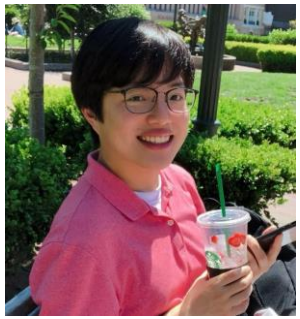
KAIST AI
Kim Jaechul Graduate School

Finding a Concise, Precise, and Exhaustive Set of Near Bi-Cliques in Dynamic Graphs

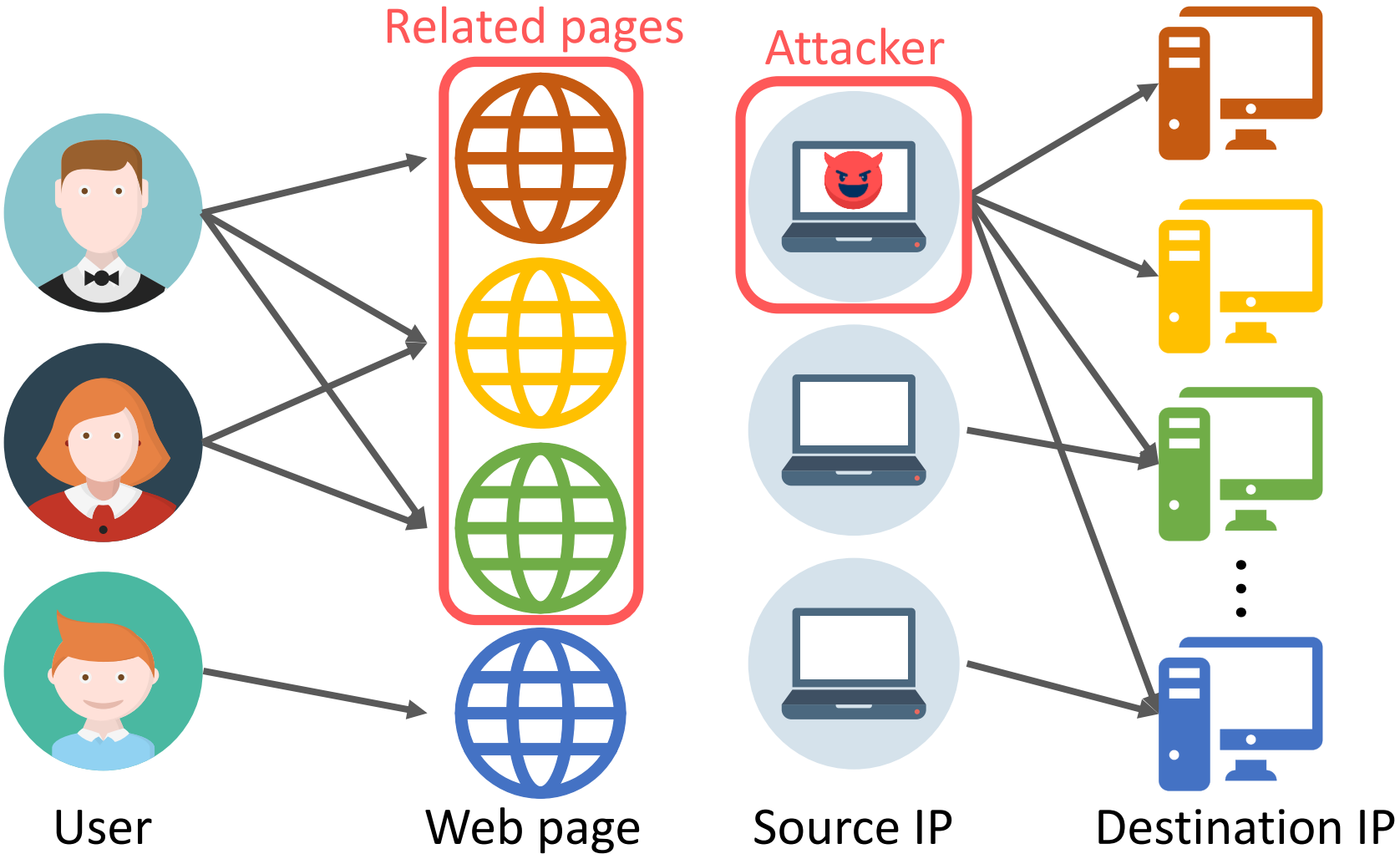
Hyeonjeong Shin[1], Taehyung Kwon[1], Neil Shah[2], Kijung Shin[1]

[1] KAIST, [2] Snap Inc.



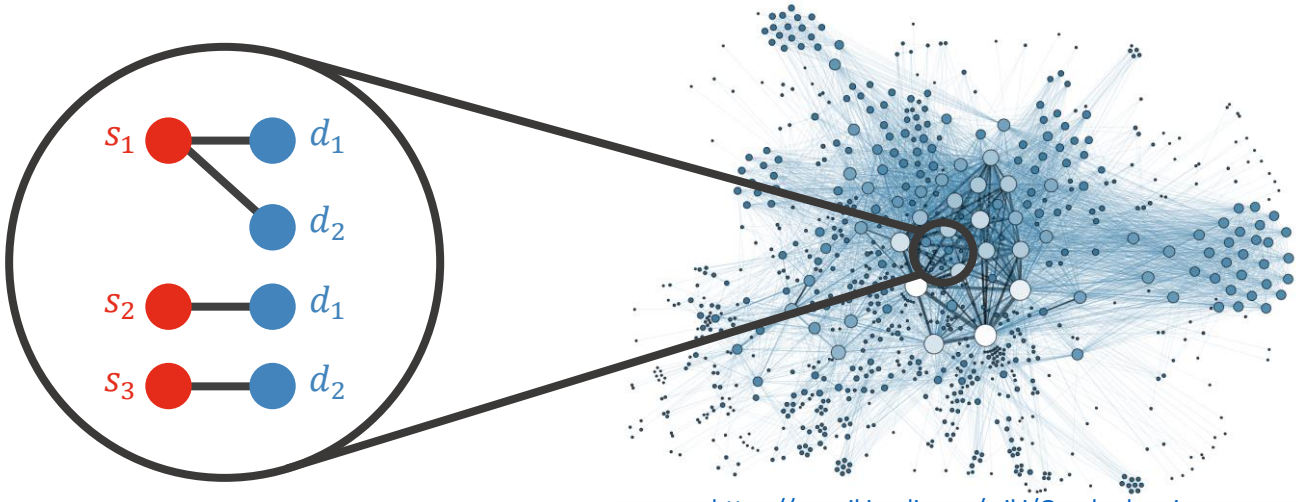



Finding (Near) Bi-Clique Problem



Problem: Too Many (Near) Bi-Cliques

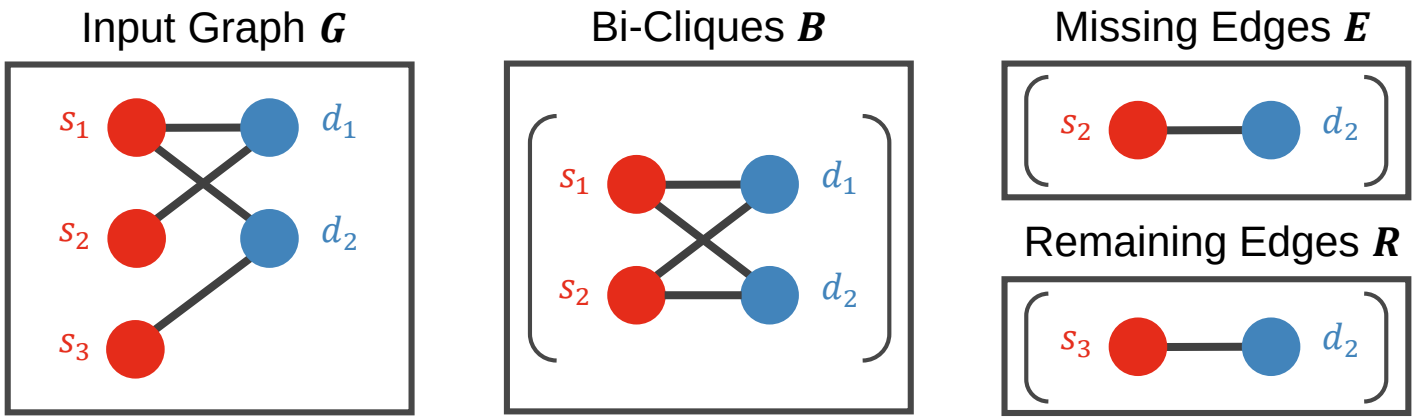
- The number of (near) bi-cliques is exponential in the number of nodes
  - Practically impossible to find all
- Some are too small or highly overlapped
- What is a "good" set of near bi-cliques?



https://en.wikipedia.org/wiki/Graph_drawing

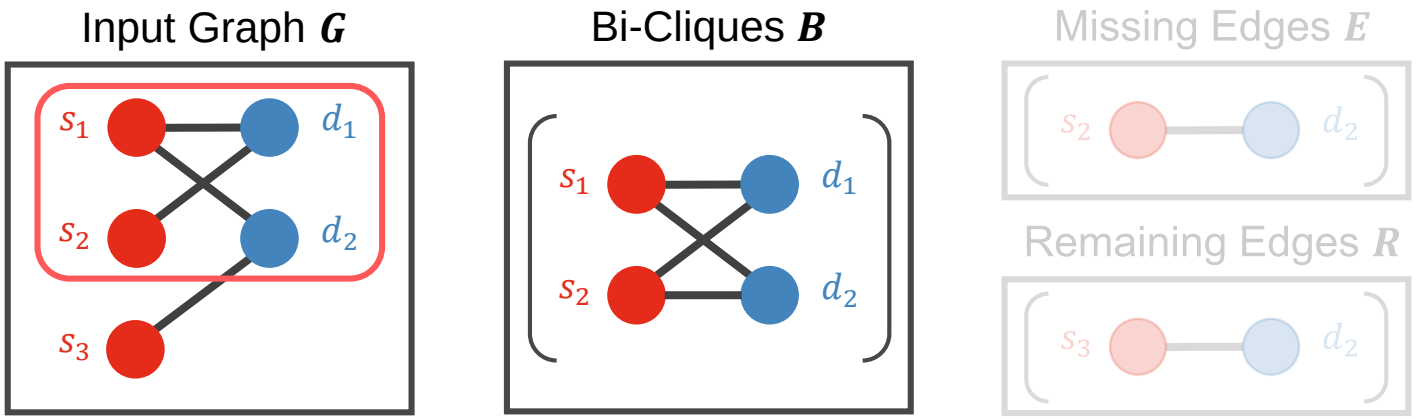
Terminologies

- Exact bi-cliques $\boldsymbol{B}$
- Missing edges $\boldsymbol{M} \coloneqq \{\mathbf{e} \notin \boldsymbol{G} \mid \exists \boldsymbol{b} \in \boldsymbol{B}\ \mathbf{s.t.}\ \mathbf{e} \in \boldsymbol{b}\}$
- Remaining edges $\boldsymbol{R} \coloneqq \{\mathbf{e} \in \boldsymbol{G} \mid \forall \boldsymbol{b} \in \boldsymbol{B}, \mathbf{e} \notin \boldsymbol{b}\}$



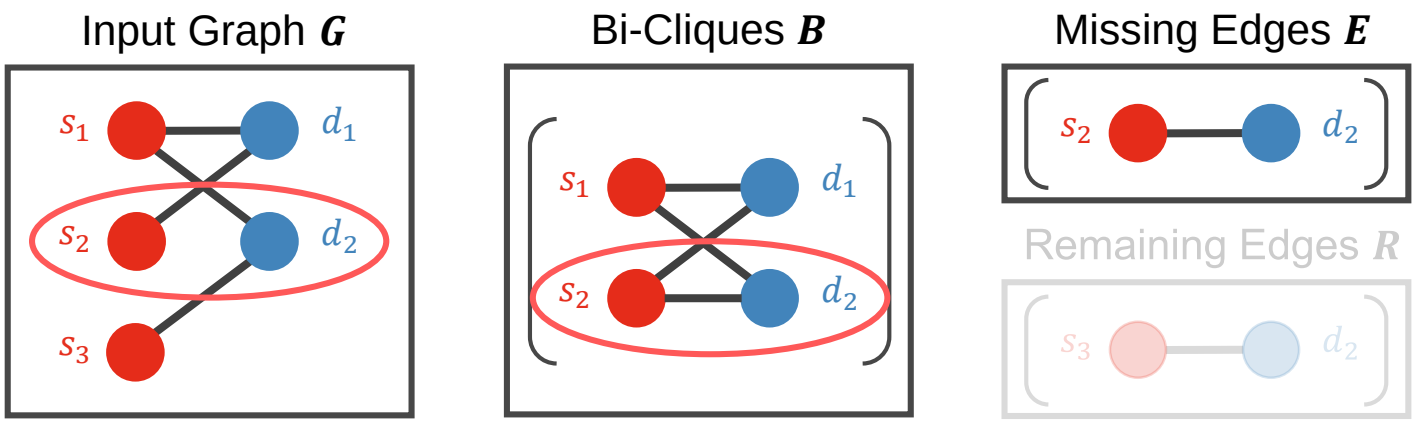
Terminologies

- Exact bi-cliques $\boldsymbol{B}$
- Missing edges $\boldsymbol{M} \coloneqq \{\mathbf{e} \notin \boldsymbol{G} \mid \exists \boldsymbol{b} \in \boldsymbol{B}\ \mathbf{s.t.}\ \mathbf{e} \in \boldsymbol{b}\}$
- Remaining edges $\boldsymbol{R} \coloneqq \{\mathbf{e} \in \boldsymbol{G} \mid \forall \boldsymbol{b} \in \boldsymbol{B}, \mathbf{e} \notin \boldsymbol{b}\}$



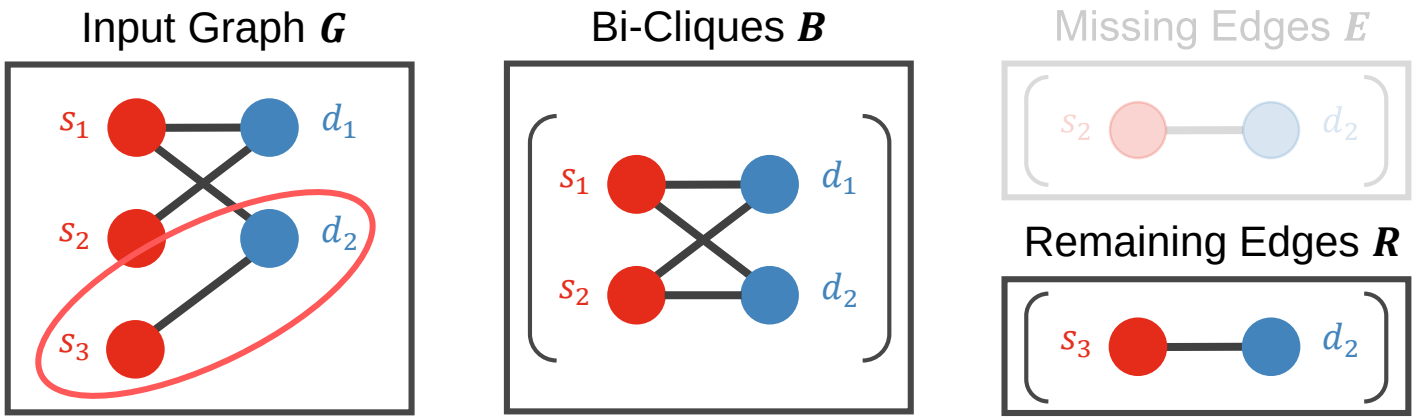
Terminologies

- Exact bi-cliques $\boldsymbol{B}$
- Missing edges $\boldsymbol{M} \coloneqq \{\mathbf{e} \notin \boldsymbol{G} \mid \exists \boldsymbol{b} \in \boldsymbol{B}\ \mathbf{s.t.}\ \mathbf{e} \in \boldsymbol{b}\}$
- Remaining edges $\boldsymbol{R} \coloneqq \{\mathbf{e} \in \boldsymbol{G} \mid \forall \boldsymbol{b} \in \boldsymbol{B}, \mathbf{e} \notin \boldsymbol{b}\}$



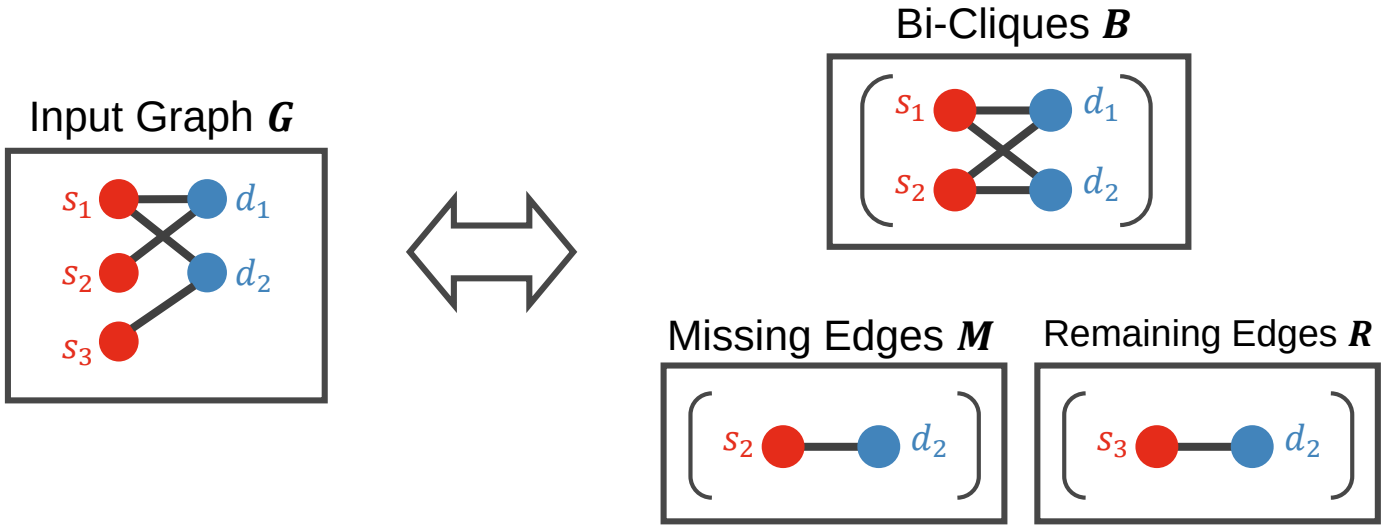
Terminologies

- Exact bi-cliques $\boldsymbol{B}$
- Missing edges $\boldsymbol{M} \coloneqq \{\mathbf{e} \notin \boldsymbol{G} \mid \exists \boldsymbol{b} \in \boldsymbol{B}\ \mathbf{s.t.}\ \mathbf{e} \in \boldsymbol{b}\}$
- Remaining edges $\boldsymbol{R} \coloneqq \{\mathbf{e} \in \boldsymbol{G} \mid \forall \boldsymbol{b} \in \boldsymbol{B}, \mathbf{e} \notin \boldsymbol{b}\}$



Problem Definition

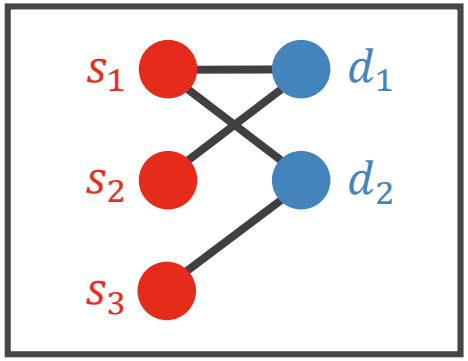
- A dynamic graph can be described using $\boldsymbol{B}$, $\boldsymbol{M}$ and $\boldsymbol{R}$
  - By removing $\mathbf{e} \in \boldsymbol{M}$ from $\boldsymbol{B}$ and adding $\mathbf{e} \in \boldsymbol{R}$ into $\boldsymbol{B}$



Solution: Find "Good" Near Bi-Cliques

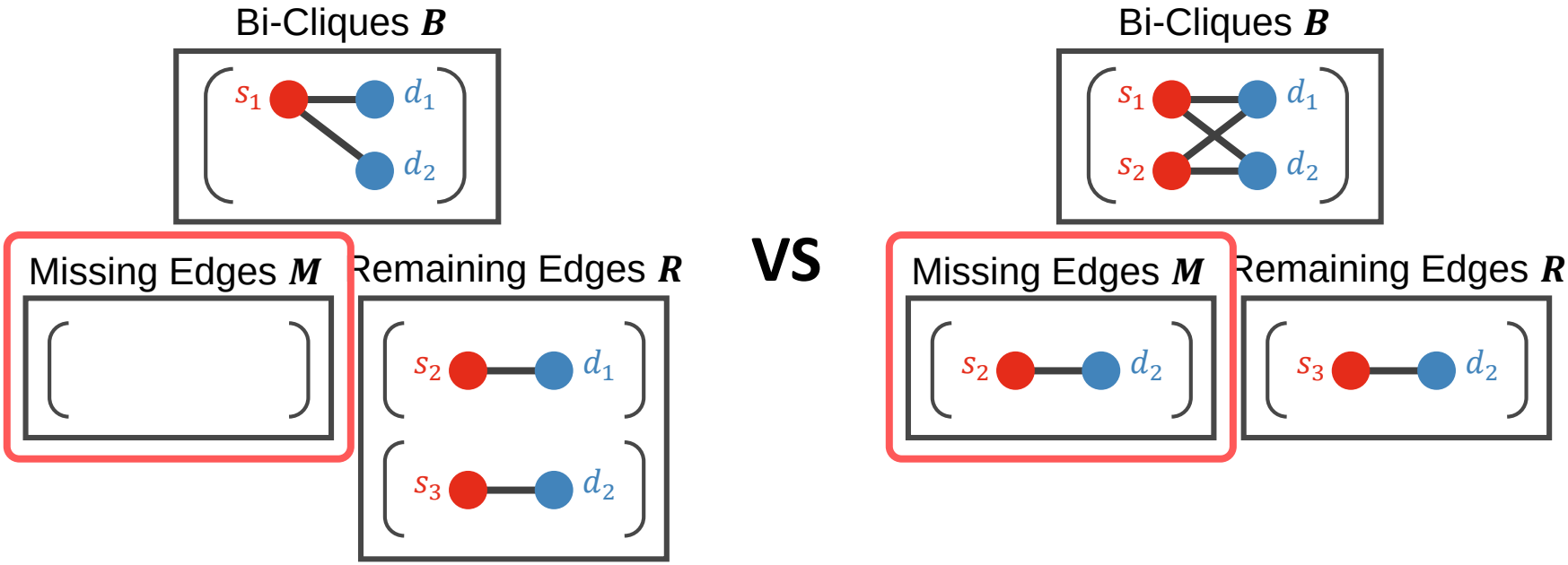
- What are the rules for a "good" set of near bi-cliques?
- Let's consider the below graph $\boldsymbol{G}$

Input Graph $\boldsymbol{G}$



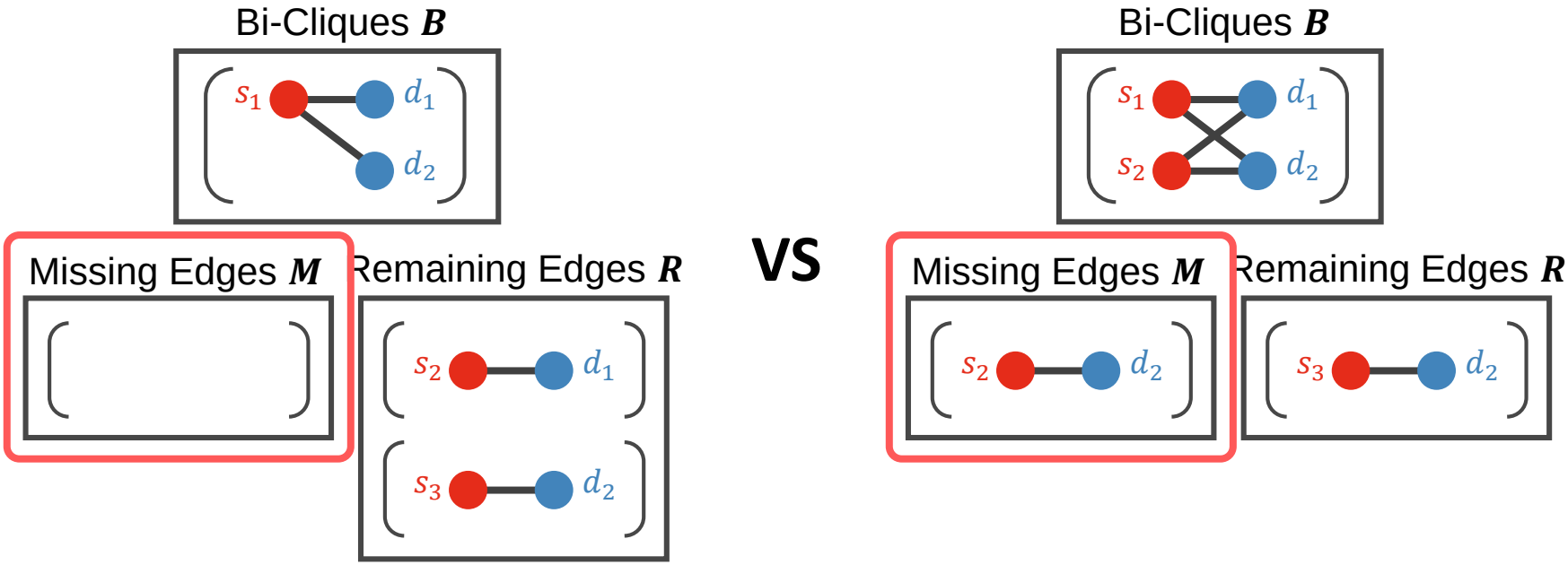
Solution: Find "Good" Near Bi-Cliques

- The left one has less missing edges $\boldsymbol{M}$
  - i.e., less errors inside B



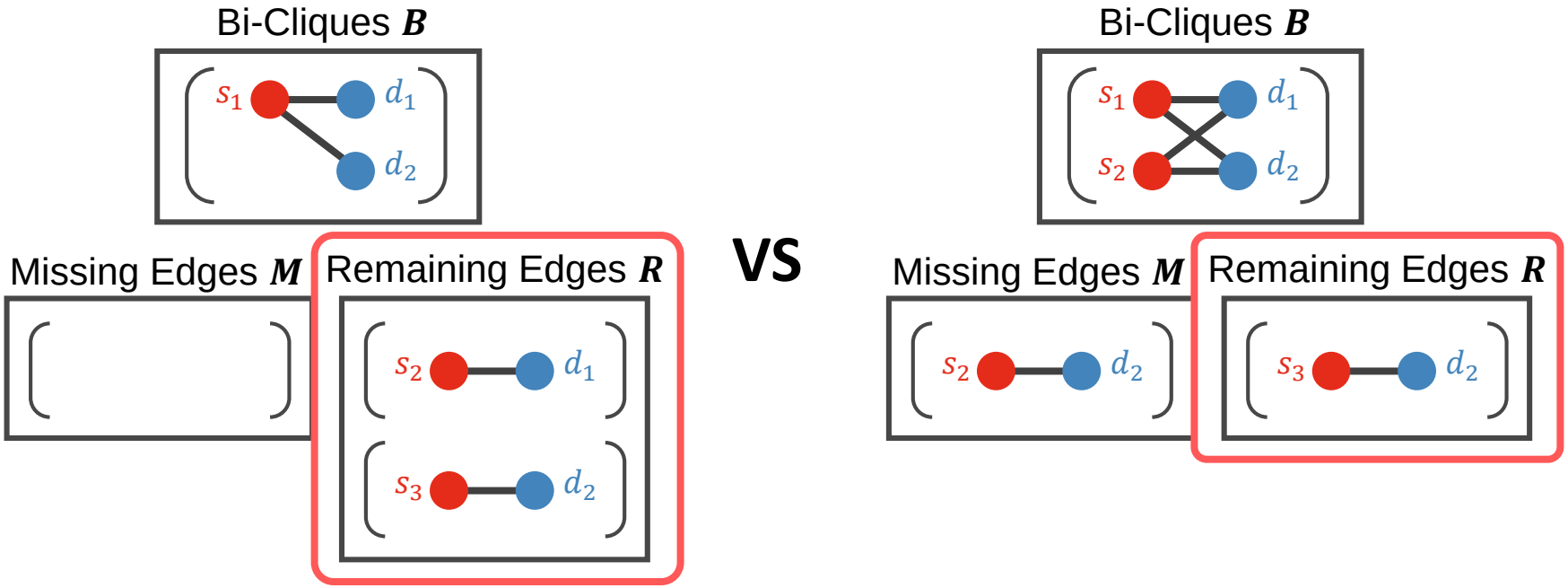
Solution: Find "Good" Near Bi-Cliques

- Rule 1) **Preciseness**: Near bi-cliques should be close to exact bi-cliques while minimizing # of missing edges



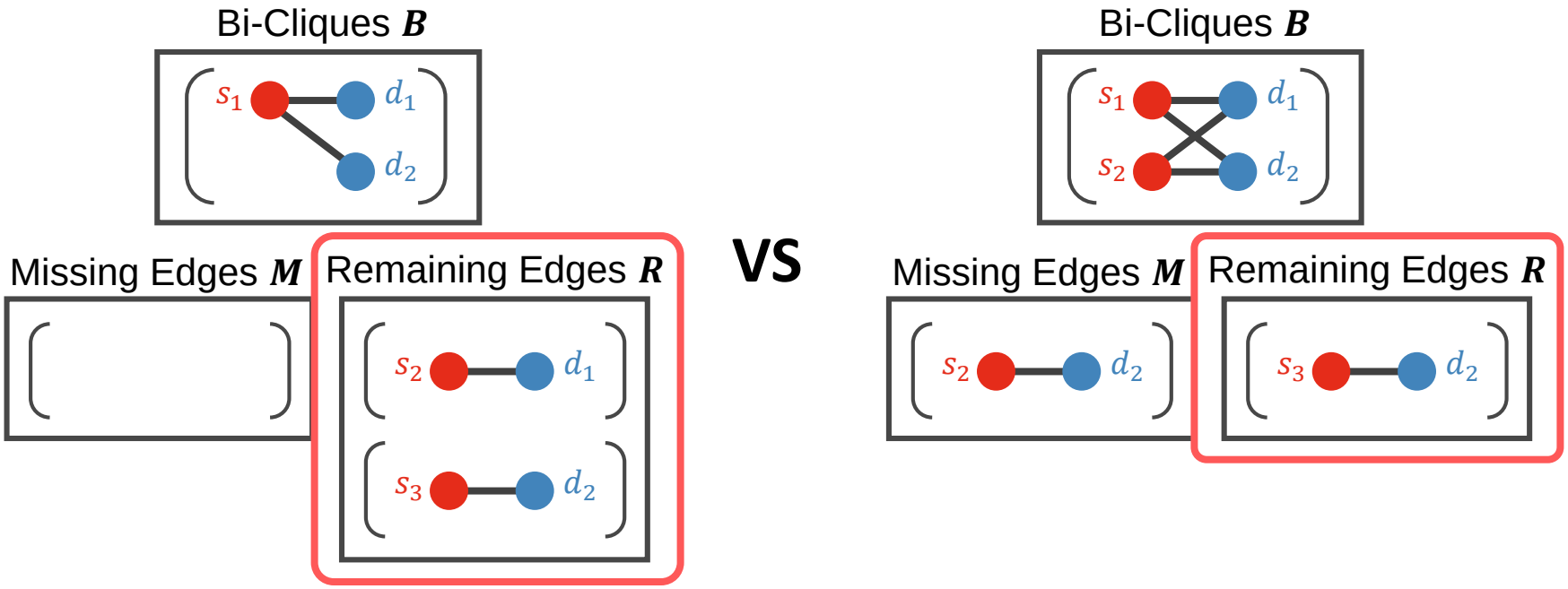
Solution: Find "Good" Near Bi-Cliques

- The right $\boldsymbol{B}$ has less remaining edges, $\boldsymbol{R}$
  - i.e., covers more edges in $\boldsymbol{G}$



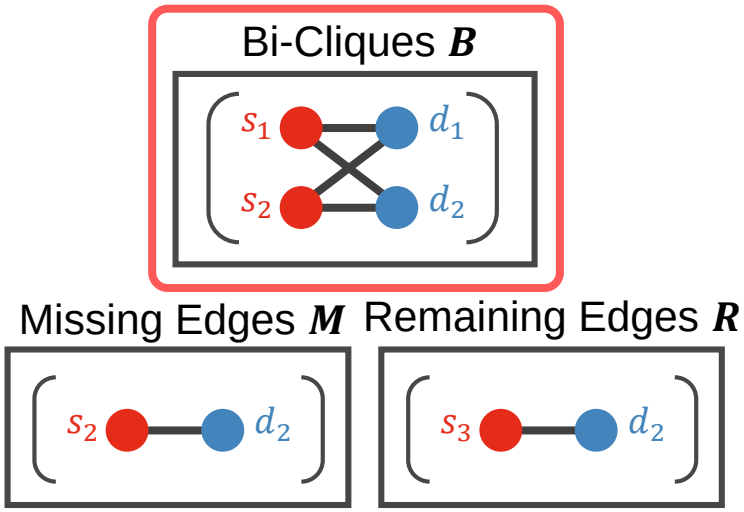
Solution: Find "Good" Near Bi-Cliques

- Rule 2) **Exhaustiveness**: Near bi-cliques should cover a large portion of the given graph $\boldsymbol{G}$

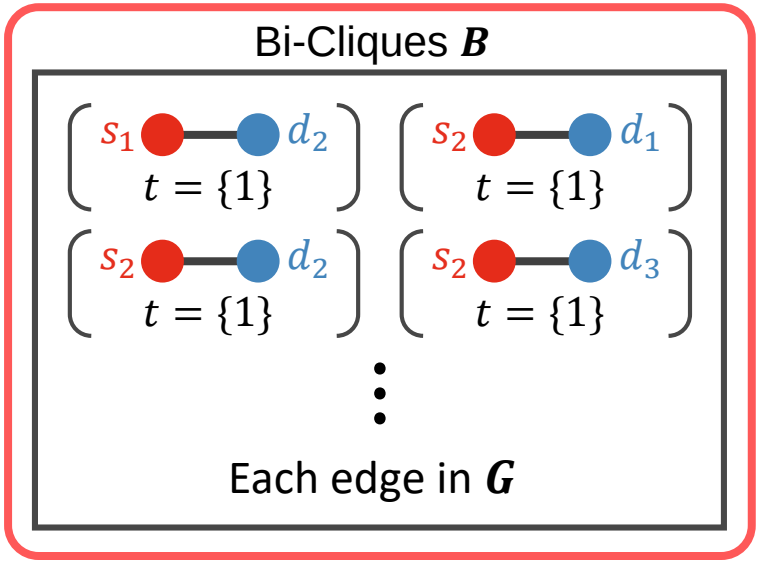


Solution: Find "Good" Near Bi-Cliques

- Rule 3) **Conciseness**: A larger near bi-clique is better than smaller one



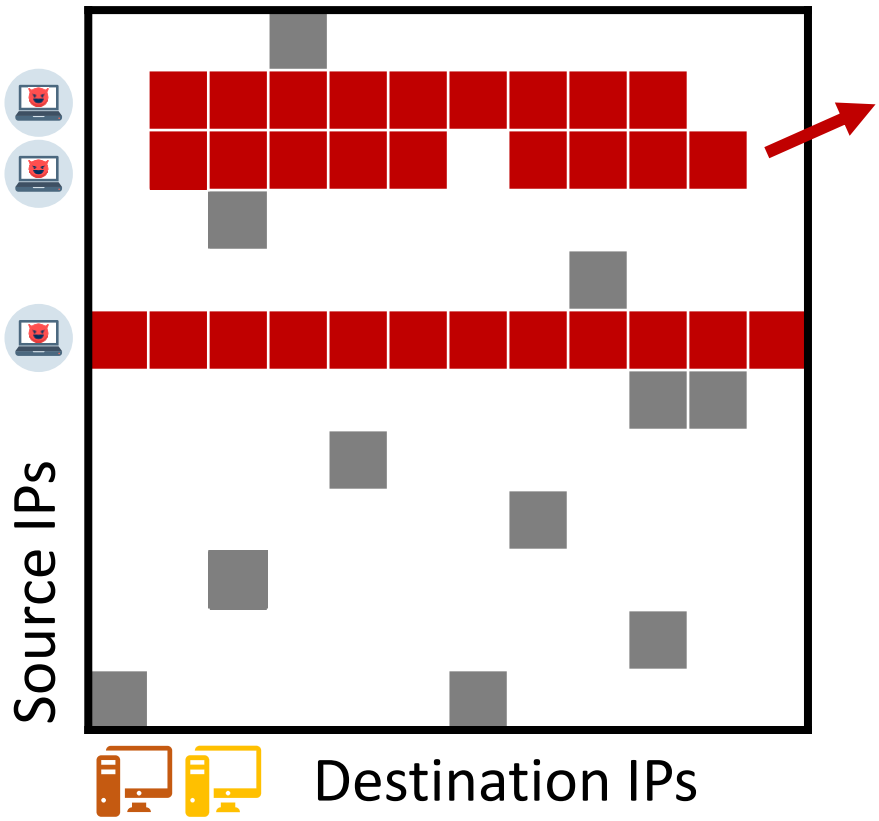
VS



Solution: Find "Good" Near Bi-Cliques

- What is a "good" set of near bi-cliques?

An Adjacency Matrix of a Dynamic Graph



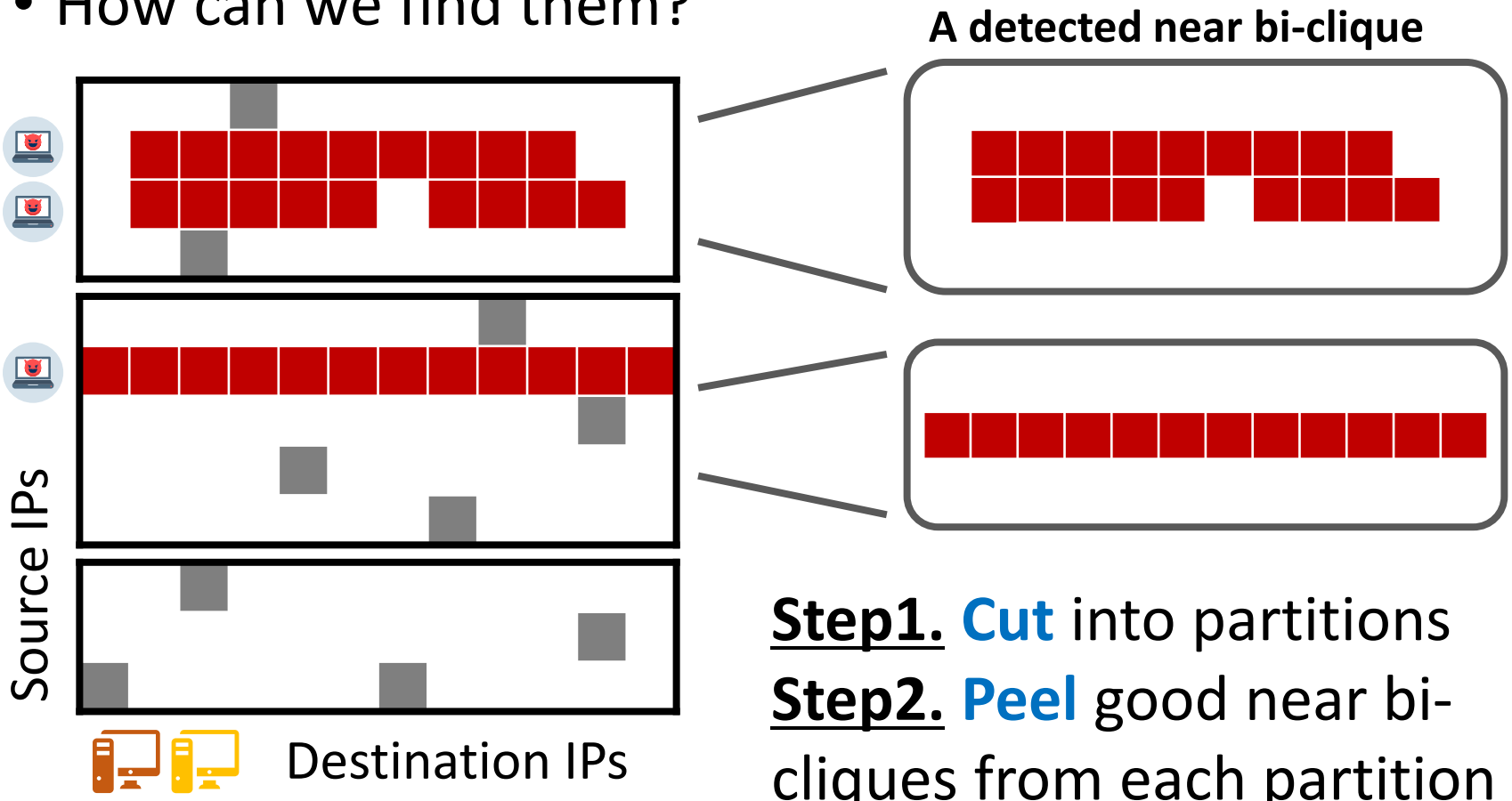
A set of near bi-cliques
- Covering a large portion
  → Exhaustively
- Consisting of many nodes
  → Concisely
- With less error
  → Precisely

Quality Definition

- Each sub-goal minimizes the encoding cost of
  - ✓ Preciseness : missing edges, $\phi_P(\boldsymbol{M})$
  - ✓ Exhaustiveness : remaining edges, $\phi_E(\boldsymbol{R})$
  - ✓ Conciseness : exact bi-cliques, $\phi_C(\boldsymbol{B})$
- Total cost, $\phi(\widehat{\boldsymbol{B}}, \boldsymbol{G}) \coloneqq \phi_P(\boldsymbol{M}) + \phi_E(\boldsymbol{R}) + \phi_C(\boldsymbol{B})$
  - $\widehat{\boldsymbol{B}}$: near bi-cliques obtained from $\boldsymbol{B}$, $\boldsymbol{R}$, and $\boldsymbol{M}$
  - Minimizing $\phi(\widehat{\boldsymbol{B}}, \boldsymbol{G})$ aligns with the Minimum Description Length (MDL) principle to describe $\boldsymbol{G}$

Solution: Find "Good" Near Bi-Cliques

- How can we find them?



Step1. **Cut** into partitions
Step2. **Peel** good near bi-cliques from each partition

Problem Definition

- **Given**: a graph $\boldsymbol{G}$
- **Find**: near bi-cliques $\widehat{\boldsymbol{B}}$
- **To minimize**: the total cost $\boldsymbol{\phi}\left(\widehat{\boldsymbol{B}}, \boldsymbol{G}\right)$

Search Algorithm

- Our goal is to design an approximation algorithm

✓ **High Quality & Speed:** quickly detects high-quality near bi-cliques

✓ **Scalability:** runs in near-linear time

✓ **Applicability:** can be applied to compression and pattern discovery

- **CutNPeel**, our proposed method, satisfies all the requirements

Finding One Near Bi-Clique

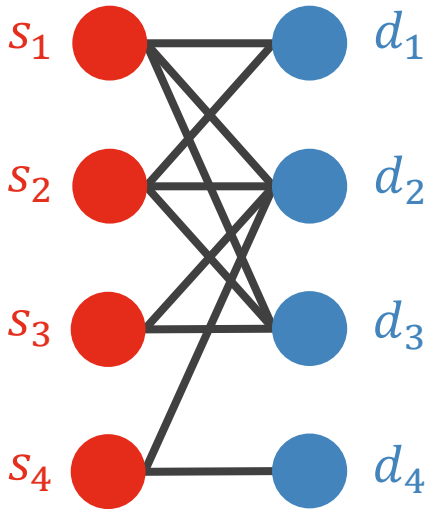
- Find a single near bi-clique $\hat{b}$ with maximum ***saving***

$$\boldsymbol{Saving}\left(\hat{\boldsymbol{b}}, \boldsymbol{G}\right) \coloneqq \boldsymbol{\phi}(\emptyset, \boldsymbol{G}) - \boldsymbol{\phi}(\{\hat{\boldsymbol{b}}\}, \boldsymbol{G})$$

- Saving approximates $\boldsymbol{\phi}\left(\hat{\boldsymbol{B}}, \boldsymbol{G}\right) - \boldsymbol{\phi}\left(\hat{\boldsymbol{B}} \cup \{\hat{\boldsymbol{b}}\}, \boldsymbol{G}\right)$
  - the amount of saving in total cost ϕ due to $\hat{b}$
- Computed in O(1) time with the guarantee in preciseness

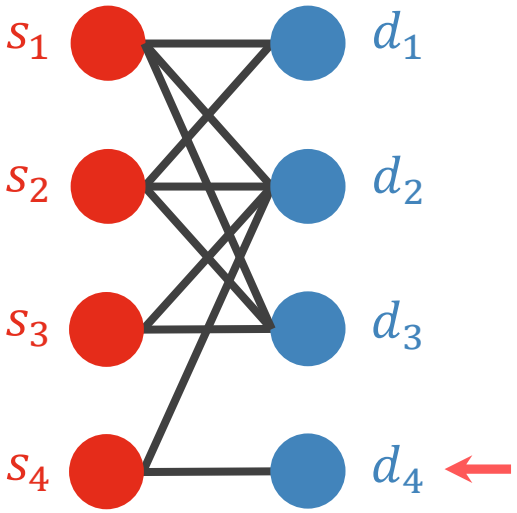
Finding One Near Bi-Clique

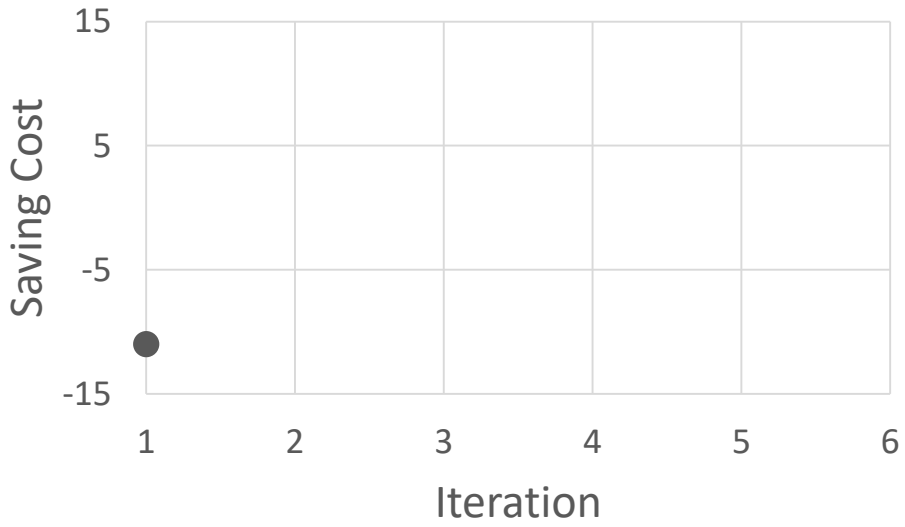
- Top-down search
- Removes a node $v \in S \cup D$ with sparsest connectivity
  - S: source nodes
  - D: destination nodes



Finding One Near Bi-Clique

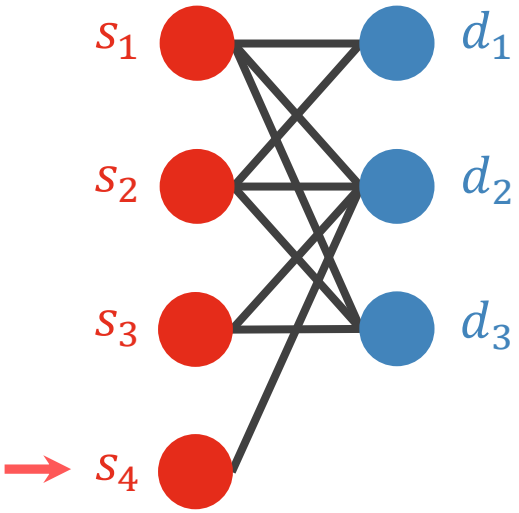
- Top-down search
- Removes a node $v \in S \cup D$ with sparsest connectivity

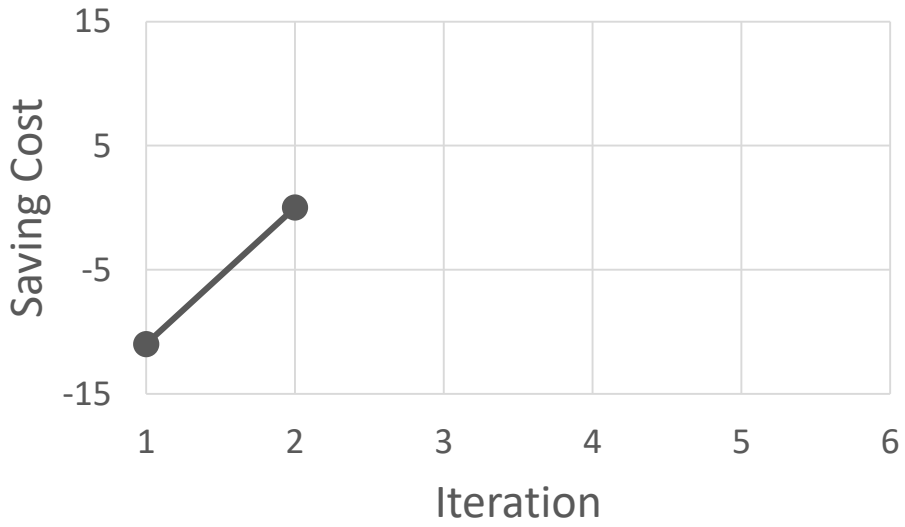




Finding One Near Bi-Clique

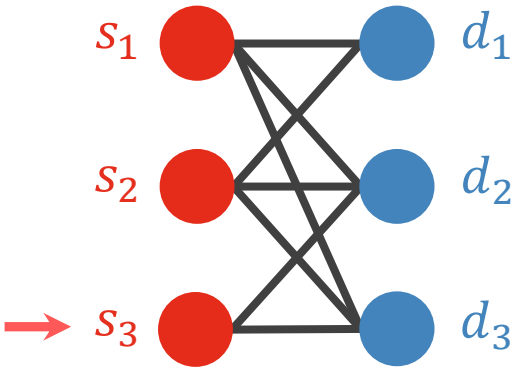
- Top-down search
- Removes a node $v \in S \cup D$ with sparsest connectivity

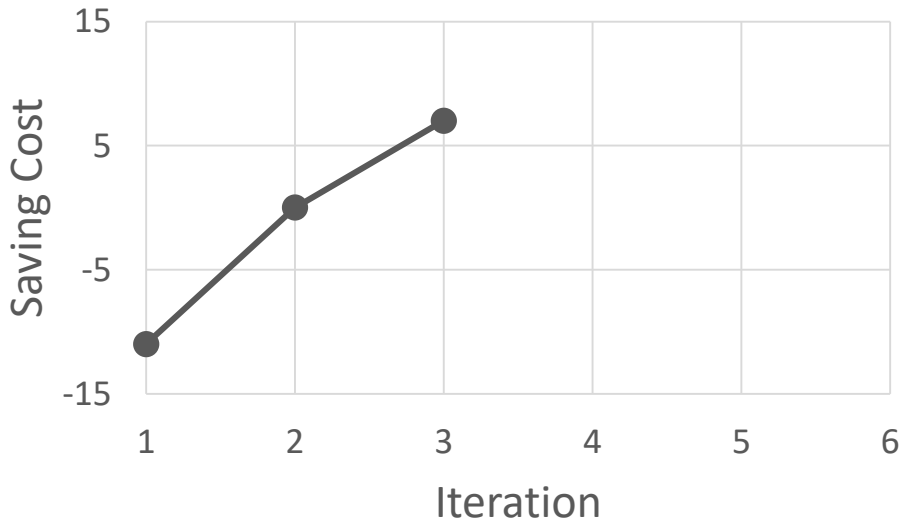




Finding One Near Bi-Clique

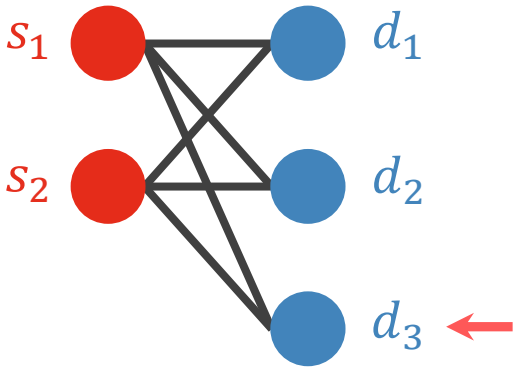
- Top-down search
- Removes a node $v \in S \cup D$ with sparsest connectivity

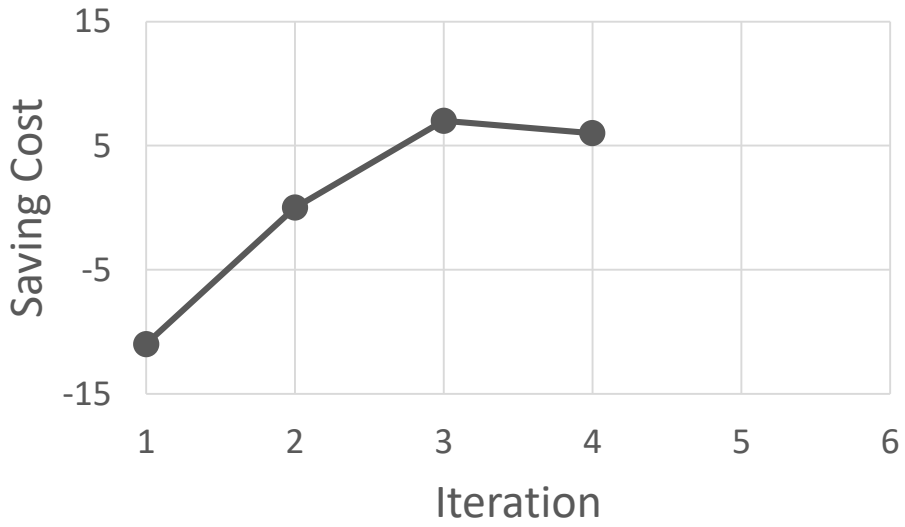




Finding One Near Bi-Clique

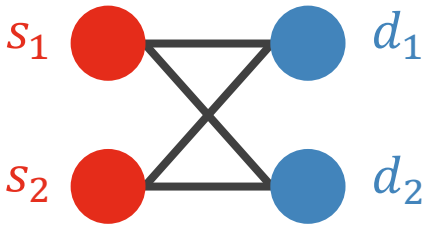
- Top-down search
- Removes a node $v \in S \cup D$ with sparsest connectivity

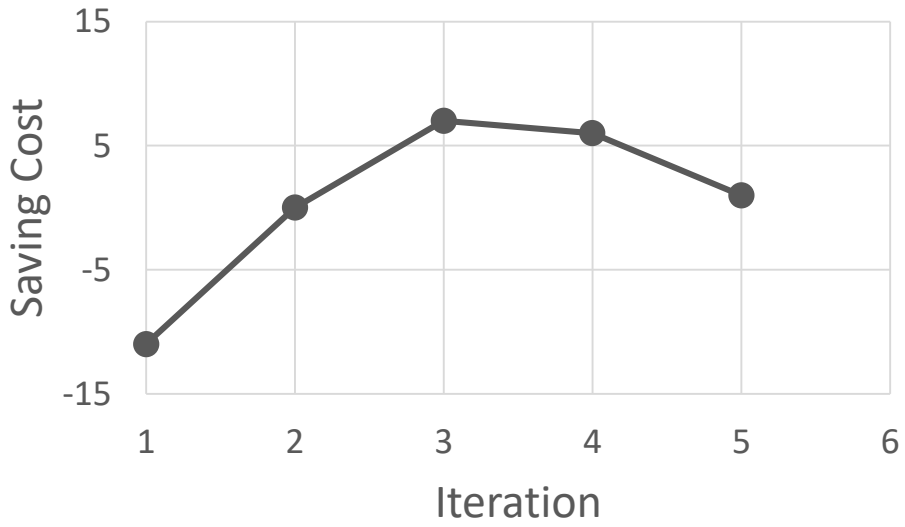




Finding One Near Bi-Clique

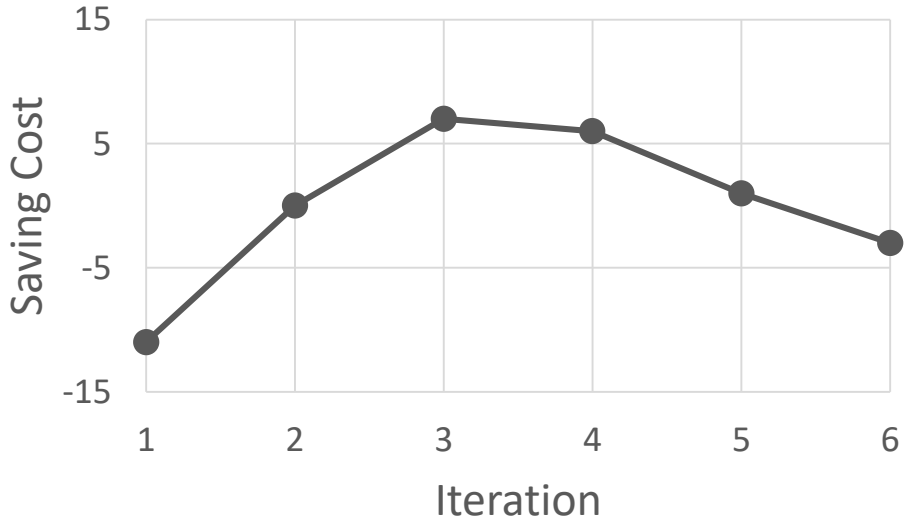
- Top-down search
- Removes a node $v \in S \cup D$ with sparsest connectivity





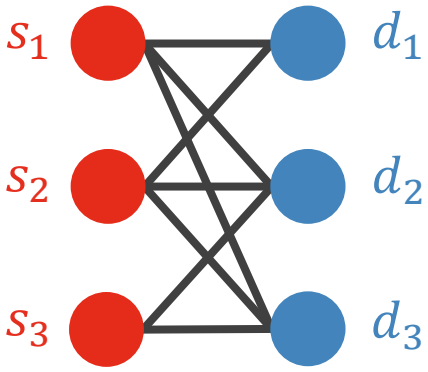
Finding One Near Bi-Clique

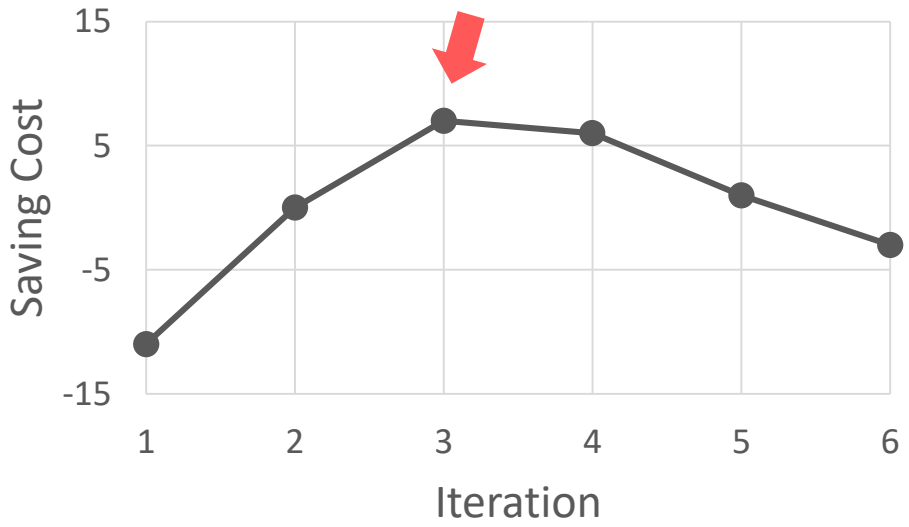
- Until all nodes are removed



Finding One Near Bi-Clique

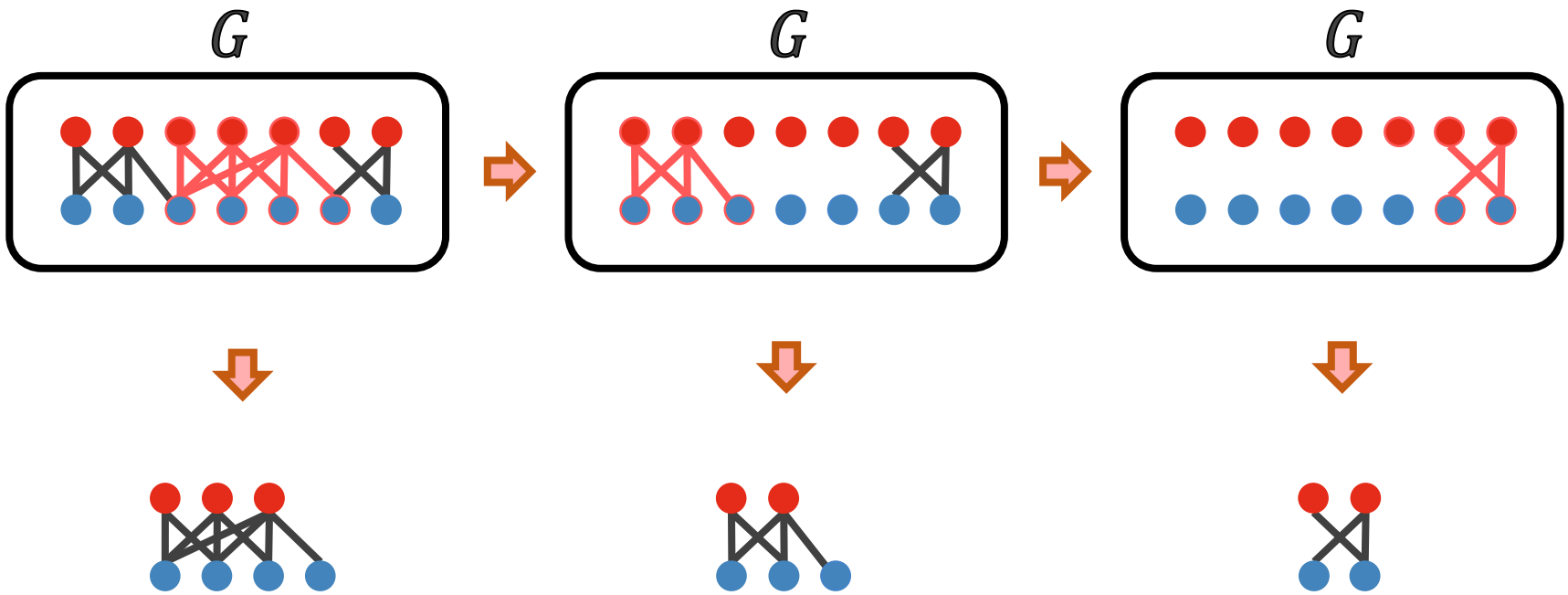
- Output: Return the snapshot where the saving cost is maximized





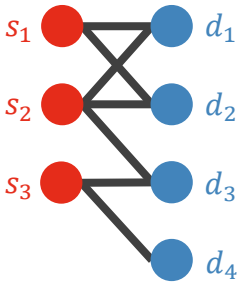
Peel: Preliminary Algorithm

- Remove each detected near bi-clique from the input graph G before finding more bi-cliques

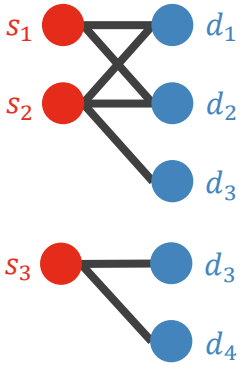


CutNPeel: Search Space Reduction

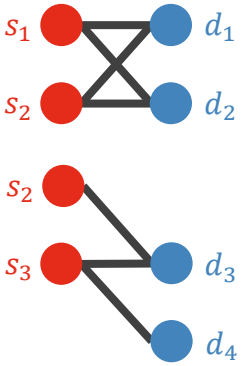
- To reduce the search space, **CutNPeel** uses a partitioning method
- Inspired by min-hashing, **CutNPeel** divides nodes into groups of nodes with similar connectivity
- In each iteration, the leftover graph is re-partitioned



Divide along



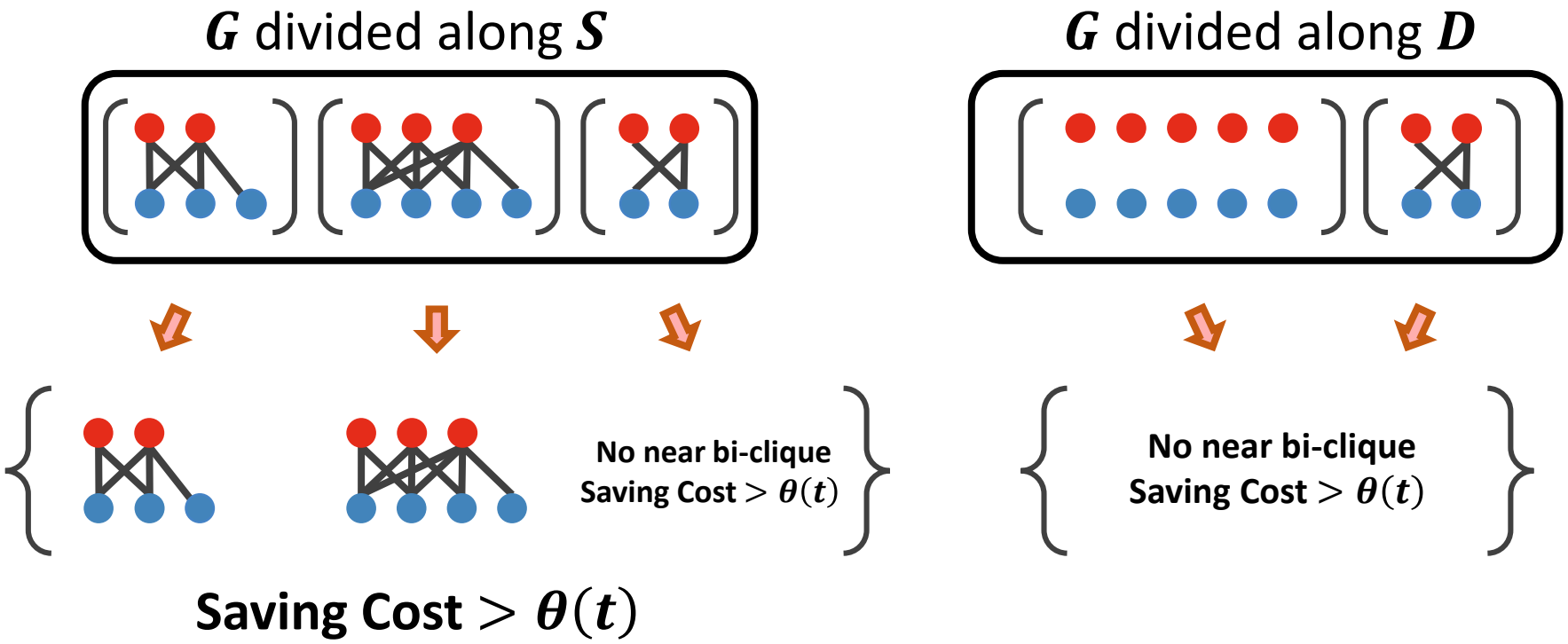
Source S



Destination D

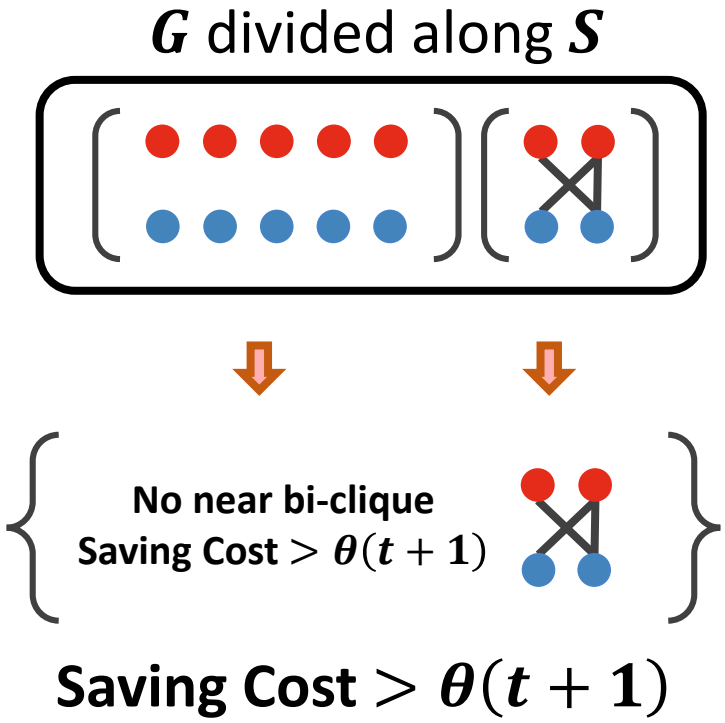
CutNPeel: Quality Guarantee

- In the t-th iteration, **CutNPeel** finds near bi-cliques with saving cost larger than $\theta(\mathrm{t})$ from each partition



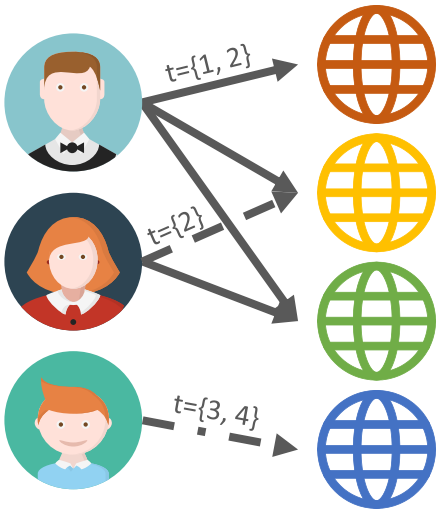
CutNPeel: Quality Guarantee

- In the (t+1)-th iteration, decrease $\theta(\mathrm{t})$ adaptively based on the detected near bi-cliques

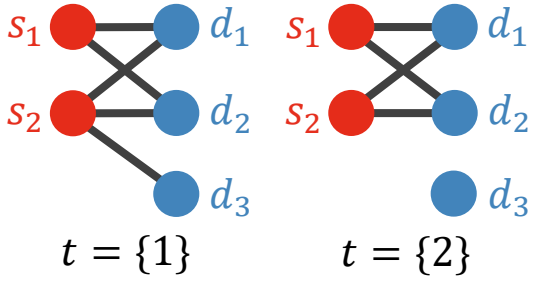


Details in the Paper

- So far, we assumed static graphs for ease of explanation
- In the paper, we considered **dynamic graphs** and **near temporal bi-cliques**



Dynamic Graph



Near Temporal Bi-Cliques

Design of Experiments

- We aim to demonstrate that **CutNPeel**

- ☐ **High Quality & Speed:** quickly detects high-quality near bi-cliques
- ☐ **Scalability:** runs in near-linear time
- ☐ **Applicability:** can be applied to compression and pattern discovery

EXP1. Search Quality & Speed

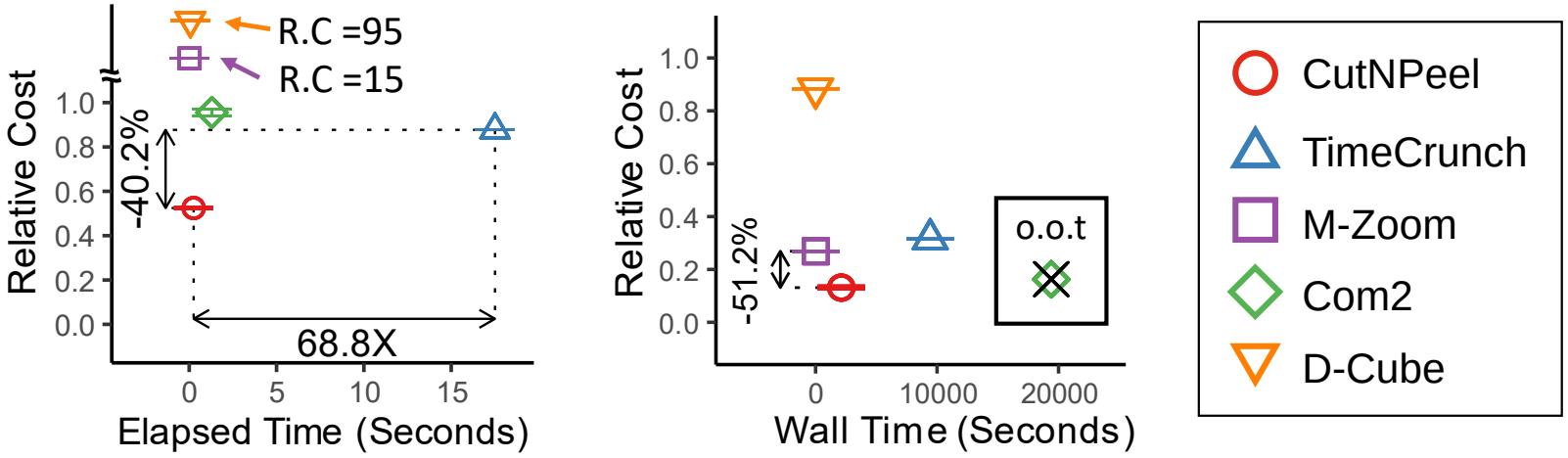
- Q1. Evaluate the **speed** and the **quality** of near bi-cliques
- Quality evaluation metric

$$\textbf{\textit{Relative Cost}} := \frac{Cost\ for\ describing\ \boldsymbol{G}\ using\ \boldsymbol{B}, \boldsymbol{R}\ and\ \boldsymbol{M}}{Cost\ for\ describing\ edges\ in\ \boldsymbol{G}\ individually}$$

- Methods compared :
  - CutNPeel (Proposed)
  - TimeCrunch
  - M-Zoom
  - Com2
  - D-Cube

EXP1. Search Quality & Speed

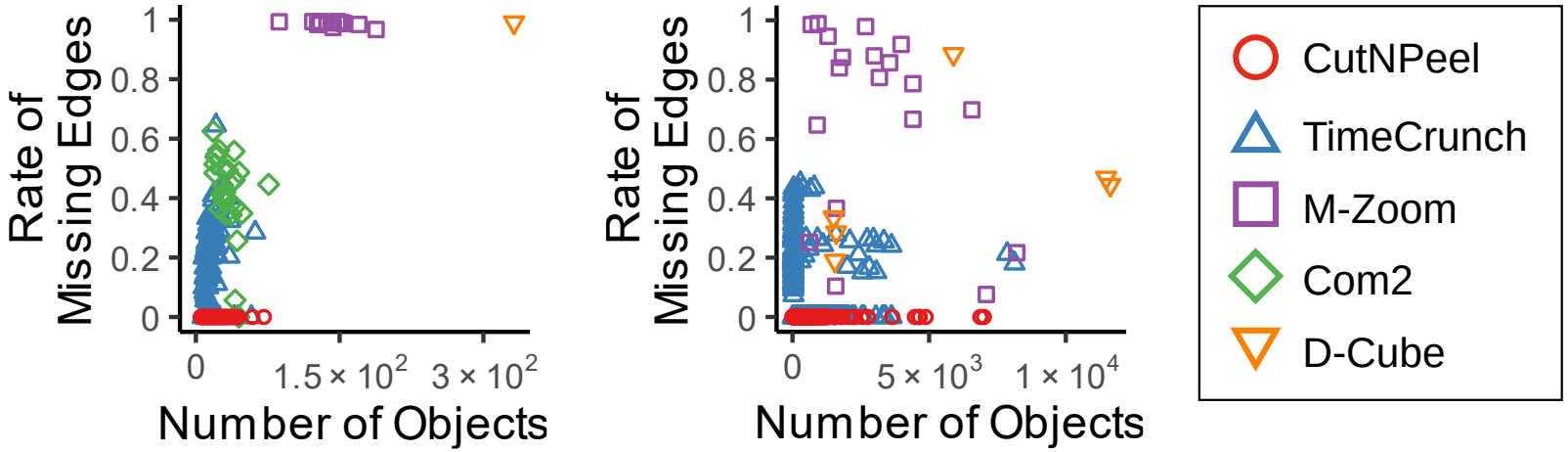
- Six real-world datasets
  - Enron (left) : Sender X Receiver X Time
  - DDoS (right) : Source IP X Destination IP X Time



- **CutNPeel** achieves up to 51.2% better quality up to 68.8X faster compared to the competitors

EXP1. Search Quality & Speed

- Six real-world datasets
  - Enron (left) : Sender X Receiver X Time
  - DDoS (right) : Source IP X Destination IP X Time



- The near bi-cliques detected by **CutNPeel** tend to be precise with smaller ratios of missing edges

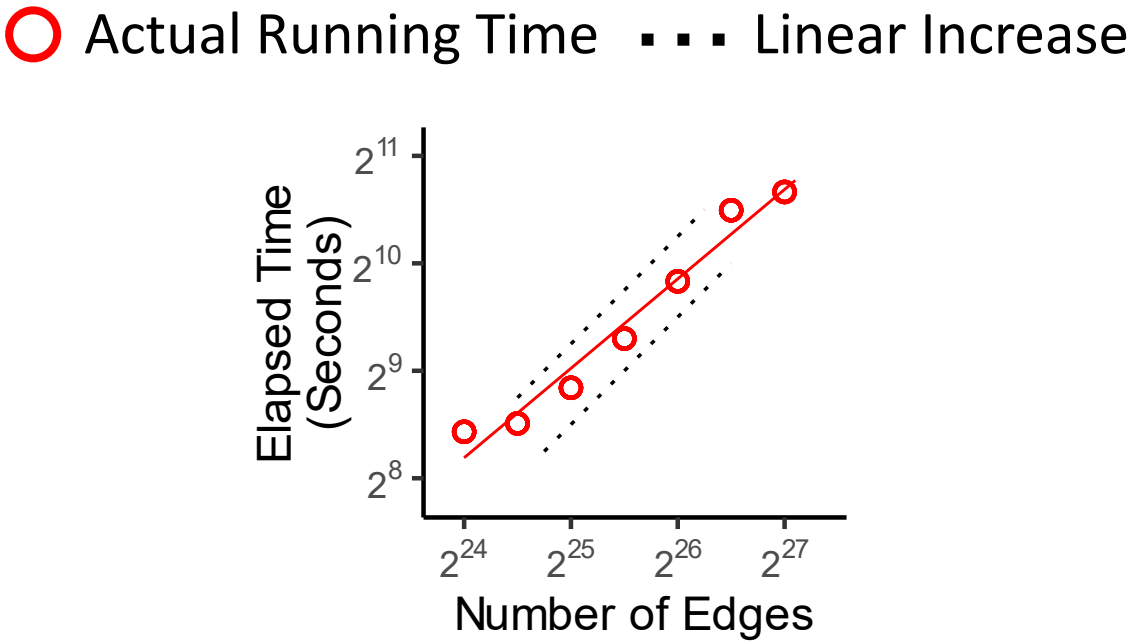
EXP1. Search Quality & Speed

- **CutNPeel** achieves up to 51.2% better quality up to 68.8X faster compared to the competitors
- **CutNPeel** detects subgraphs close to bi-cliques

- ☑ **High Quality & Speed:** quickly detects high-quality near bi-cliques
- ☐ **Scalability:** runs in near-linear time
- ☐ **Applicability:** can be applied to compression and pattern discovery

EXP2. Scalability

- Q2. Measure the **scalability** w.r.t. the size of $\boldsymbol{G}$
- **CutNPeel** scales almost linearly



EXP2. Scalability

- ☑ **High Quality & Speed:** quickly detects high-quality near bi-cliques
- ☑ **Scalability:** runs in near-linear time
- ☐ **Applicability:** can be applied to compression and pattern discovery

EXP3. Application: Compression

- Q3. Evaluate the **compression** ability of CutNPeel
- **CutNPeel** achieved the best compression in all considered datasets

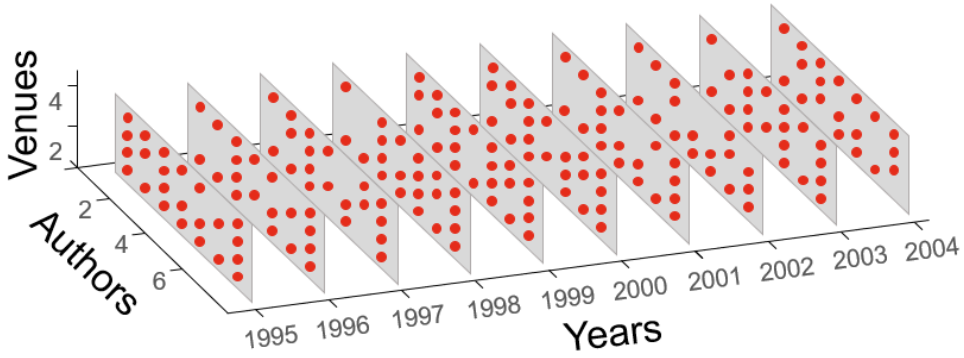
Metric*	Method	Compression Rates in % (the lower, the better)					
		Enron	Darpa	DDoS	DBLP	Yelp	Weeplaces
CutNPeel	CutNPeel	52.5	32.3	13.1	53.7	**77.1**	55.9
	TimeCrunch	87.7	37.2	31.5	74.2	100.6	o.o.t.
	Com2	95.5	100	o.o.t.	99.9	o.o.t.	o.o.t.
TimeCrunch**	TimeCrunch	86.3	36.7	16.3	79.7	100.0	o.o.t.
Com2	CutNPeel	**48.3**	**27.1**	**8.0**	**52.4**	78.0	**52.4**
	TimeCrunch	79.3	34.6	22.8	63.9	100.2	o.o.t.
	Com2	58.3	202.8	o.o.t.	57.5	o.o.t.	o.o.t.

*Different metrics are based on different encoding methods

**The encoding method used in TimeCrunch is not applicable to CutNPeel and Com2

EXP4. Application: Pattern Mining

- **CutNPeel** detects some interesting patterns on the DBLP and DDoS datasets



- ✓ 7 researchers
- ✓ who presented at the 4 same venues
- ✓ from 10 consecutive years from 1995 to 2004

EXP3&4. Application

- ☑ **High Quality & Speed:** quickly detects high-quality near bi-cliques
- ☑ **Scalability:** runs in near-linear time
- ☑ **Applicability:** can be applied to compression and pattern discovery

Conclusion

Considering **the problem of finding a concise, precise and exhaustive set of near bi-cliques** in a dynamic graph,

- We formulated the above problem as an **optimization problem** based on the **MDL principle**
- To solve this problem, we designed **CutNPeel** with the following strengths:

☑ **High Quality & Speed**

☑ **Scalability** ☑ **Applicability**

- Github Link: https://github.com/hyeonjeong1/cutnpeel

KAIST AI
Kim Jaechul Graduate School

Finding a Concise, Precise, and Exhaustive Set of Near Bi-Cliques in Dynamic Graphs

Hyeonjeong Shin[1], Taehyung Kwon[1], Neil Shah[2], Kijung Shin[1]

[1] KAIST, [2] Snap Inc.