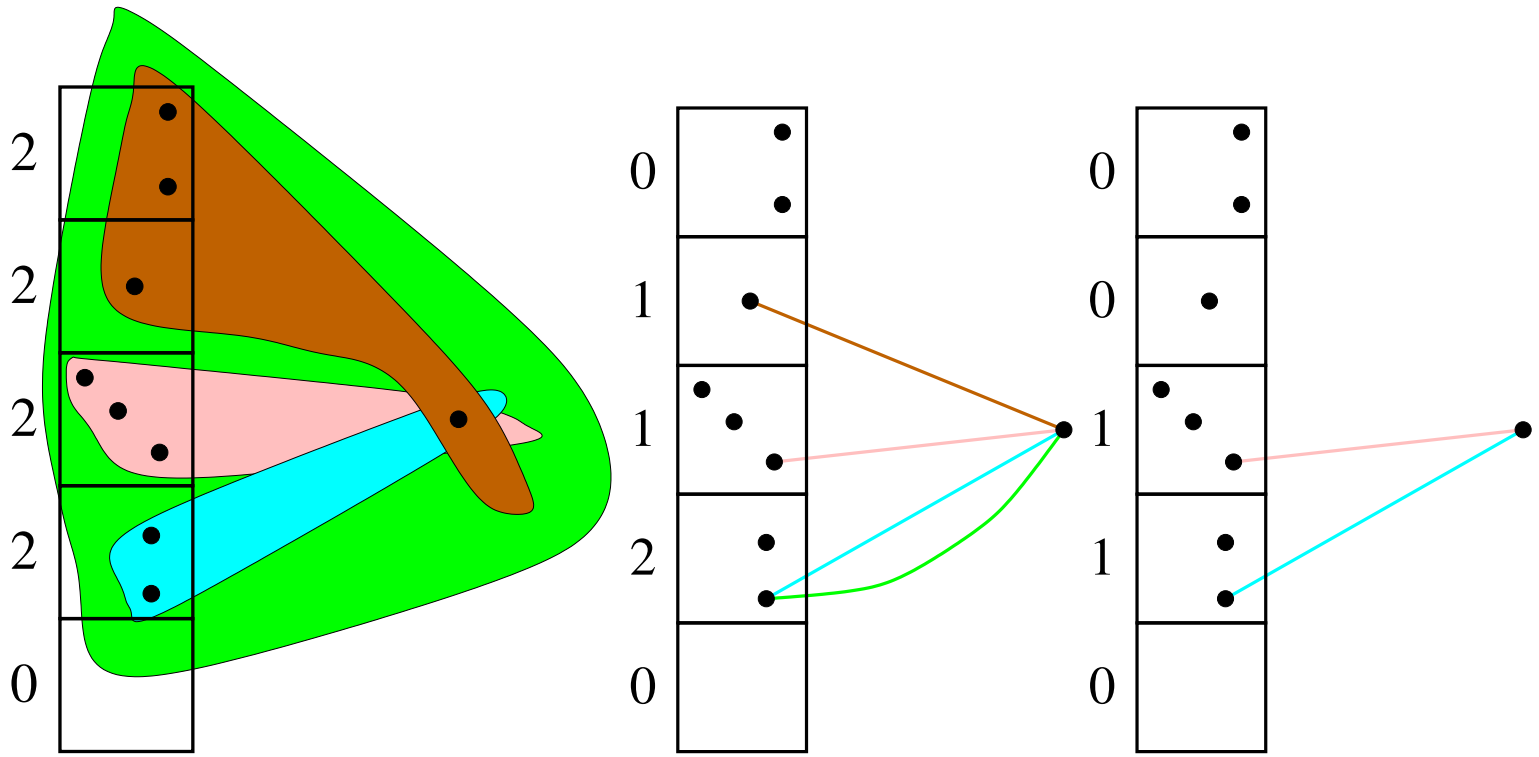
($E'$: set of already cut nets)

# FREIGHT

## Mathematical Definition

- FREIGHT (for hypergraphs):

$$d = \underset{i,\ |V_i|<L_{\max}}{\operatorname{argmax}} \left\{ |I^i_{obj}(v)| - \alpha * \gamma * |V_i|^{\gamma-1} \right\} \tag{2}$$



# FREIGHT

## Memory Usage

- FREIGHT (for hypergraphs):

$$d = \underset{i,\ |V_i| < L_{\max}}{\operatorname{argmax}} \left\{ |I^i_{obj}(v)| - \alpha * \gamma * |V_i|^{\gamma-1} \right\} \tag{2}$$

FREIGHT uses $O(m+k)$ memory.

- $O(k)$ to keep block cardinalities $|V_i|$, $1 \leq i \leq k$
- $O(m)$ to keep one assignment $d_e$ per edge.

This is more memory than streaming GP.

- $O(n+k)$ used by the state-of-the-art.

This is less memory than streaming HGP.

- $O(mk)$ used by the state-of-the-art.

# FREIGHT

## Efficient Implementation

- FREIGHT (for hypergraphs):

$$d = \underset{i,\ |V_i|<L_{\max}}{\operatorname{argmax}} \left\{ |I^i_{obj}(v)| - \alpha * \gamma * |V_i|^{\gamma-1} \right\} \tag{2}$$

For every loaded vertex $v$, FREIGHT splits Equation (2) by separating blocks $V_i$ for which $|V_i| < L_{\max}$ into two sets: $S_1$ and $S_2$.

- Set $S_1$: Blocks $V_i$ where $|I^i_{obj}(v)| > 0$.
- Set $S_2$: Blocks $V_i$ where $|I^i_{obj}(v)| = 0$.

$$d = \underset{i \in S_1}{\operatorname{argmax}} \left\{ |I^i_{obj}(v)| - \alpha * \gamma * |V_i|^{\gamma-1} \right\} \tag{3}$$

$$d = \underset{i \in S_2}{\operatorname{argmax}} \left\{ |I^i_{obj}(v)| - \alpha * \gamma * |V_i|^{\gamma-1} \right\} = \underset{i \in S_2}{\operatorname{argmin}} |V_i| \tag{4}$$

# FREIGHT

## Efficient Implementation

$$d = \underset{i \in S_1}{\operatorname{argmax}} \left\{ |I^i_{obj}(v)| - \alpha * \gamma * |V_i|^{\gamma-1} \right\} \tag{3}$$

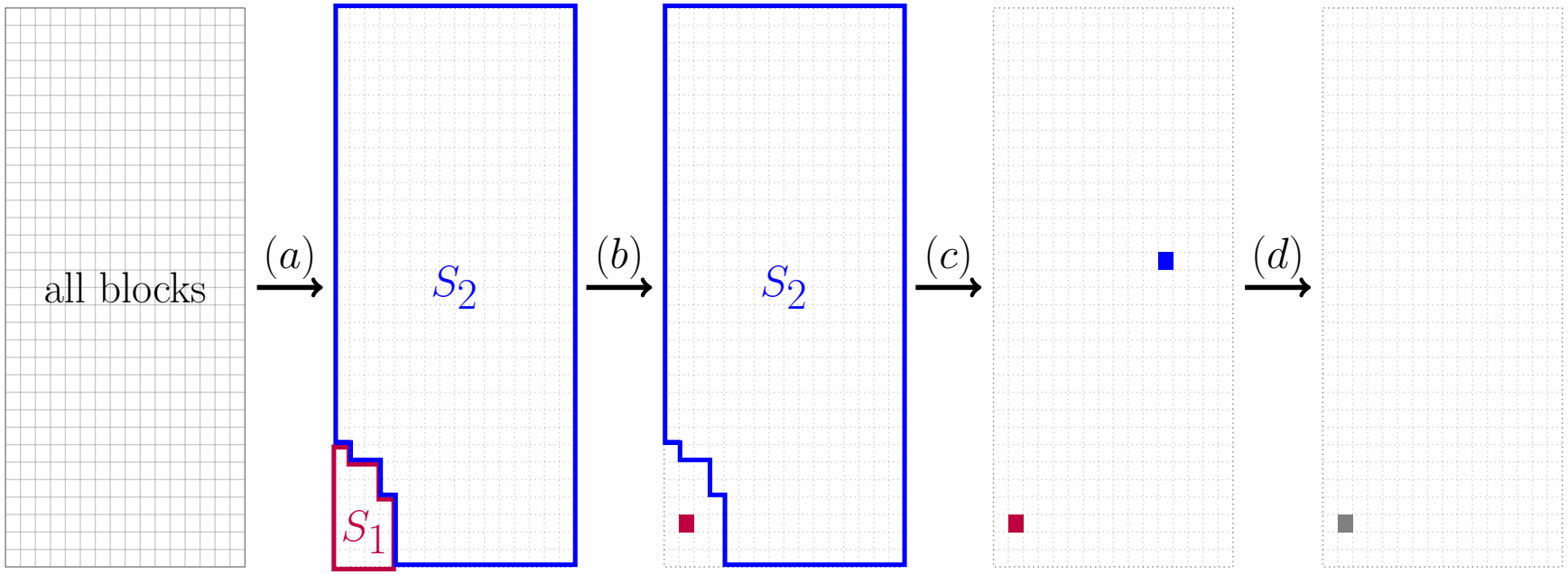
$$d = \underset{i \in S_2}{\operatorname{argmax}} \left\{ |I^i_{obj}(v)| - \alpha * \gamma * |V_i|^{\gamma-1} \right\} = \underset{i \in S_2}{\operatorname{argmin}} |V_i| \tag{4}$$

# FREIGHT

## Efficient Implementation

$$d = \underset{i \in S_1}{\operatorname{argmax}} \left\{ |I^i_{obj}(v)| - \alpha * \gamma * |V_i|^{\gamma-1} \right\} \tag{3}$$

$$d = \underset{i \in S_2}{\operatorname{argmax}} \left\{ |I^i_{obj}(v)| - \alpha * \gamma * |V_i|^{\gamma-1} \right\} = \underset{i \in S_2}{\operatorname{argmin}} |V_i| \tag{4}$$



# FREIGHT

## Efficient Implementation

$$d = \operatorname*{argmax}_{i \in S_1} \left\{ |I^i_{obj}(v)| - \alpha * \gamma * |V_i|^{\gamma-1} \right\} \tag{3}$$

$$d = \operatorname*{argmax}_{i \in S_2} \left\{ |I^i_{obj}(v)| - \alpha * \gamma * |V_i|^{\gamma-1} \right\} = \operatorname*{argmin}_{i \in S_2} |V_i| \tag{4}$$

**Theorem**

*Equation (3) can be solved in time $O(|I(v)|)$ ($|I(v)|$ is the degree of $v$).*

**Proof.**

- The terms $|I^i_{obj}(v)|$ in Equation (3) can be computed by iterating through the nets of $v$ at a cost of $O(|I(v)|)$ and determining their status as cut, unassigned, or assigned to a block.
- The calculation of the factors $-\alpha * \gamma * |V_i|^{\gamma-1}$ in Equation (3) can be done in time $O(|S_1|) = O(|I(v)|)$, thus completing the proof.

□

# FREIGHT

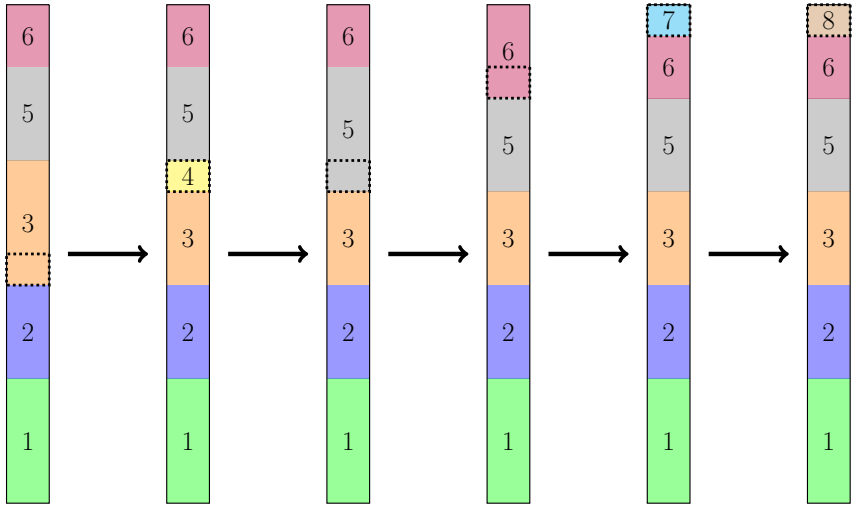
## Efficient Implementation

$$d = \underset{i \in S_1}{\operatorname{argmax}} \left\{ |I^i_{obj}(v)| - \alpha * \gamma * |V_i|^{\gamma-1} \right\} \tag{3}$$

$$d = \underset{i \in S_2}{\operatorname{argmax}} \left\{ |I^i_{obj}(v)| - \alpha * \gamma * |V_i|^{\gamma-1} \right\} = \underset{i \in S_2}{\operatorname{argmin}} |V_i| \tag{4}$$

### Theorem

*Our proposed data structure keeps the blocks sorted in ascending order of cardinality at cost $O(1)$ per vertex.*



# FREIGHT

## Efficient Implementation

$$d = \underset{i \in S_1}{\operatorname{argmax}} \left\{ |I^i_{obj}(v)| - \alpha * \gamma * |V_i|^{\gamma-1} \right\} \tag{3}$$

$$d = \underset{i \in S_2}{\operatorname{argmax}} \left\{ |I^i_{obj}(v)| - \alpha * \gamma * |V_i|^{\gamma-1} \right\} = \underset{i \in S_2}{\operatorname{argmin}} |V_i| \tag{4}$$

### Theorem

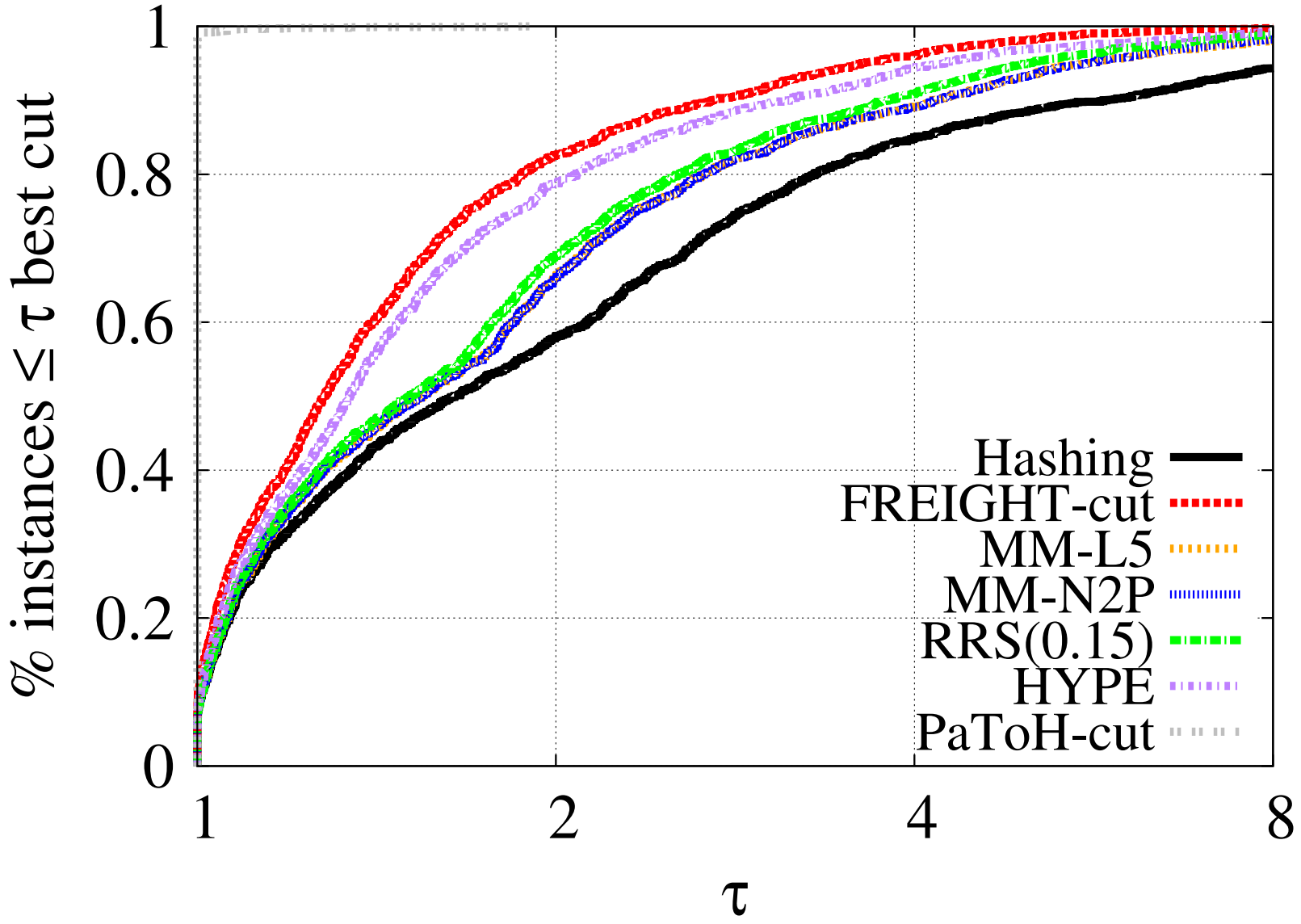
*By utilizing our proposed data structure, solving Equation (4) or demonstrating that any solution for Equation (3) is also a solution for Equation (2) can be accomplished in $O(1)$ time.*

### Corollary

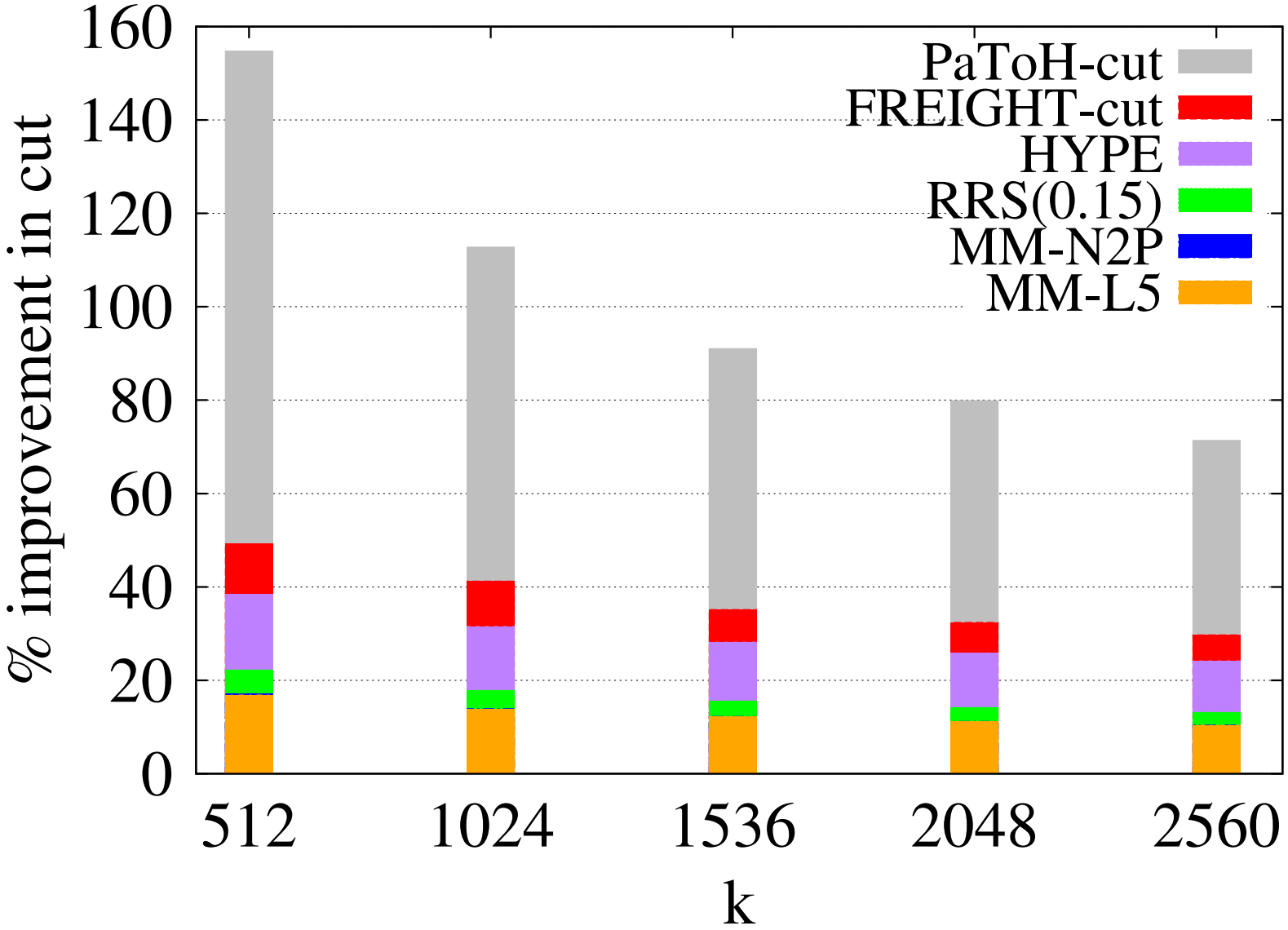
*The overall complexity of* `FREIGHT` *is* $O\left(\sum_{e \in E} |e| + n\right)$.

# Experimental Results

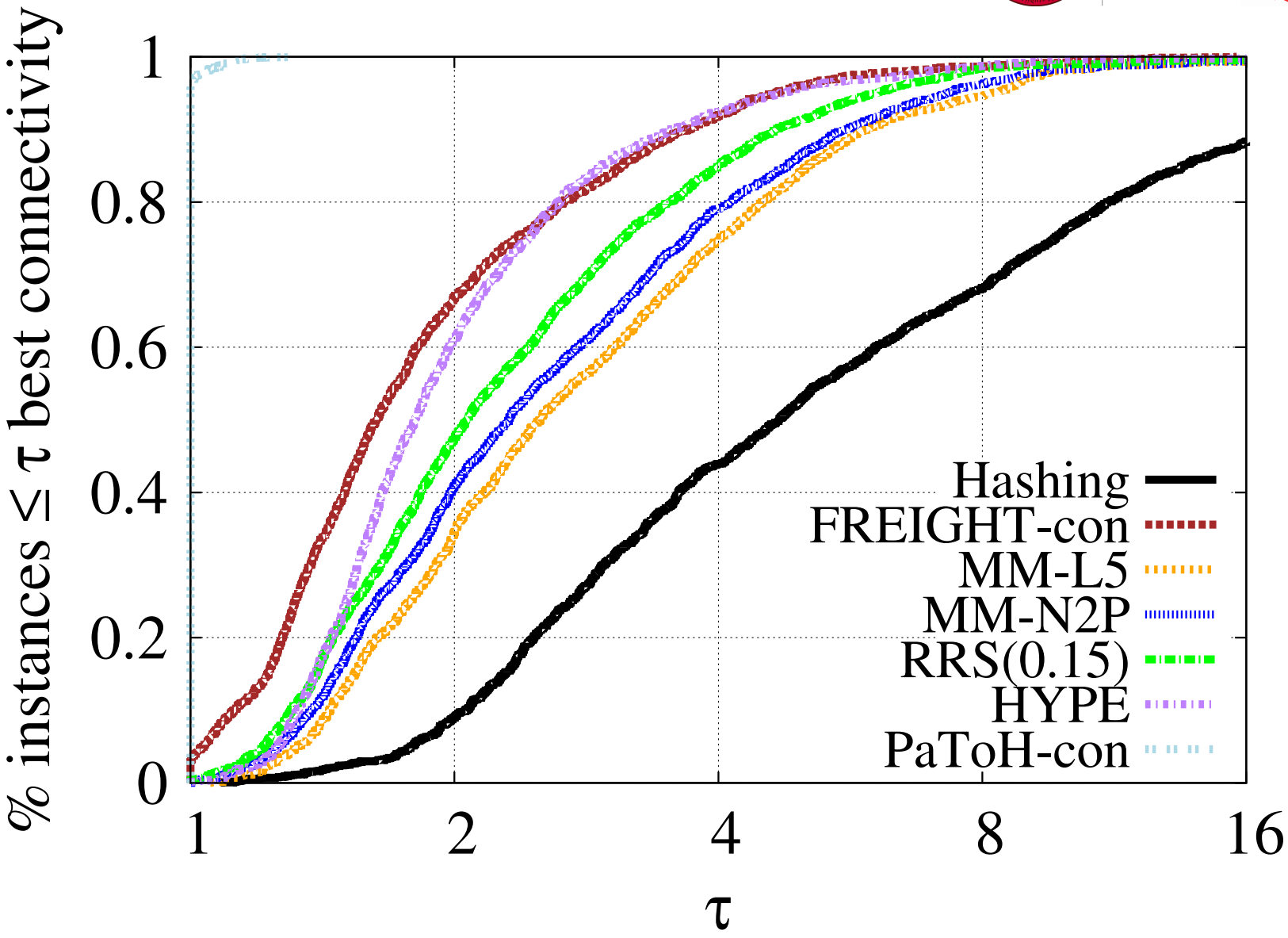
# Experimental Results



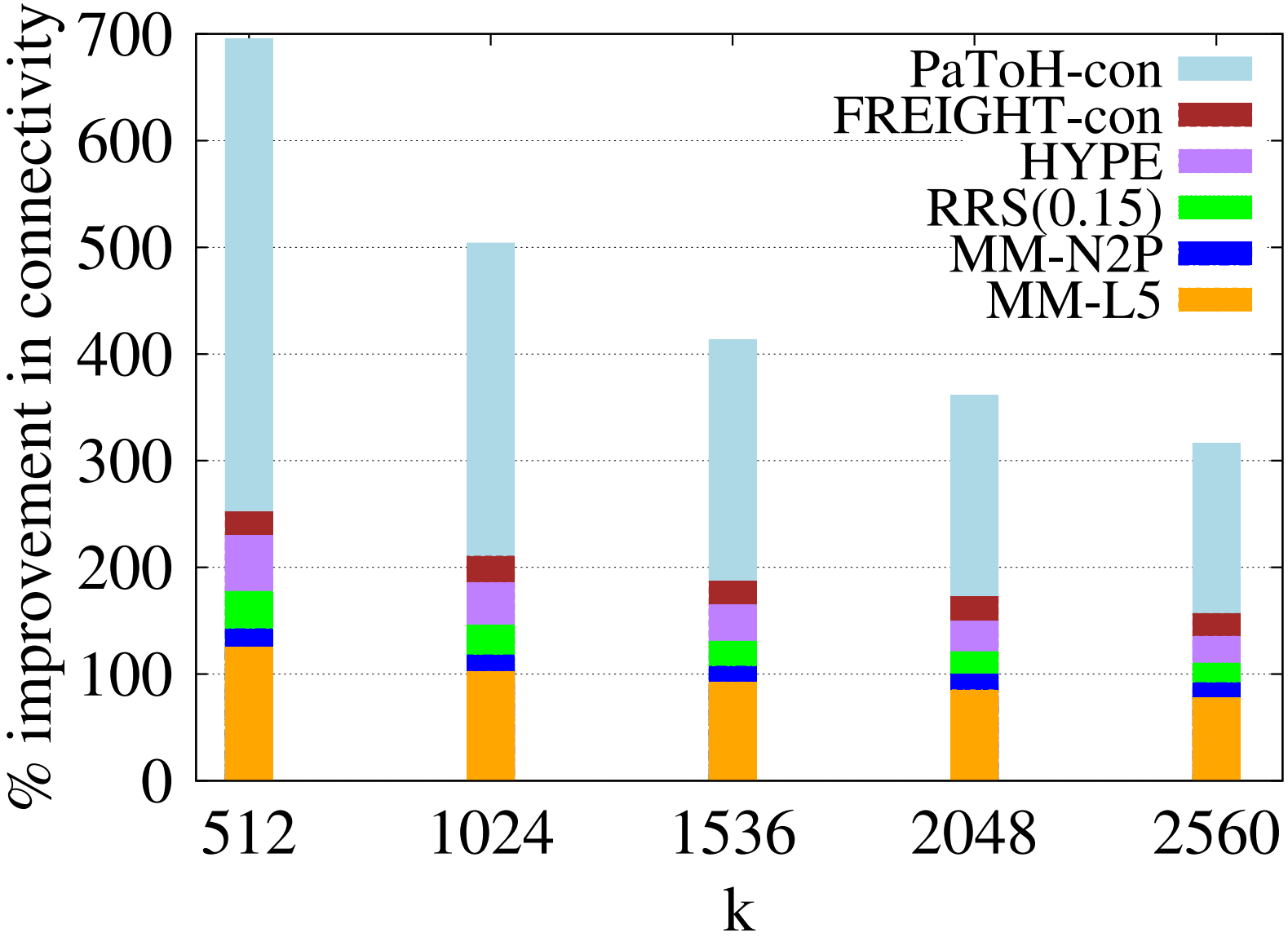
# Experimental Results



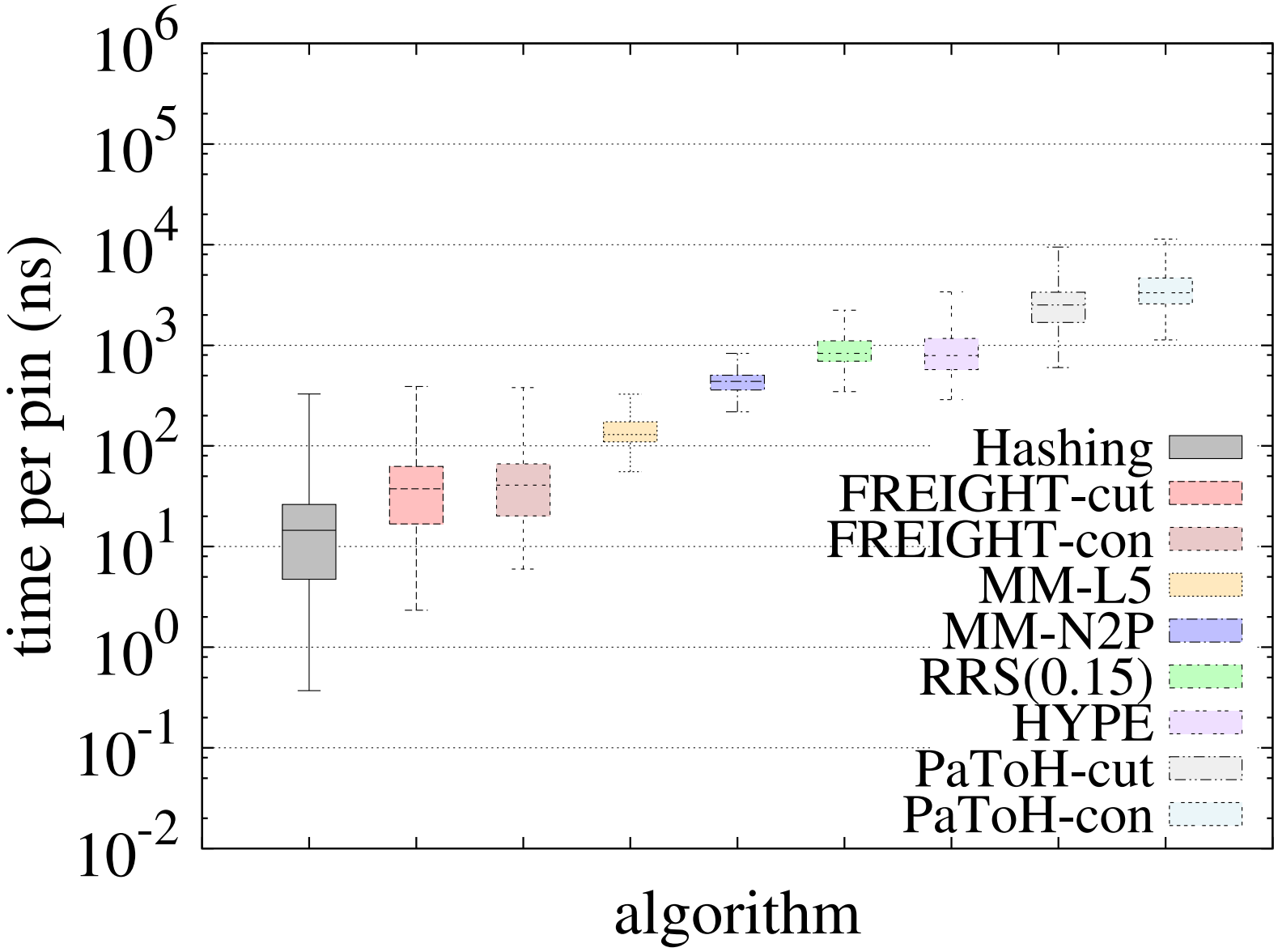
# Experimental Results



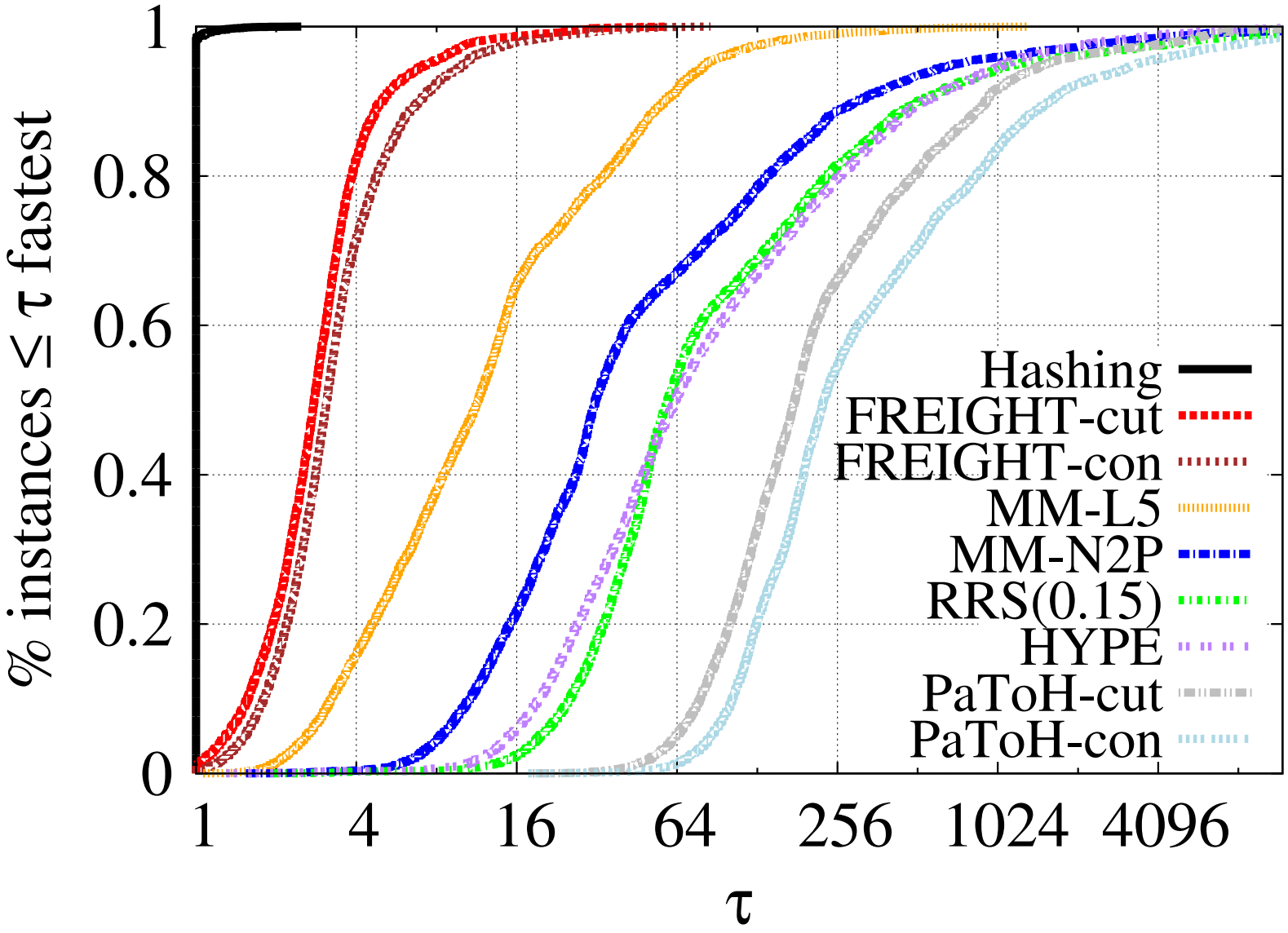
# Experimental Results



# Experimental Results



# Experimental Results



# Conclusion

# FREIGHT

## Conclusion

It is highly efficient and effective in theory and in practice.

It utilizes optimized data structure
$\Rightarrow$ linear running time w.r.t. pin-count.

Linear memory consumption w.r.t. the numbers of nets and blocks.

Empirically competitive with the speed of `Hashing`
$\Rightarrow$ at most four times difference in runtime for 80% of instances

Outperforms all existing streaming algorithms and even the in-memory algorithm `HYPE` in both cut-net and connectivity.

**We just released FREIGHT [MIT License]!**
`https://github.com/KaHIP/FREIGHT`

# FREIGHT: Fast Streaming Hypergraph Partitioning

Kamal Eyubov, Heidelberg University
**Marcelo Fonseca Faraj, Heidelberg University**
Christian Schulz, Heidelberg University