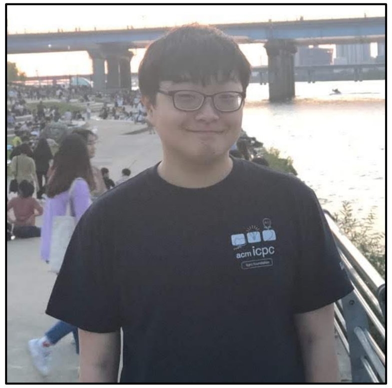
KAIST AI

KDD2020

SSumM : Sparse Summarization of Massive Graphs

Kyuhan Lee*

Hyeonsoo Jo*



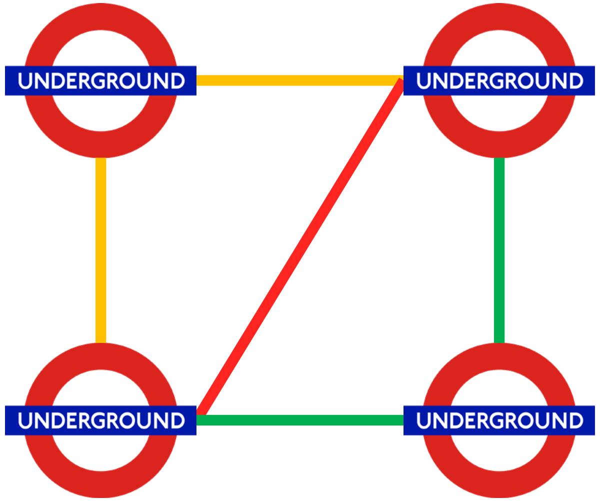
Jihoon Ko

Sungsu Lim

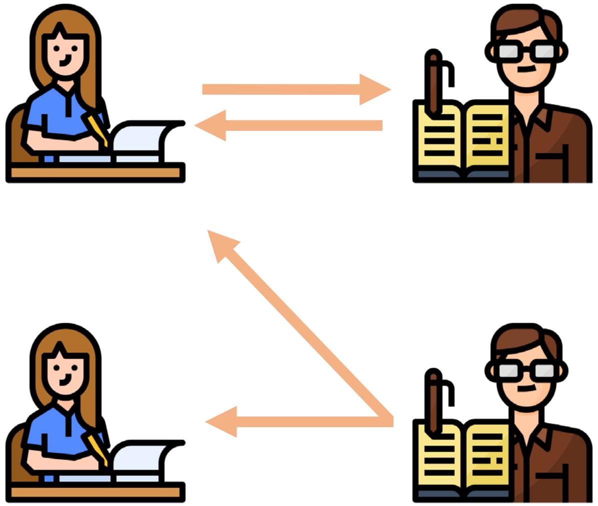
Kijung Shin

Graphs are Everywhere

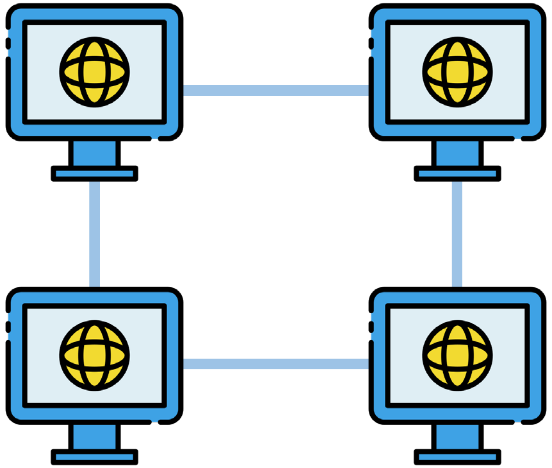
Subway networks



Citation networks



Internet topologies

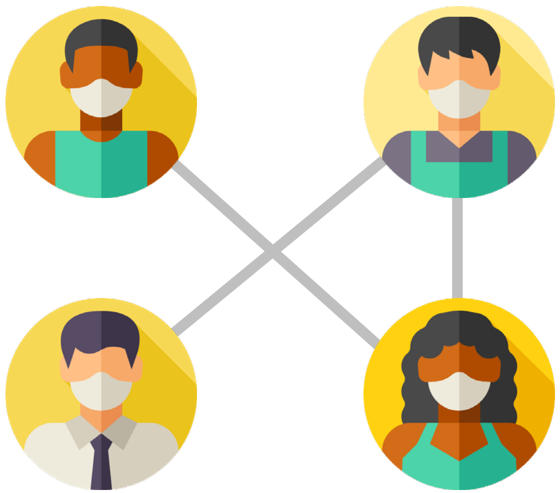


Designed by Freepik from Flaticon

Massive Graphs Appeared

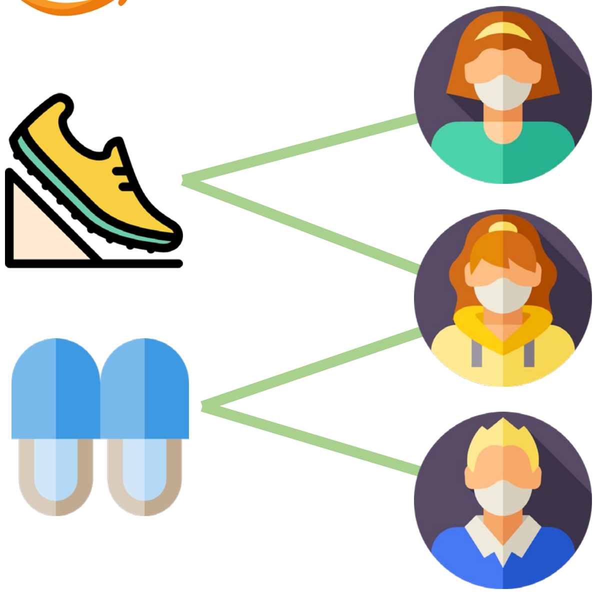
Social networks

2.49 Billion active users



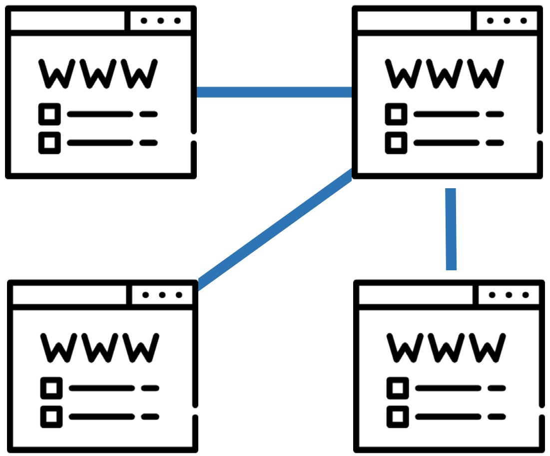
Purchase histories

0.5 Billion products



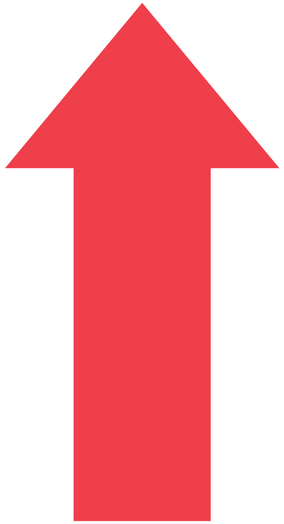
World Wide Web

5.49 Billion web pages

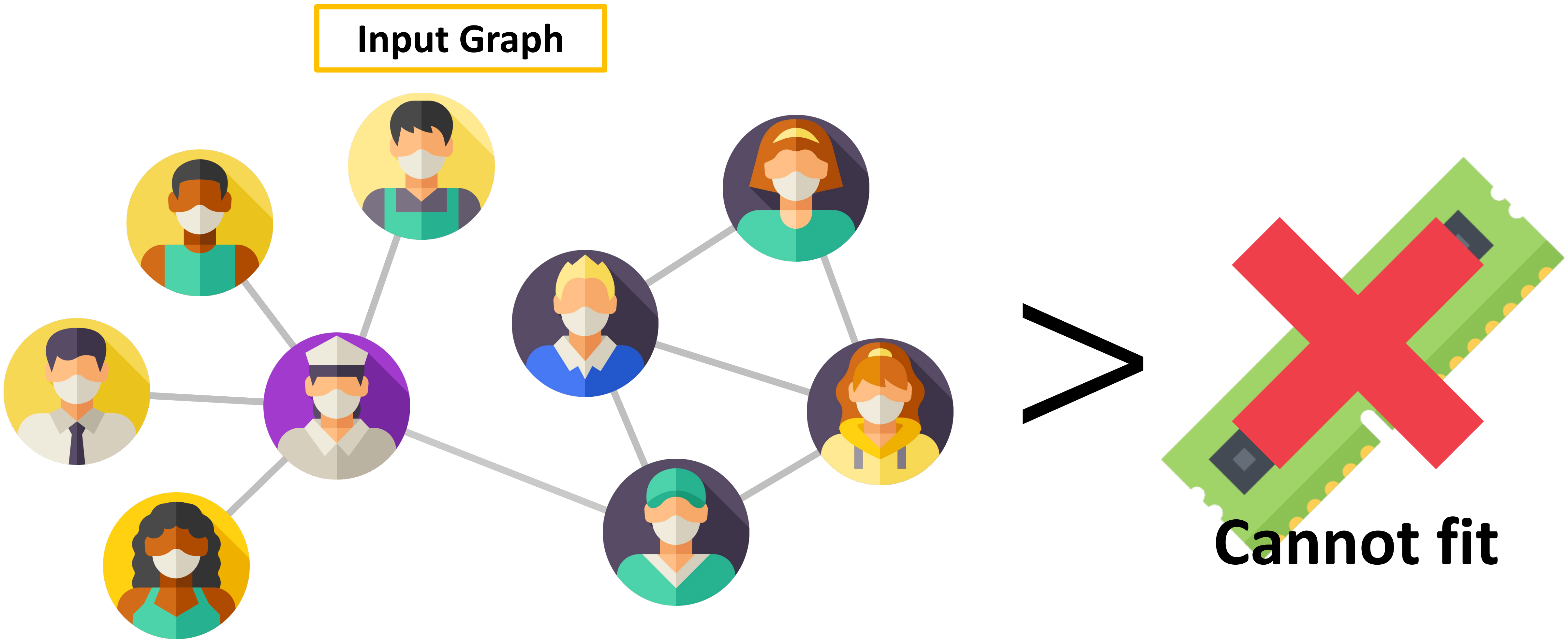


Difficulties in Analyzing Massive graphs

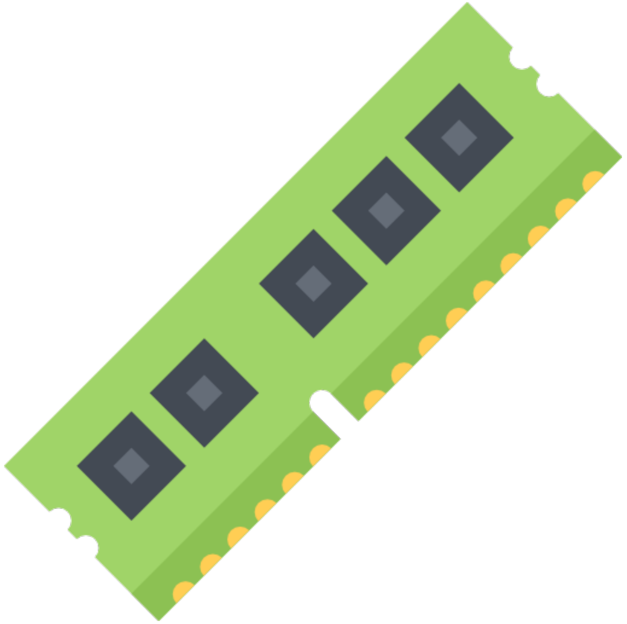
Computational cost
(number of nodes & edges)



Difficulties in Analyzing Massive graphs



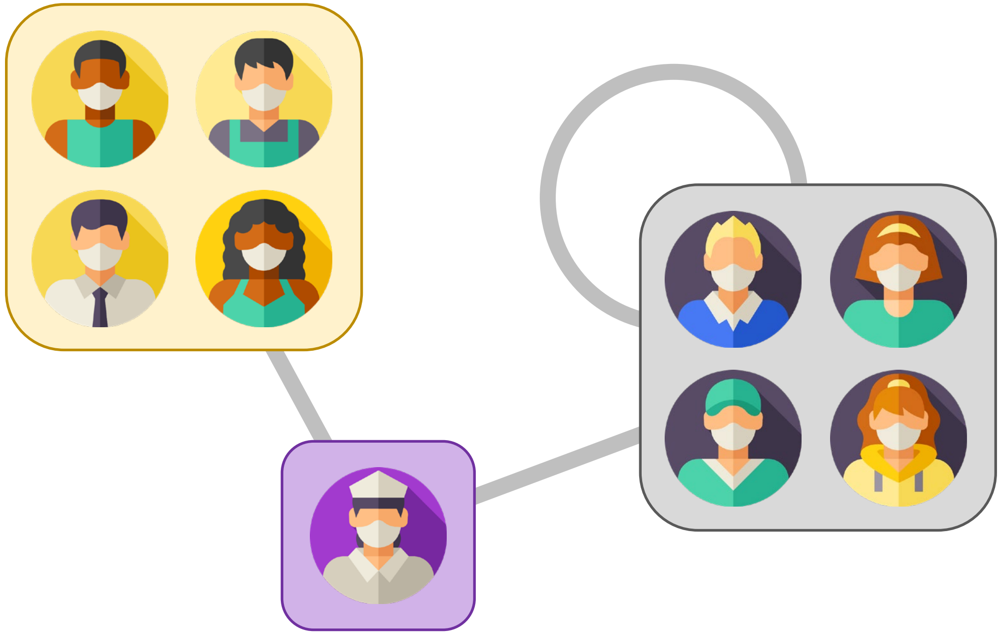
Solution: Graph Summarization



$>$

Can fit

Summary Graph

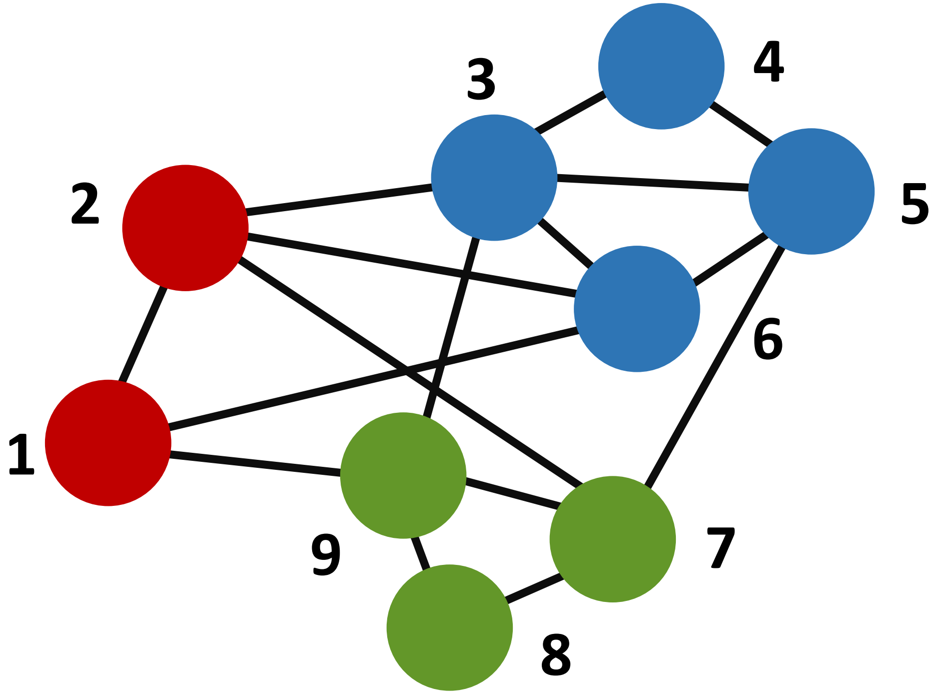


Advantages of Graph Summarization

- Many graph compression techniques are available
  - TheWebGraph Framework [BV04]
  - BFS encoding [AD09]
  - SlashBurn [KF11]
  - VoG [KKVF14]

- **Graph summarization** stands out because
  - **Elastic**: reduce size of outputs as much as we want
  - **Analyzable**: existing graph analysis and tools can be applied
  - **Combinable for Additional Compression**: can be further compressed

Example of Graph Summarization

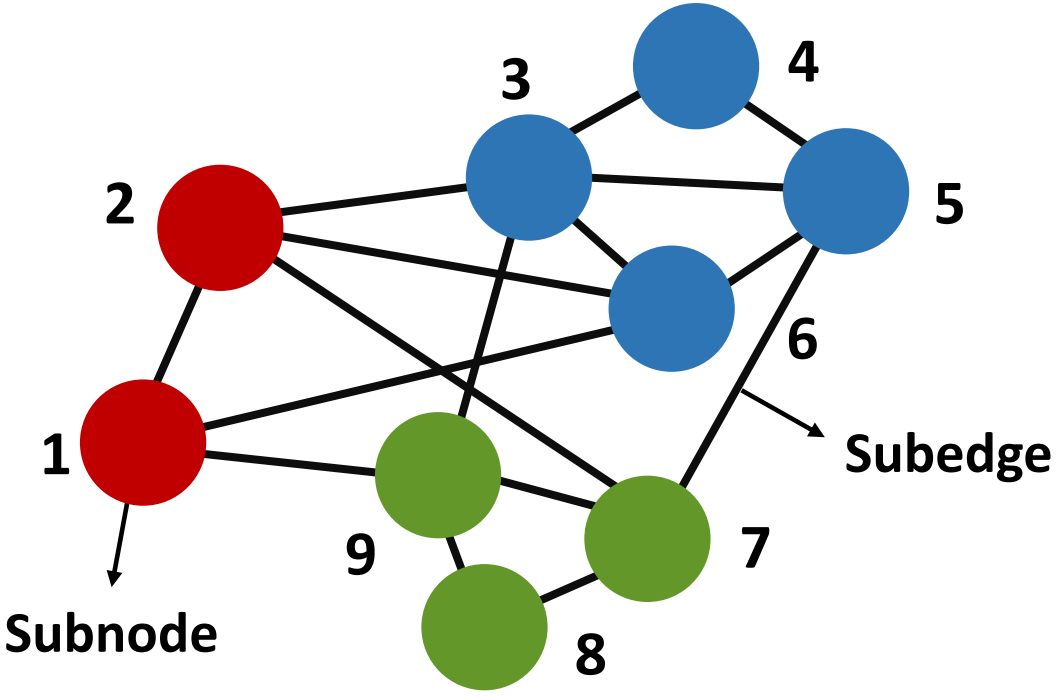


Input Graph

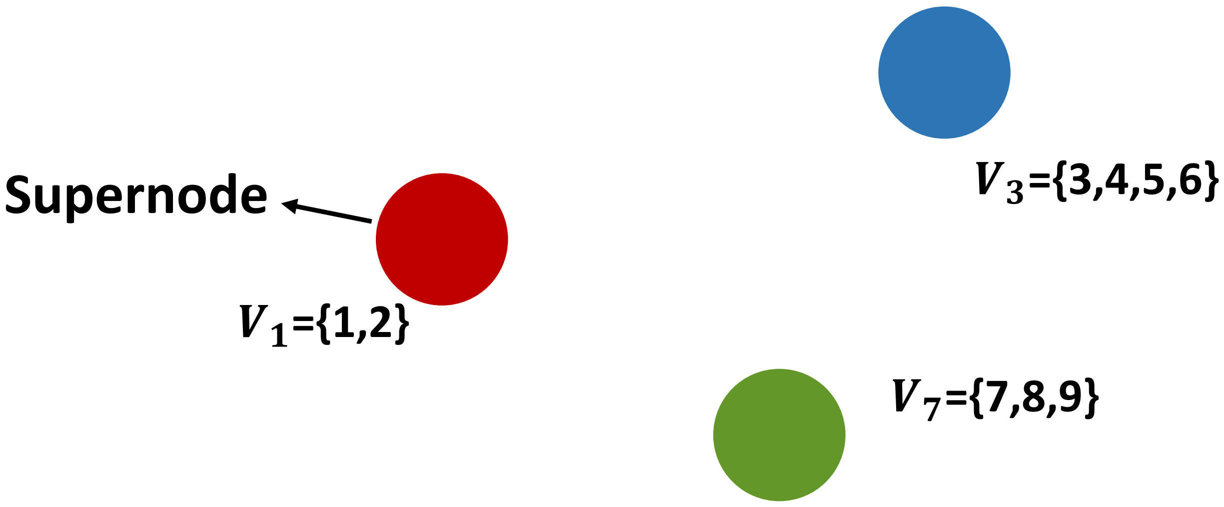
	1	2	3	4	5	6	7	8	9
1	0	1	0	0	0	1	0	0	1
2	1	0	1	0	0	1	1	0	0
3	0	1	0	1	1	1	0	0	1
4	0	0	1	0	1	0	0	0	0
5	0	0	1	1	0	1	1	0	0
6	1	1	1	0	1	0	0	0	0
7	0	1	0	0	1	0	0	1	1
8	0	0	0	0	0	0	1	0	1
9	1	0	1	0	0	0	1	1	0

Adjacency Matrix

Example of Graph Summarization

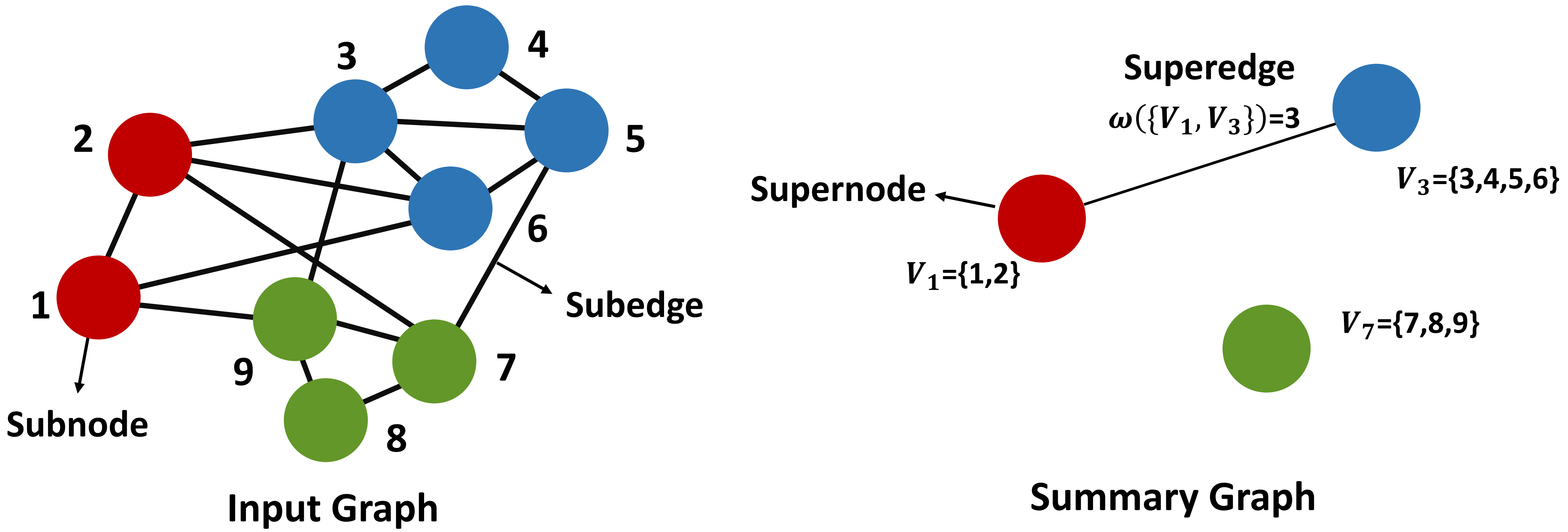


Input Graph

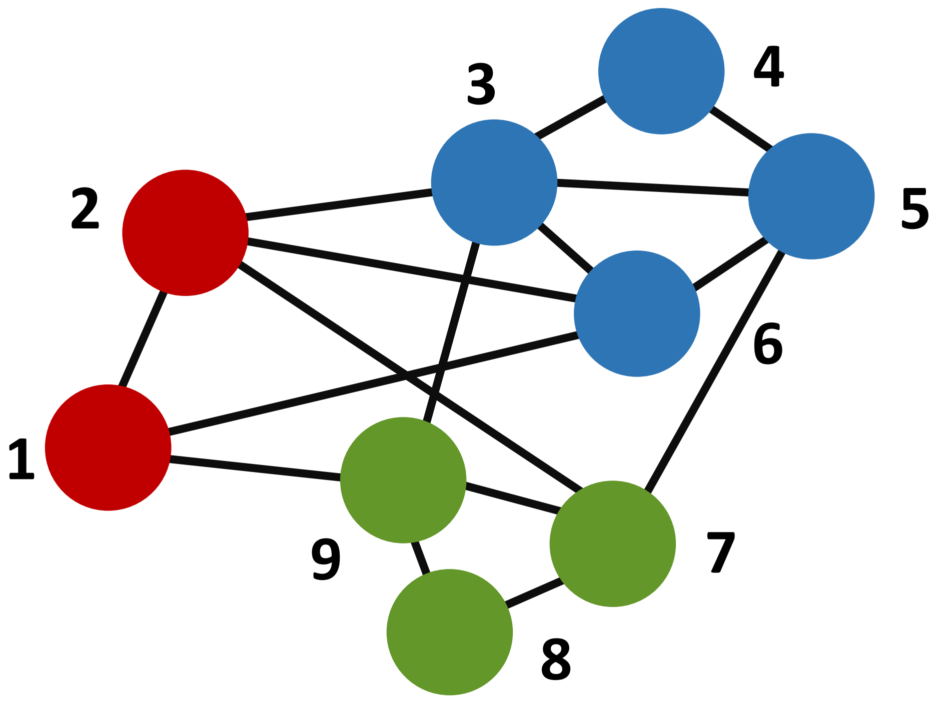


Summary Graph

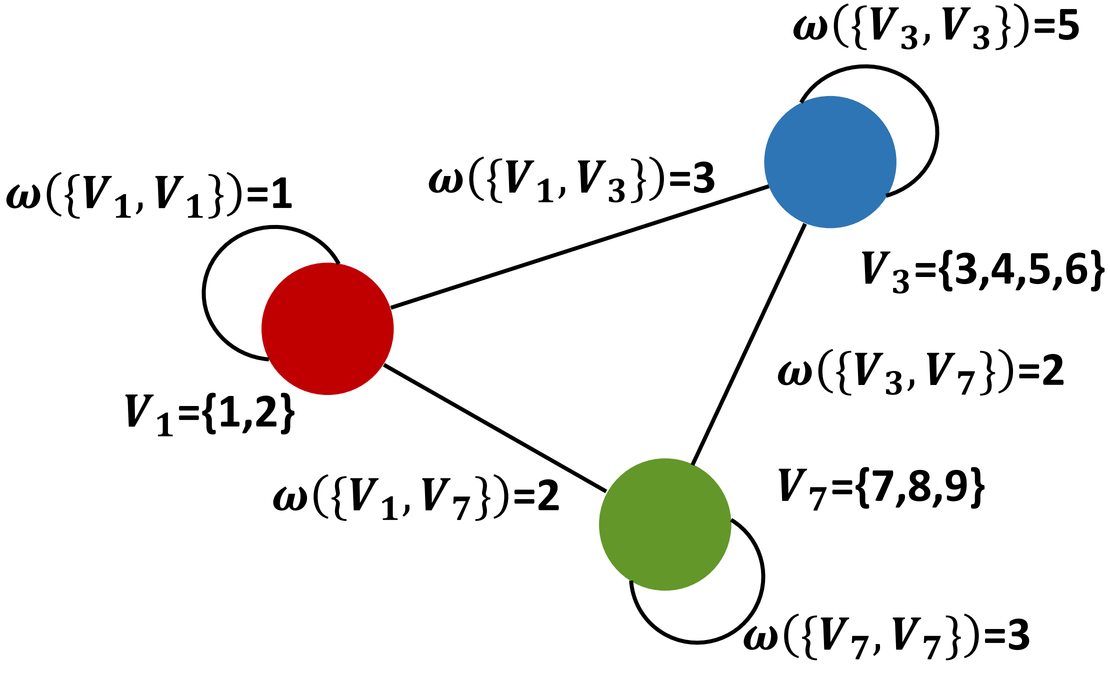
Example of Graph Summarization



Example of Graph Summarization

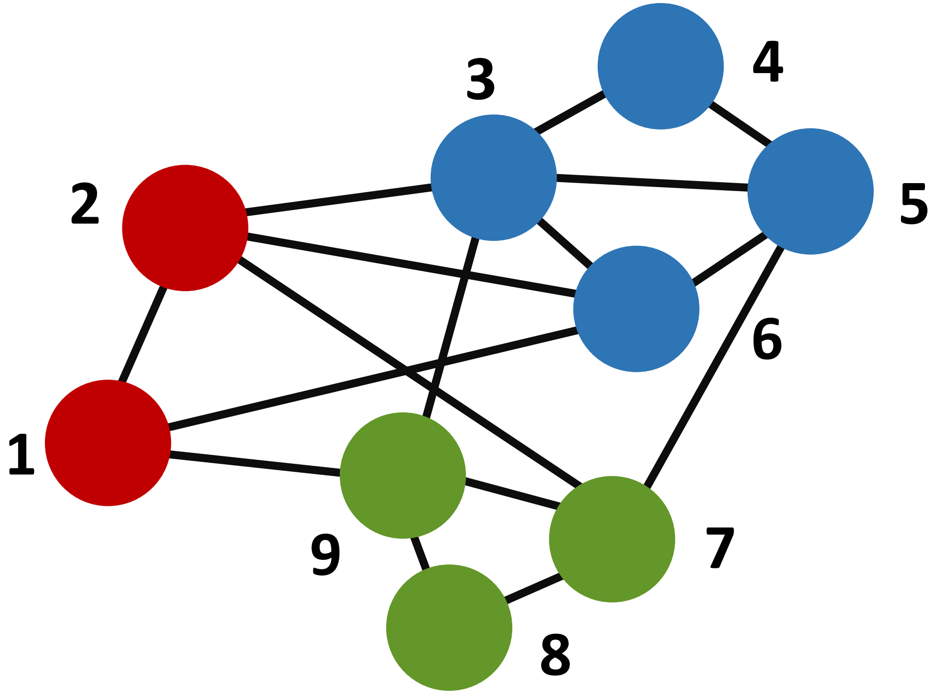


Input Graph

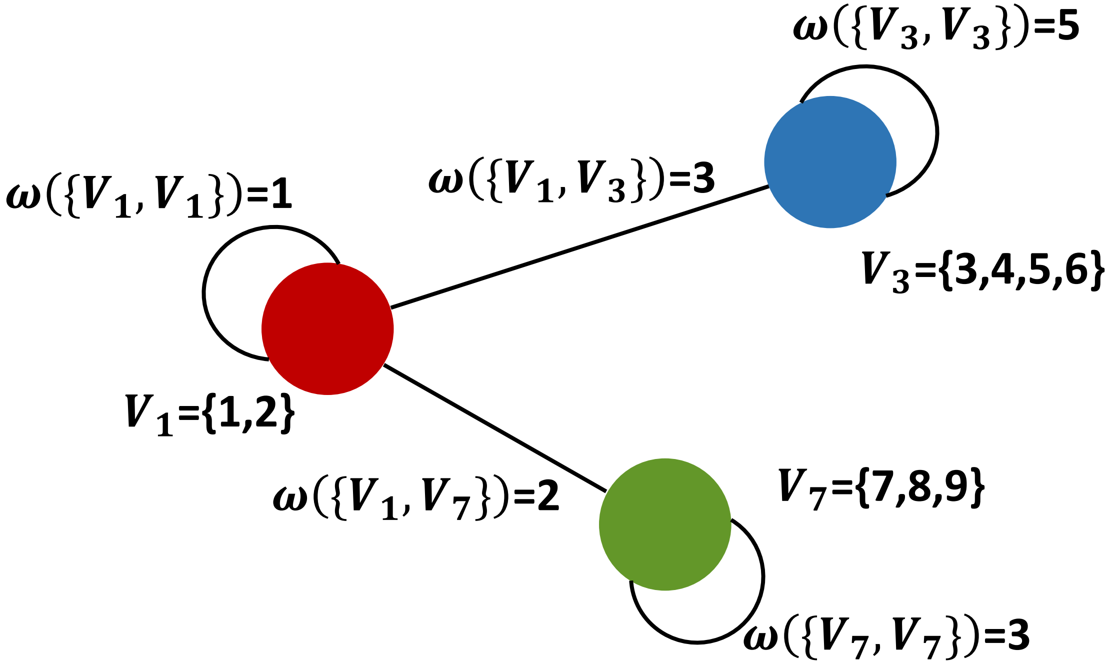


Summary Graph

Example of Graph Summarization

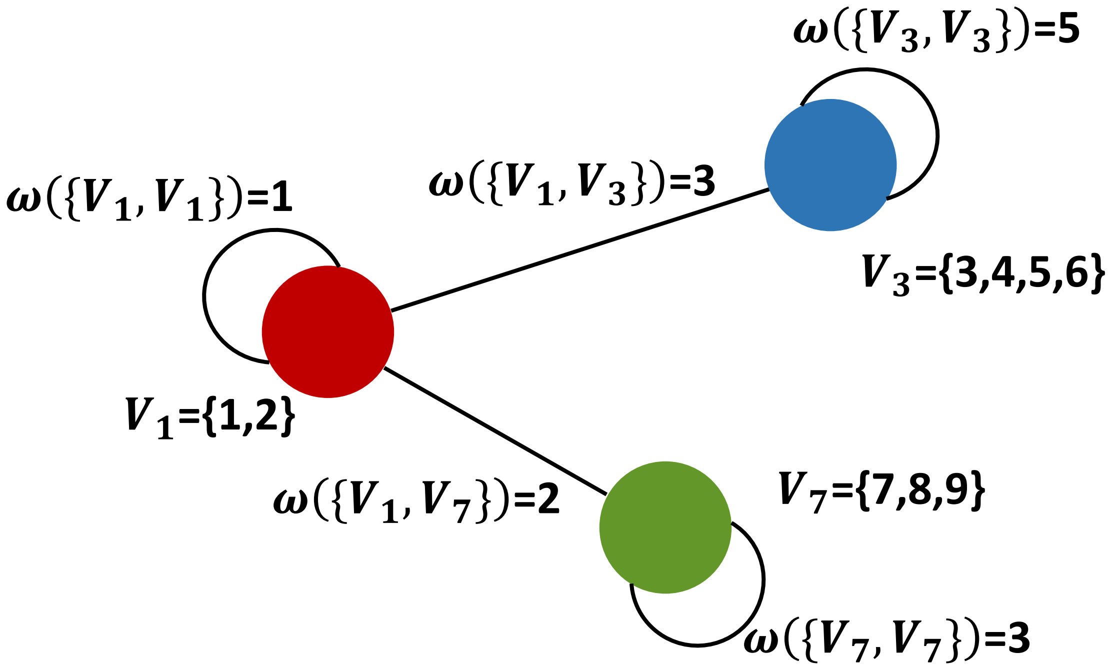


Input Graph



Summary Graph

Example of Graph Summarization



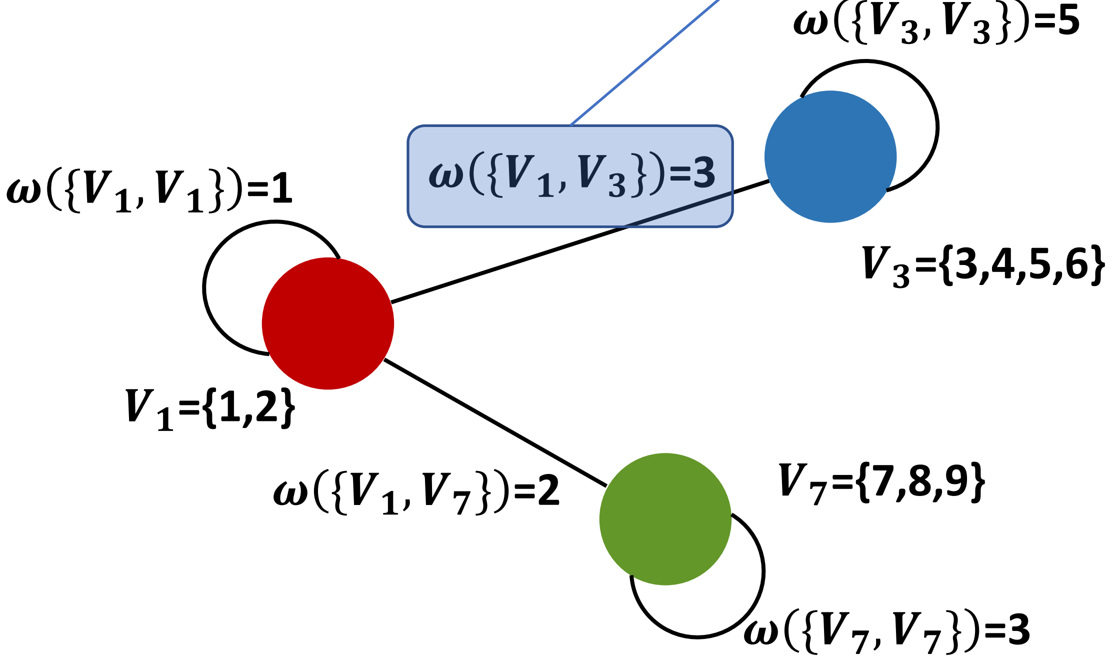
Summary Graph

	1	2	3	4	5	6	7	8	9
1	0	1	3/8	3/8	3/8	3/8	1/3	1/3	1/3
2	1	0	3/8	3/8	3/8	3/8	1/3	1/3	1/3
3	3/8	3/8	0	5/6	5/6	5/6	0	0	0
4	3/8	3/8	5/6	0	5/6	5/6	0	0	0
5	3/8	3/8	5/6	5/6	0	5/6	0	0	0
6	3/8	3/8	5/6	5/6	5/6	0	0	0	0
7	1/3	1/3	0	0	0	0	0	1	1
8	1/3	1/3	0	0	0	0	1	0	1
9	1/3	1/3	0	0	0	0	1	1	0

Reconstructed Adjacency Matrix

Example of Graph Summarization

$$\frac{\omega(\{V_1, V_3\}) = 3}{Maximum\ number\ of\ subedges = 8}$$



Summary Graph

	1	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**
1	0	1	3/8	3/8	3/8	3/8	1/3	1/3	1/3
2	1	0	3/8	3/8	3/8	3/8	1/3	1/3	1/3
3	3/8	3/8	0	5/6	5/6	5/6	0	0	0
4	3/8	3/8	5/6	0	5/6	5/6	0	0	0
5	3/8	3/8	5/6	5/6	0	5/6	0	0	0
6	3/8	3/8	5/6	5/6	5/6	0	0	0	0
7	1/3	1/3	0	0	0	0	0	1	1
8	1/3	1/3	0	0	0	0	1	0	1
9	1/3	1/3	0	0	0	0	1	1	0

Reconstructed Adjacency Matrix

Road Map

- Introduction
- **Problem <<**
- Proposed Algorithm: SSumM
- Experimental Results
- Conclusions

Problem Definition: Graph Summarization

$\boldsymbol{Given}$:

a **graph** $\boldsymbol{G}$ and the **target number of node** $\boldsymbol{K}$

$\boldsymbol{Find}$:

a **summary graph** $\overline{\boldsymbol{G}}$

$\boldsymbol{To\ Minimize}$:

the difference between graph $\boldsymbol{G}$ and the restored graph $\widehat{\boldsymbol{G}}$

$\boldsymbol{Subject\ to}$:

the number of supernodes in $\overline{\boldsymbol{G}} \leq \boldsymbol{K}$

Problem Definition: Graph Summarization

Given:

a **graph** $\boldsymbol{G}$ an… …de $\boldsymbol{K}$

Find:

a **summary** g…

To Minimize:

the differenc… …restored graph $\widehat{\boldsymbol{G}}$

Subject to:

the number of supernodes in $\overline{\boldsymbol{G}} \leq \boldsymbol{K}$

NEW

Problem Definition: Graph Summarization

Given:

a **graph** $\boldsymbol{G}$ and the **desired size** $\boldsymbol{K}$ **(in bits)**

Find:

a **summary graph** $\overline{\boldsymbol{G}}$

To Minimize:

the difference with graph graph $\boldsymbol{G}$ and the restored graph $\widehat{\boldsymbol{G}}$

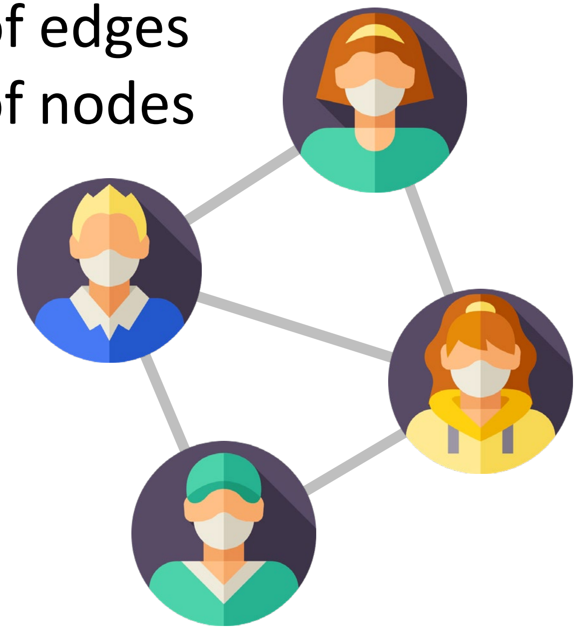
Subject to:

size of $\overline{\boldsymbol{G}}$ in bits $\leq \boldsymbol{K}$

Details: Size in Bits of a Graph

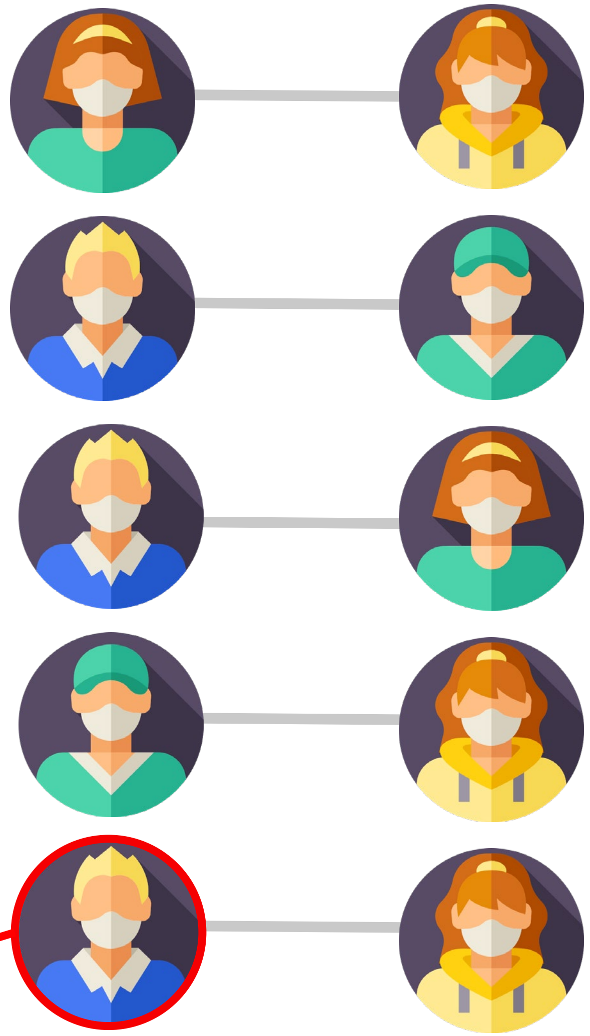
Input graph G

E : set of edges
V : set of nodes



Size of graph: $2|E|\log_2|V|$

Encoded using $\log_2|V|$ bits



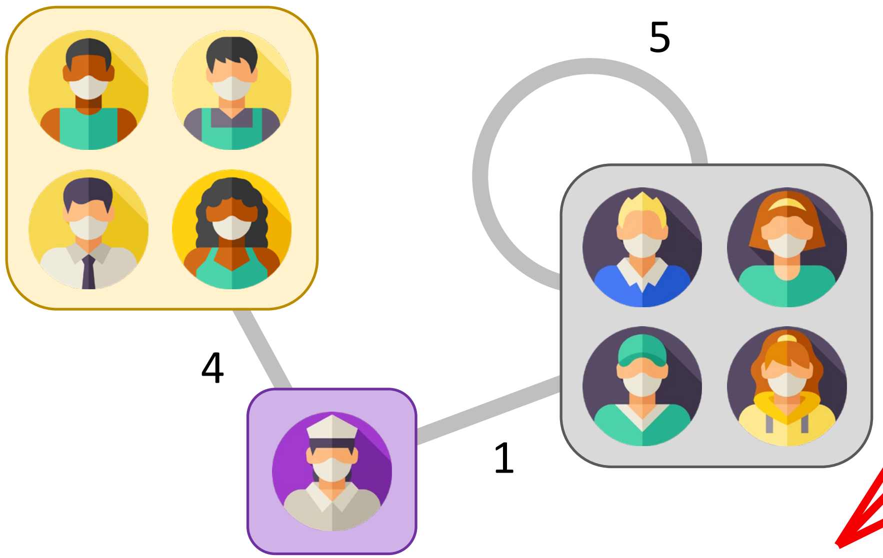
Details: Size in Bits of a Summary Graph

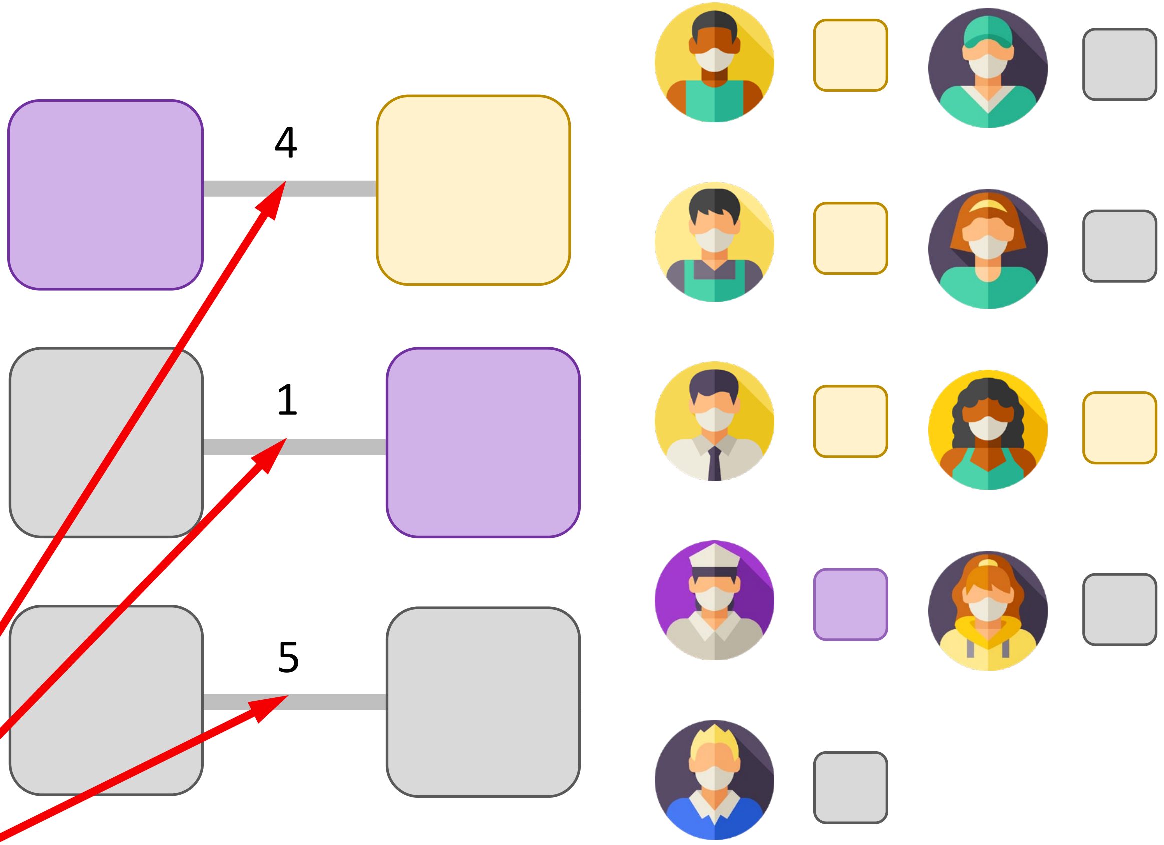
Summary graph $\overline{G}$

S : set of supernodes

P : set of superedges

w_{max} : maximum superedge weight





Size of summary graph: $|P|(2\log_2|S| + \log_2 \omega_{max}) + |V|\log_2|S|$

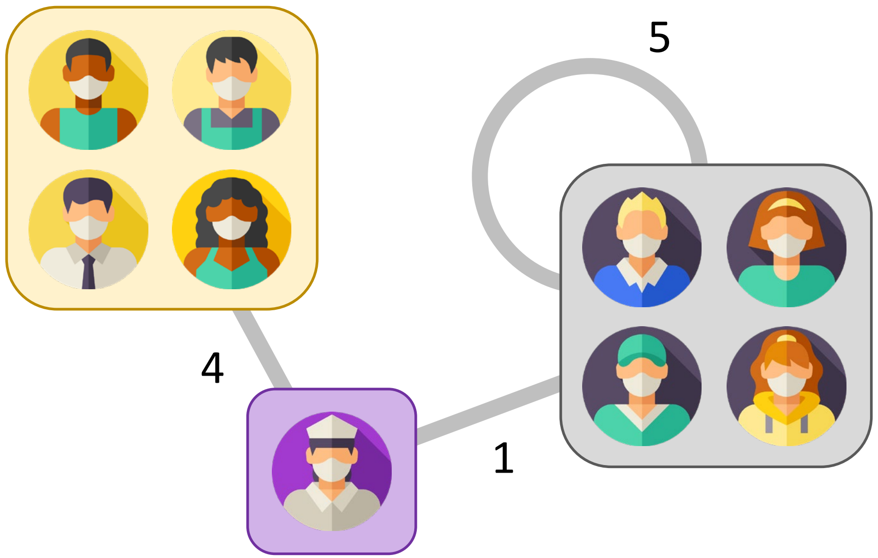
Details: Size in Bits of a Summary Graph

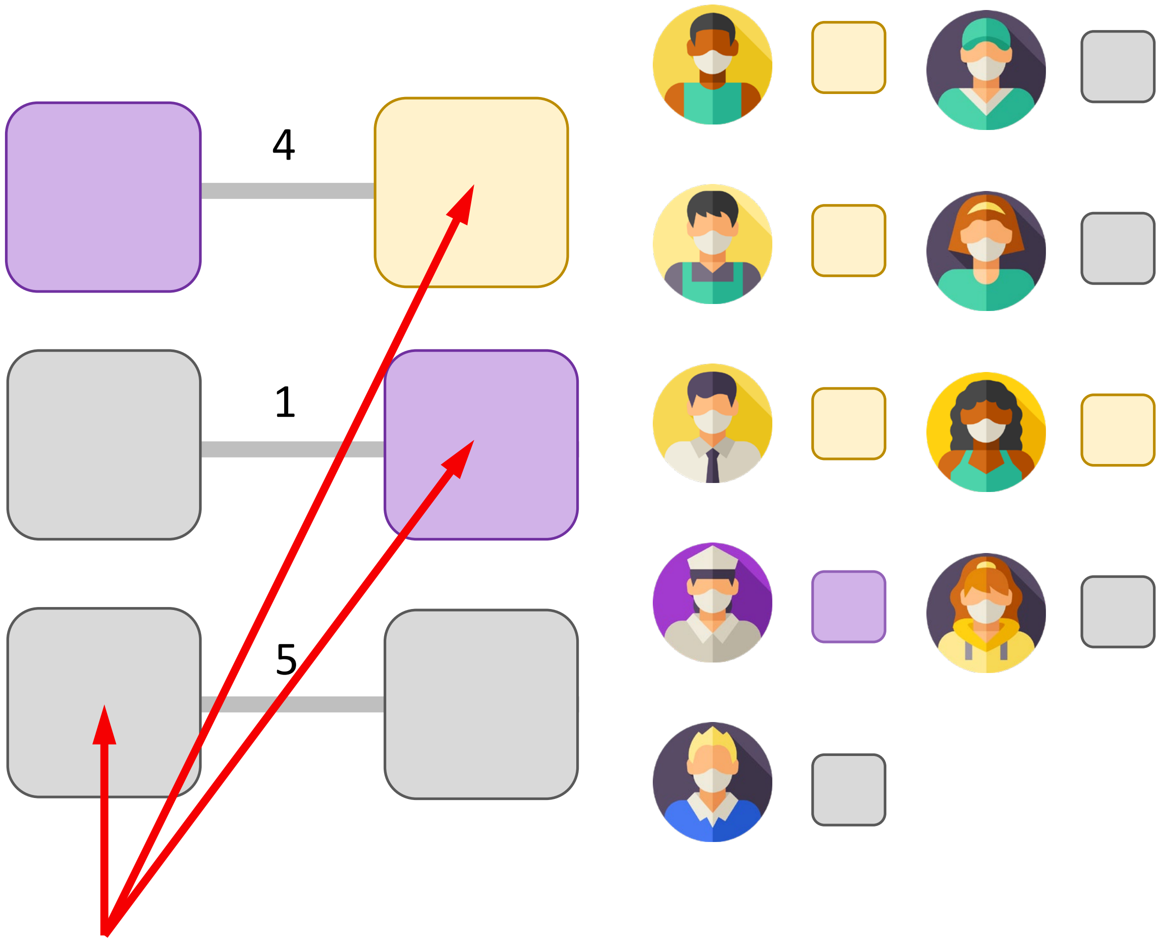
Summary graph $\overline{G}$

S : set of supernodes

P : set of superedges

w_{max} : maximum superedge weight





Size of summary graph: $|P|(2\log_2|S| + \log_2 \omega_{max}) + |V|\log_2|S|$

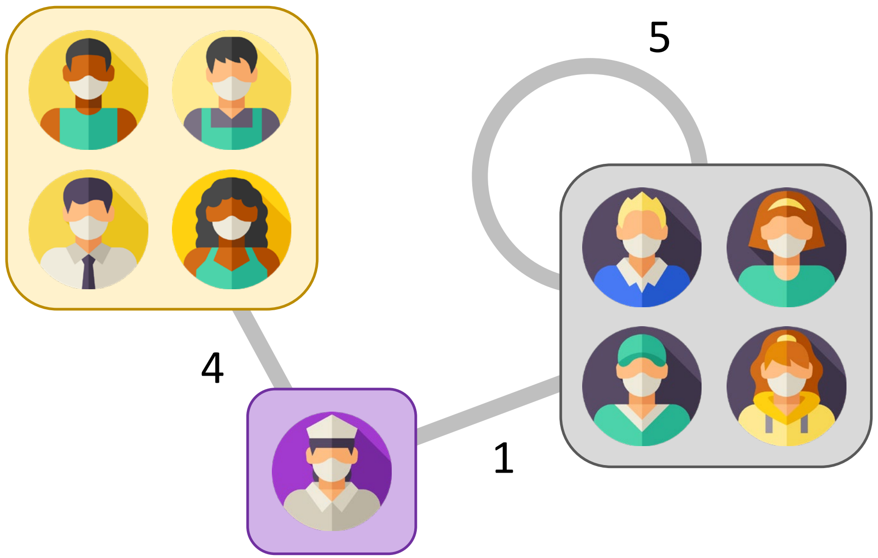
Details: Size in Bits of a Summary Graph

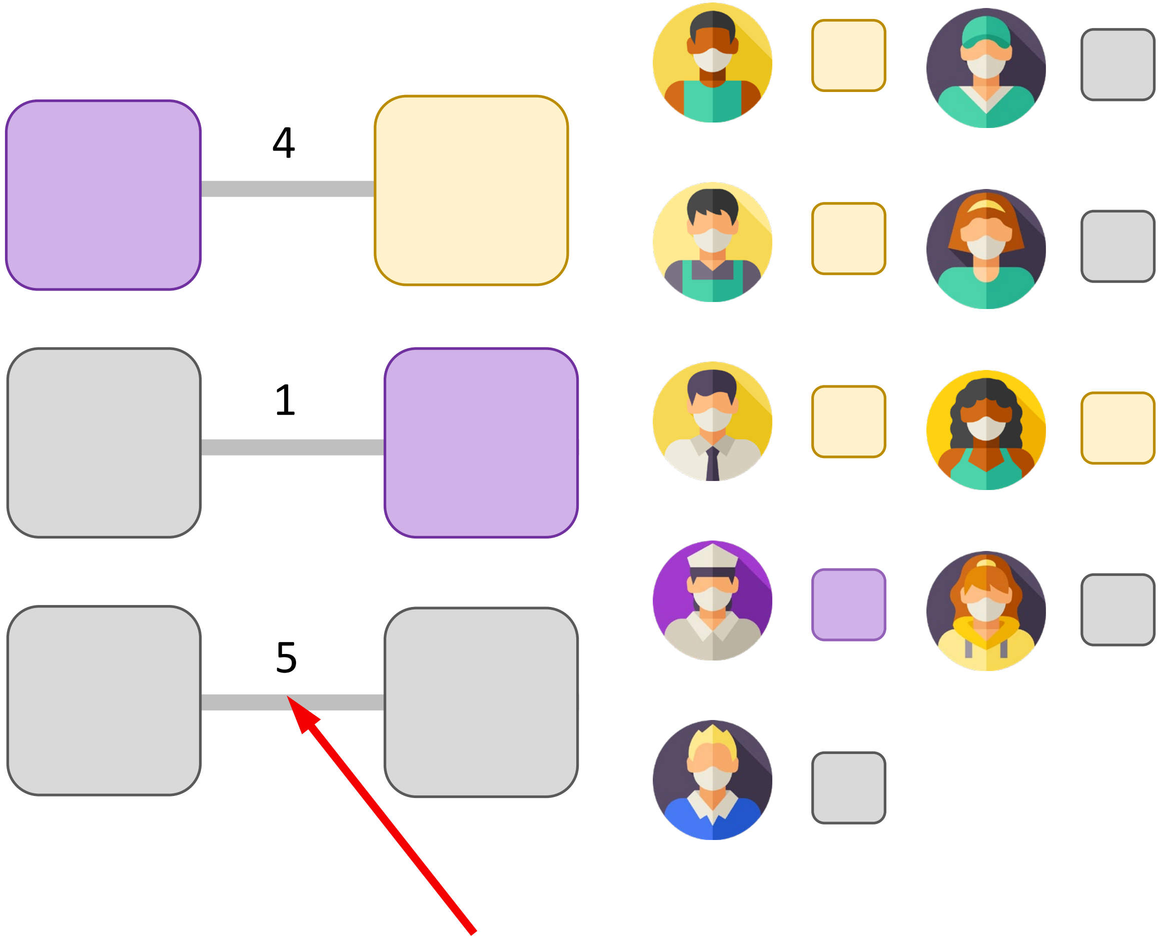
Summary graph $\bar{G}$

S : set of supernodes

P : set of superedges

w_{max} : maximum superedge weight





Size of summary graph: $|P|(2\log_2|S| + \log_2 \omega_{max}) + |V|\log_2|S|$

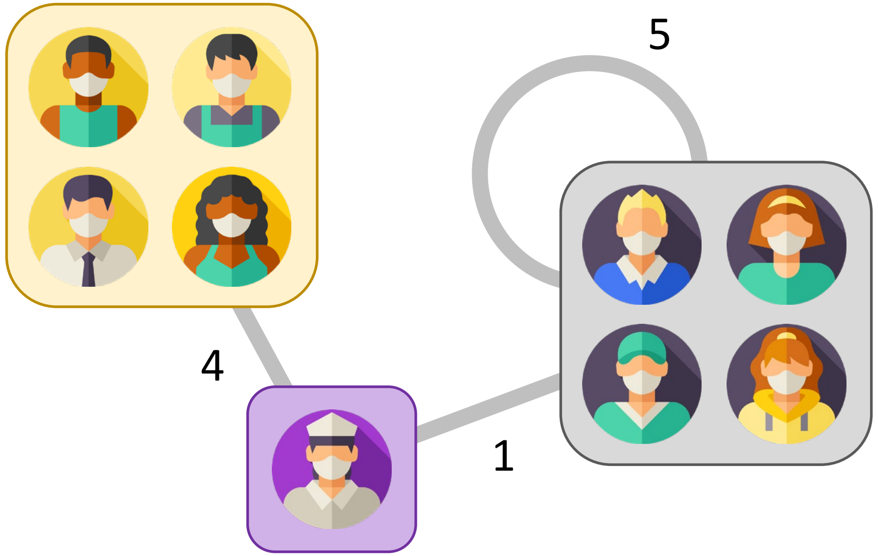
Details: Size in Bits of a Summary Graph

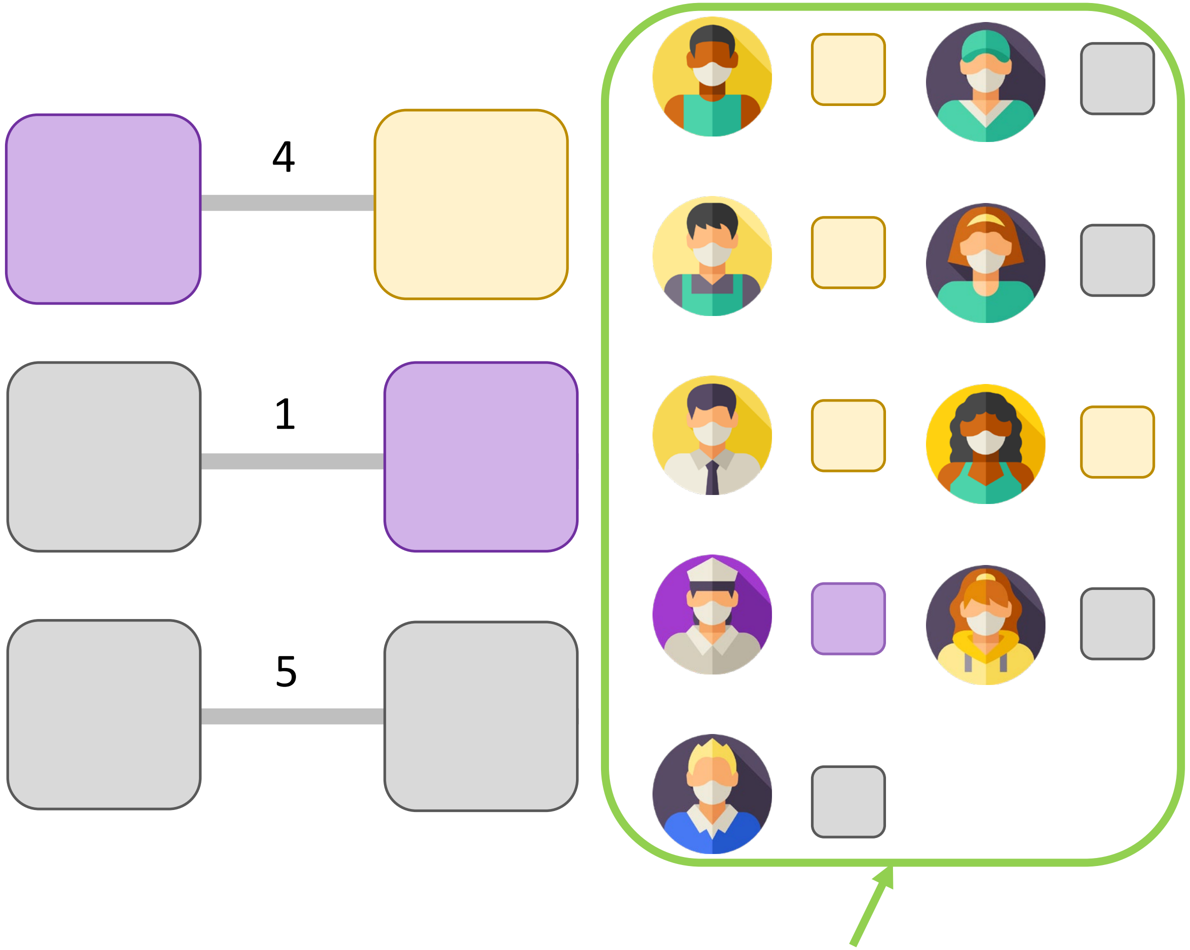
Summary graph $\overline{G}$

S : set of supernodes

P : set of superedges

w_{max} : maximum superedge weight





Size of summary graph: $|P|(2\log_2|S| + \log_2 \omega_{max}) + |V|\log_2|S|$

Details: Error Measurement

	1	2	3	4	5	6	7	8	9
1	0	1	0	0	0	1	0	0	1
2	1	0	1	0	0	1	1	0	0
3	0	1	0	1	1	1	0	0	1
4	0	0	1	0	1	0	0	0	0
5	0	0	1	1	0	1	1	0	0
6	1	1	1	0	1	0	0	0	0
7	0	1	0	0	1	0	0	1	1
8	0	0	0	0	0	0	1	0	1
9	1	0	1	0	0	0	1	1	0

Reconstructed Adjacency Matrix A

	1	2	3	4	5	6	7	8	9
1	0	1	3/8	3/8	3/8	3/8	1/3	1/3	1/3
2	1	0	3/8	3/8	3/8	3/8	1/3	1/3	1/3
3	3/8	3/8	0	5/6	5/6	5/6	0	0	0
4	3/8	3/8	5/6	0	5/6	5/6	0	0	0
5	3/8	3/8	5/6	5/6	0	5/6	0	0	0
6	3/8	3/8	5/6	5/6	5/6	0	0	0	0
7	1/3	1/3	0	0	0	0	0	1	1
8	1/3	1/3	0	0	0	0	1	0	1
9	1/3	1/3	0	0	0	0	1	1	0

Reconstructed Adjacency Matrix $\widehat{A}$

$$RE_p(\boldsymbol{A}, \widehat{\boldsymbol{A}}) = \left(\sum_{i=1}^{|V|} \sum_{j=1}^{|V|} \left| A(i,j) - \hat{A}(i,j) \right|^p \right)^{\frac{1}{p}}$$

Road Map

- Introduction
- Problem
- **Proposed Algorithm: SSumM <<**
- Experimental Results
- Conclusions

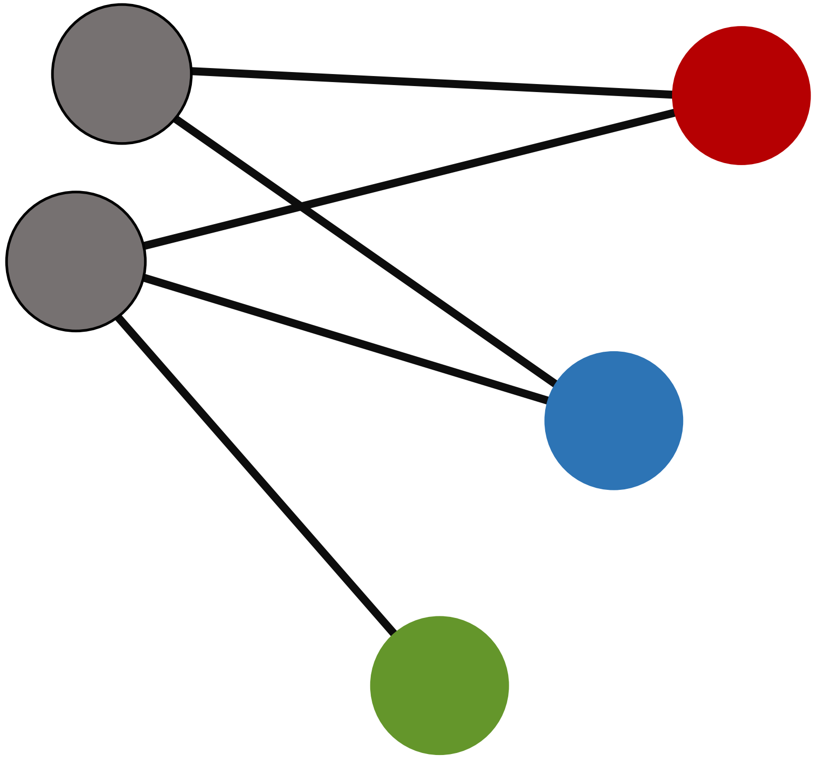
Main ideas of SSumM

- Practical graph summarization problem
  - *Given*: a graph $\boldsymbol{G}$
  - *Find*: a summary graph $\overline{\boldsymbol{G}}$
  - *To minimize*: the difference between $\boldsymbol{G}$ and the restored graph $\widehat{\boldsymbol{G}}$
  - *Subject to*: $Size\ of\ \overline{\boldsymbol{G}}$ in bits $\leq \boldsymbol{K}$

- Combines node grouping and edge sparsification
- Prunes search space
- Balances error and size of the summary graph using MDL principle

Main Idea: Combining Two Strategies

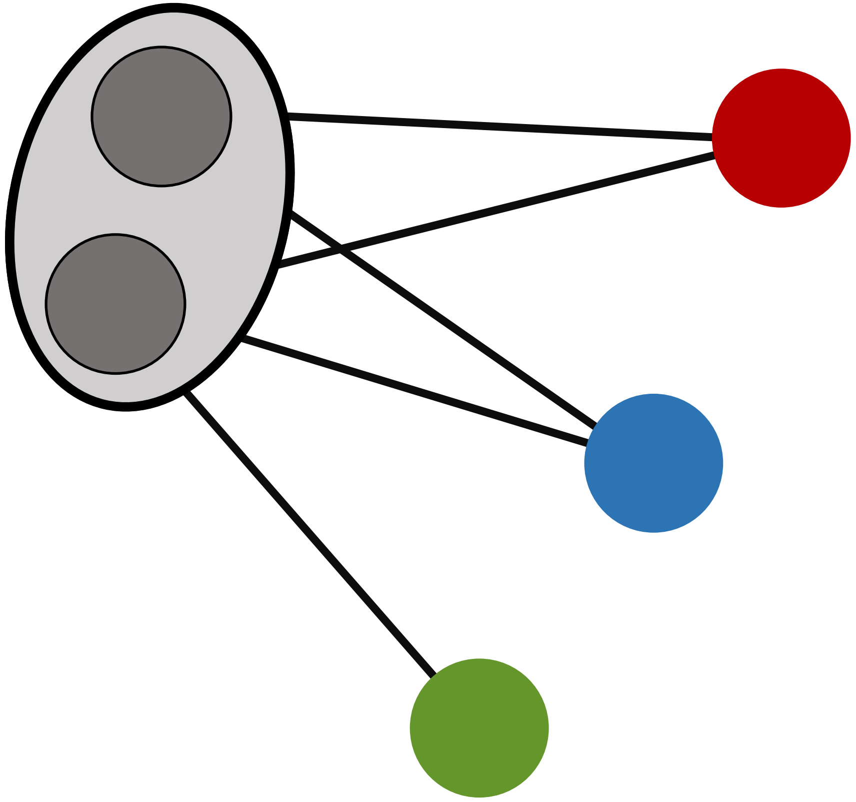
Node Grouping



Sparsification

Main Idea: Combining Two Strategies

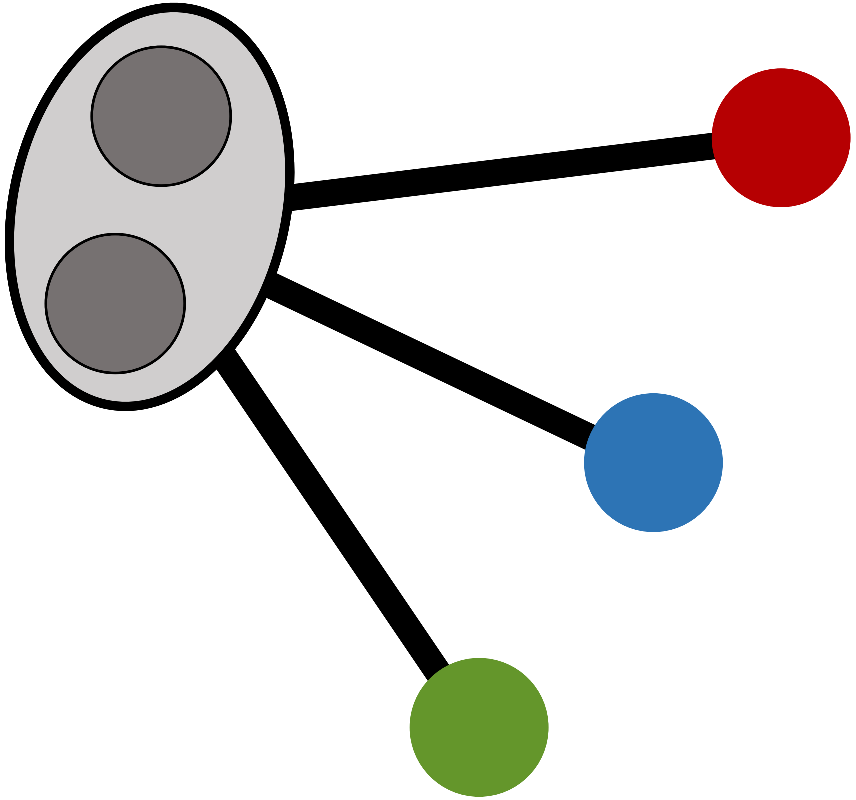
Node Grouping



Sparsification

Main Idea: Combining Two Strategies

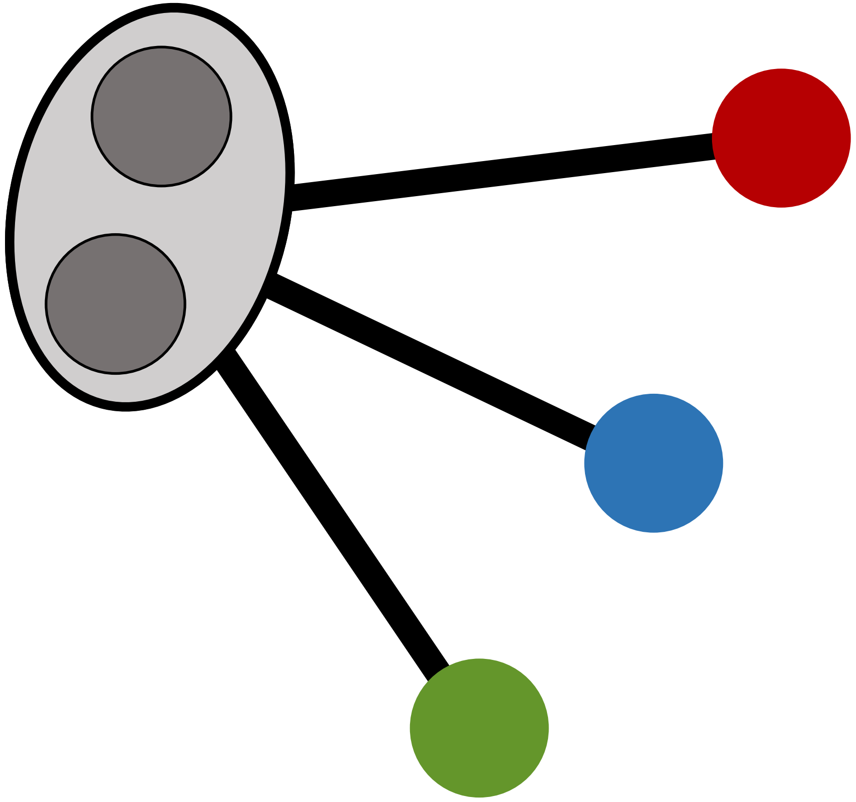
Node Grouping



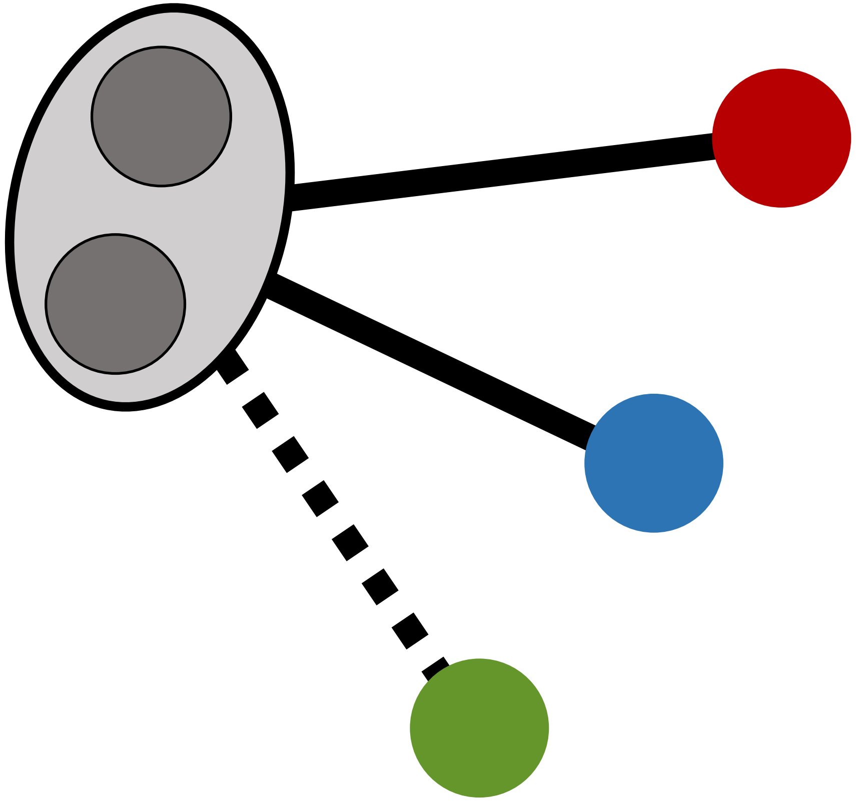
Sparsification

Main Idea: Combining Two Strategies

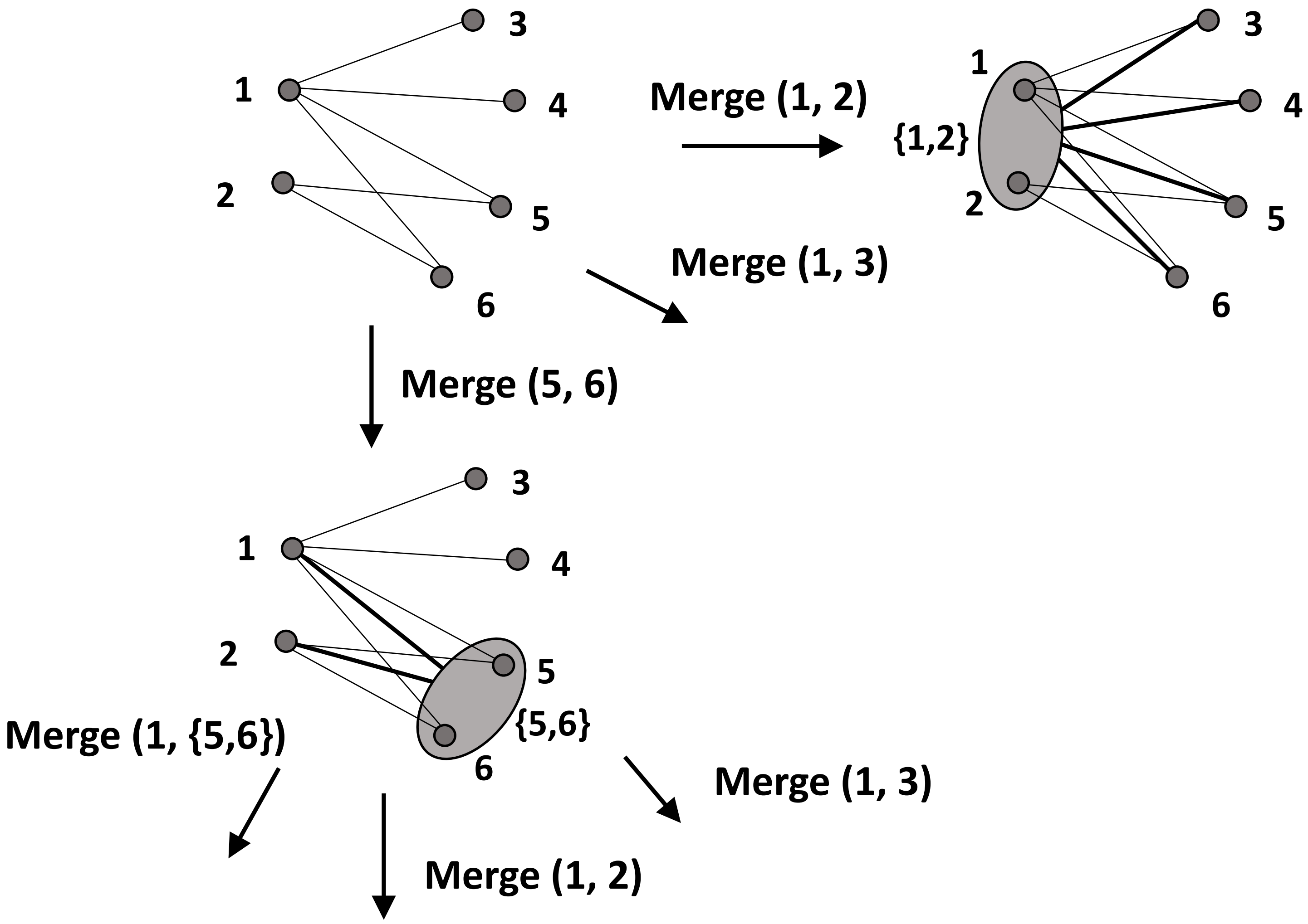
Node Grouping



Sparsification

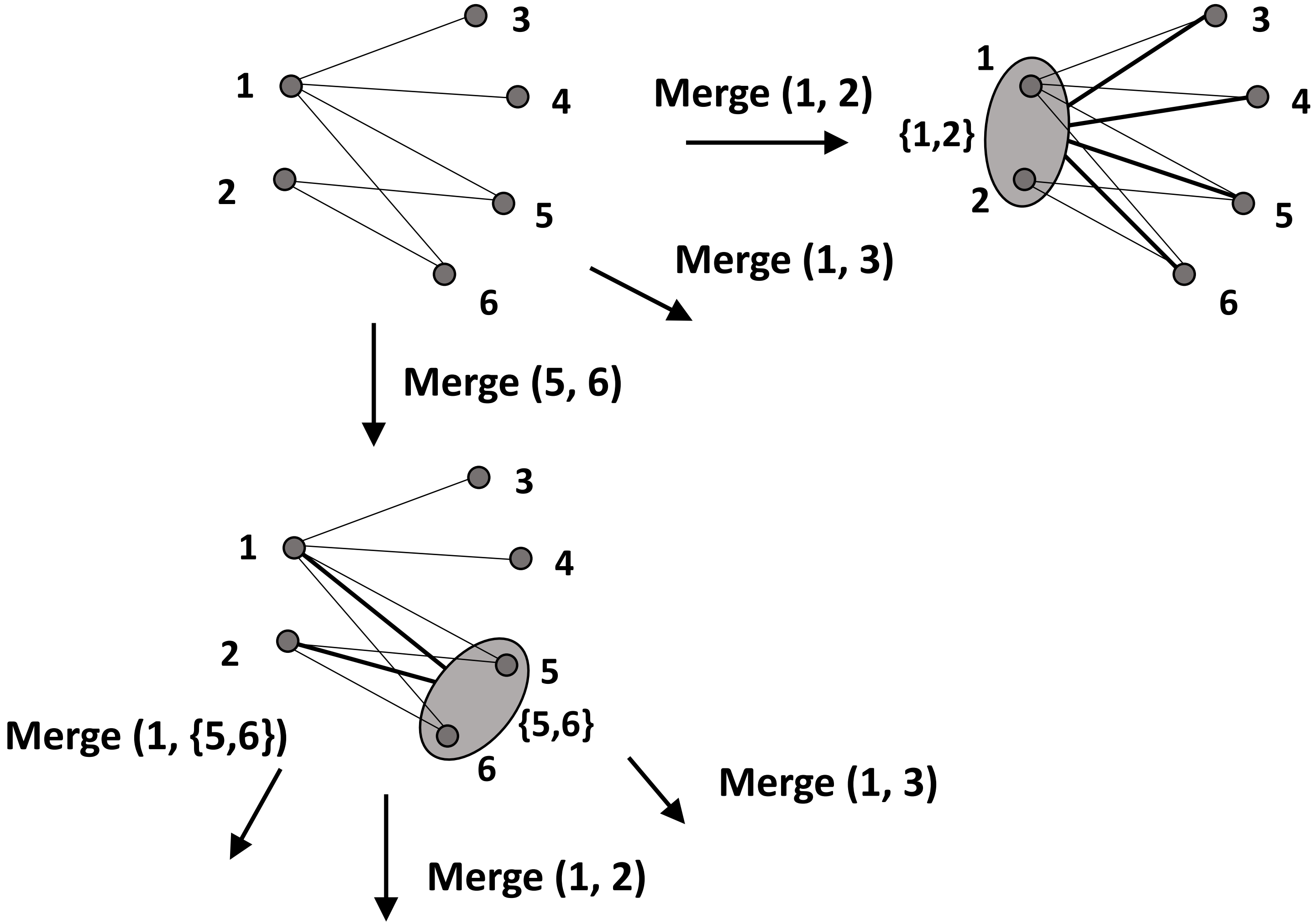


Main Idea: MDL Principle



How to choose a next action?

Main Idea: MDL Principle

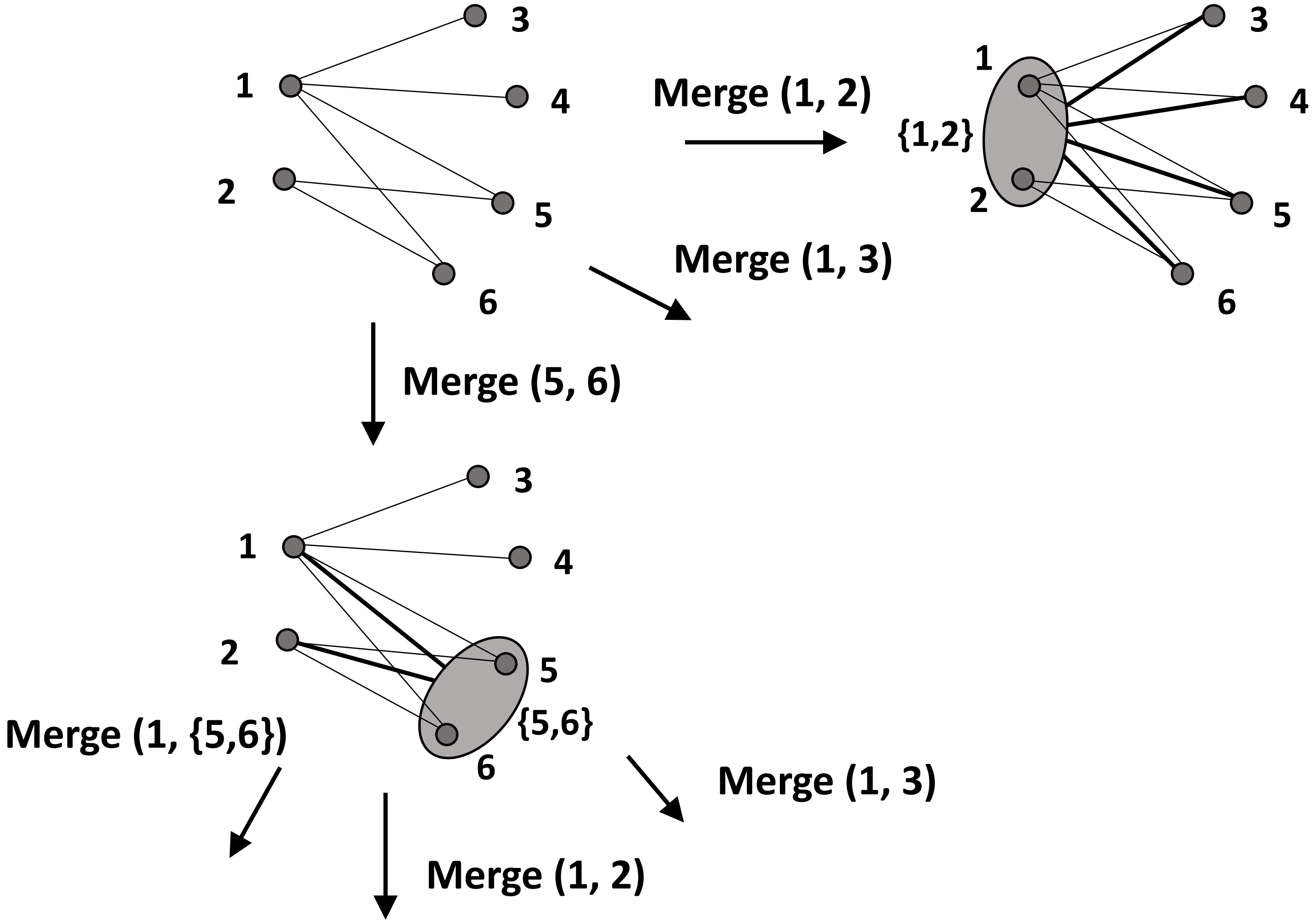


How to choose a next action?

Graph Summarization is

A Search Problem

Main Idea: MDL Principle



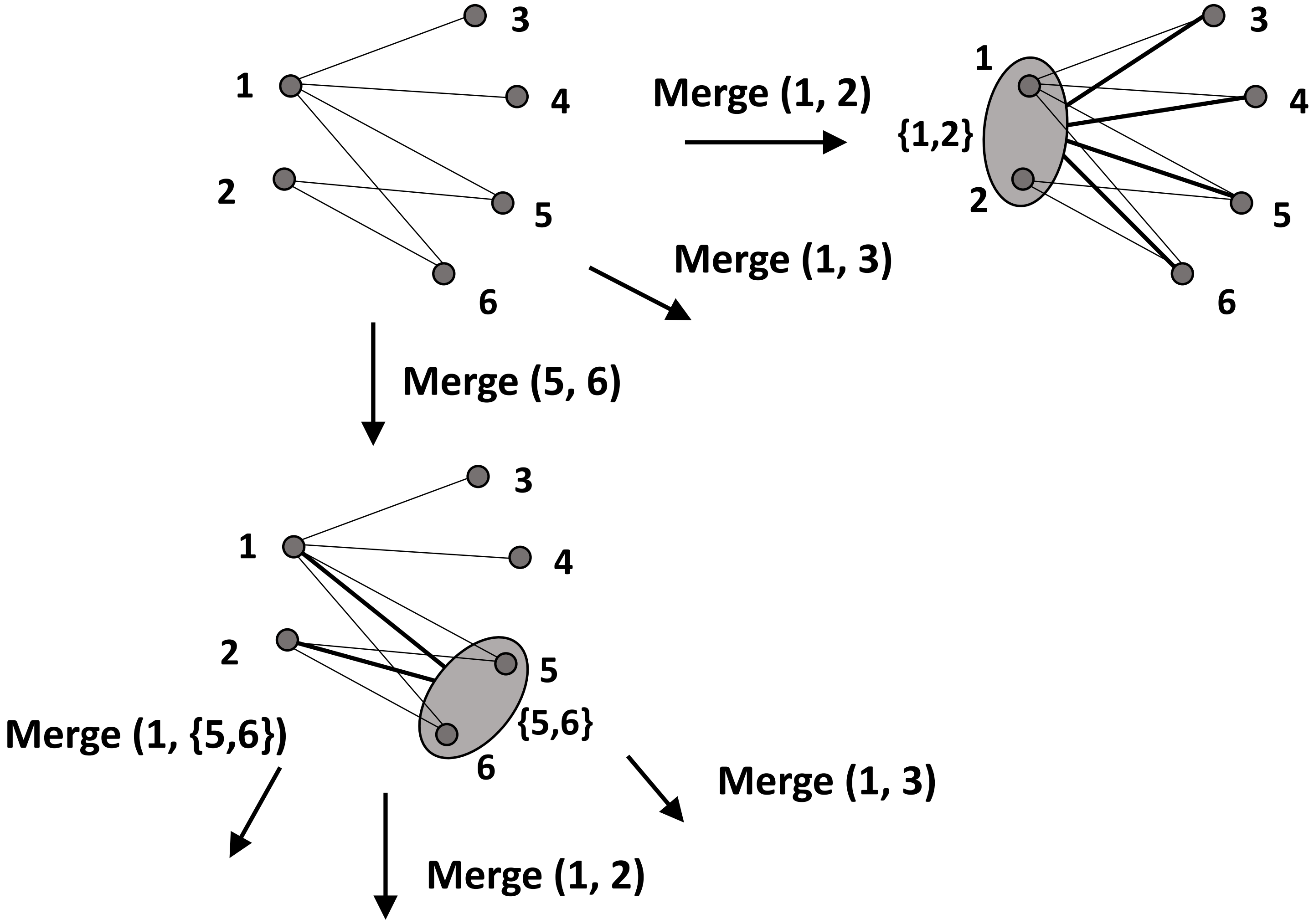
How to choose a next action?

Graph Summarization is

A Search Problem

Summary graph size + Information loss

Main Idea: MDL Principle



How to choose a next action?

Graph Summarization is

A Search Problem

Summary graph size + Information loss

⇩

MDL Principle

$$\arg\min_{\overline{G}} Cost(\overline{G}) + Cost(G|\overline{G})$$

$\# \, bits \, for \, \overline{G}$ $\# \, bits \, for \, representing \, G \, using \, \overline{G}$

Overview: SSumM

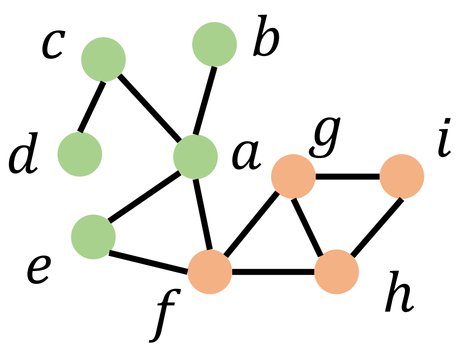
- ***Given:***
  - *(1) An input graph* $\boldsymbol{G}$, (2) *the desired size* $\boldsymbol{K}$, *(3) the number* $\boldsymbol{T}$ *of iterations*
- ***Outputs:***
  - *Summary graph* $\overline{\boldsymbol{G}}$

Procedure

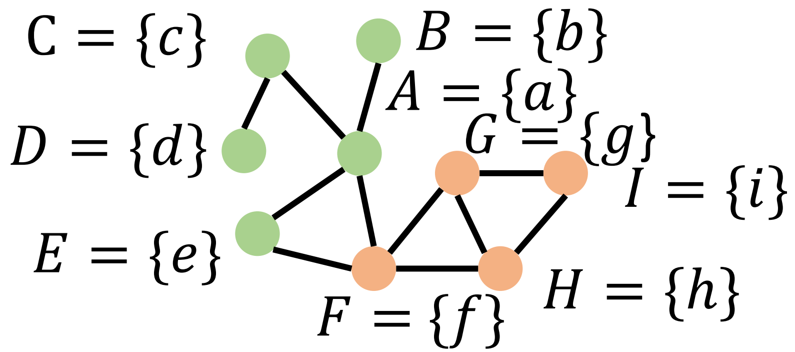
- Initialization phase
- $t = 1$
- While t++ ≤ $\boldsymbol{T}$ and $\boldsymbol{K}$ < size of $\overline{\boldsymbol{G}}$ in bits
  - Candidate generation phase
  - Merge and sparsification phase
- Further sparsification phase

Initialization Phase

Input graph G



Summary graph $\bar{G}$

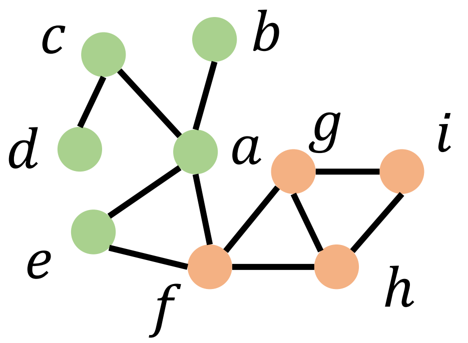


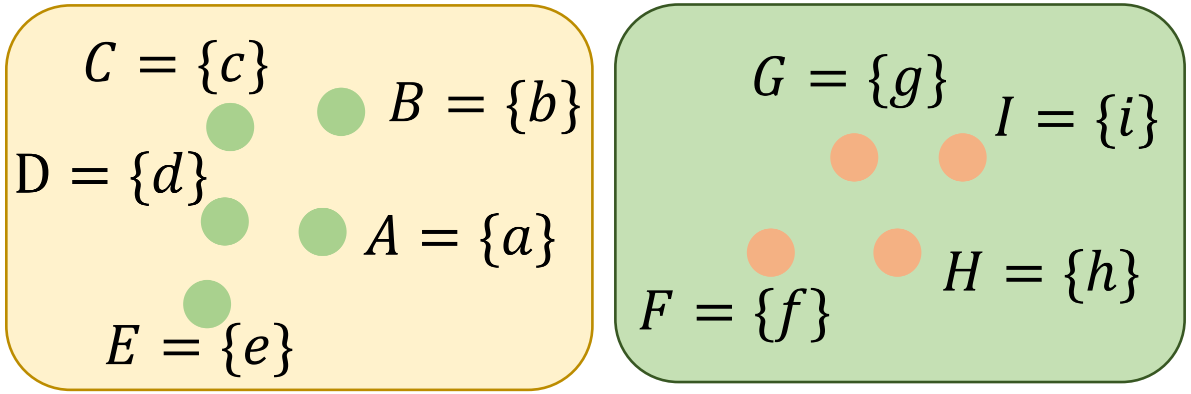
Procedure

- **Initialization phase <<**
- $t = 1$
- While t++ $\leq \boldsymbol{T}$ and $\boldsymbol{K}$ < size of $\bar{\boldsymbol{G}}$ in bits
  - Candidate generation phase
  - Merge and sparsification phase
- Further sparsification phase

Candidate Generation Phase

Input graph G



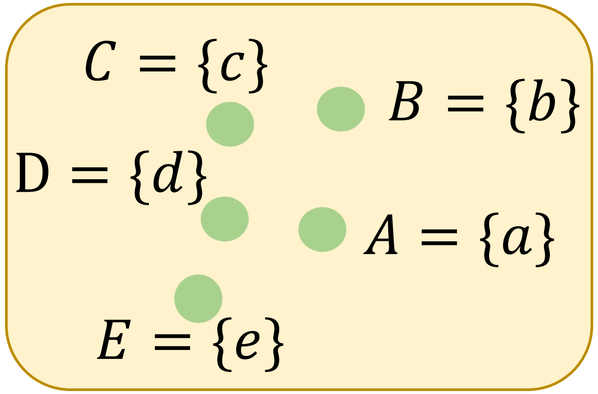


Procedure

- Initialization phase
- $t = 1$
- While t++ ≤ $\boldsymbol{T}$ and $\boldsymbol{K}$ < size of $\overline{\boldsymbol{G}}$ in bits
  - **Candidate generation phase <<**
  - Merge and sparsification phase
- Further sparsification phase

Merging and Sparsification Phase

For each candidate set $\boldsymbol{C}$



Among possible candidate pairs

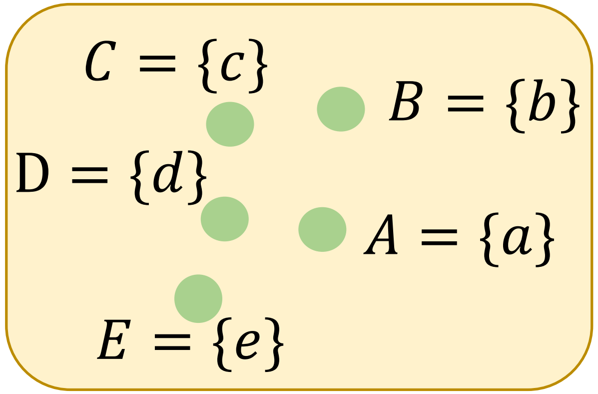
(A, B) (B, C) (C, D) (D, E)
(A, C) (B, D) (C, E)
(A, D) (B, E)
(A, E)

Procedure

- Initialization phase
- $t = 1$
- While t++ ≤ $\boldsymbol{T}$ and $\boldsymbol{K}$ < size of $\overline{\boldsymbol{G}}$ in bits
  - Candidate generation phase
  - **Merge and sparsification phase <<**
- Further sparsification phase

Merging and Sparsification Phase

For each candidate set $\boldsymbol{C}$



Among possible candidate pairs

(A, B) (B, C) (C, D) (D, E)
(A, C) (B, D) (C, E)
(A, D) (B, E)
(A, E)

Sample $\log_2 |\boldsymbol{C}|$ **pairs**

Procedure

- Initialization phase
- $t = 1$
- While $t{+}{+} \leq \boldsymbol{T}$ and $\boldsymbol{K}$ < size of $\overline{\boldsymbol{G}}$ in bits
  - Candidate generation phase
  - **Merge and sparsification phase <<**
- Further sparsification phase

Merging and Sparsification Phase

Select the pair with
the greatest (relative) reduction in the cost function

(A, B) (A, D) (C, D)

$\boldsymbol{if}$ reduction(C, D) > $\boldsymbol{\theta}$:
  merge(C, D)
$\boldsymbol{else}$
  sample $\log_2 |\boldsymbol{C}|$ pairs again

Procedure

- Initialization phase
- $t = 1$
- While t++ ≤ $\boldsymbol{T}$ and $\boldsymbol{K}$ < size of $\overline{\boldsymbol{G}}$ in bits
  - Candidate generation phase
  - **Merge and sparsification phase <<**
- Further sparsification phase

Merging and Sparsification Phase

Select the pair with
the greatest (relative) reduction in the cost function

(A, B) (A, D) $\boxed{\text{(C, D)}}$

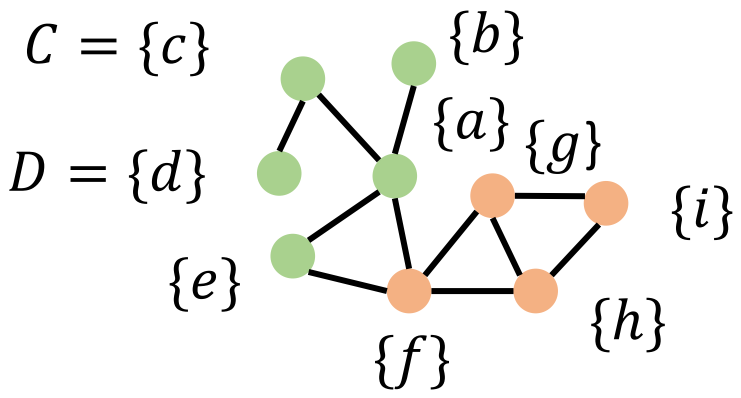
if reduction(C, D) > $\boldsymbol{\theta}$:
    merge(C, D)
else
    sample $\log_2 |\boldsymbol{C}|$ pairs again

Procedure

- Initialization phase
- $t = 1$
- While t++ ≤ $\boldsymbol{T}$ and $\boldsymbol{K}$ < size of $\overline{\boldsymbol{G}}$ in bits
    - Candidate generation phase
    - **Merge and sparsification phase <<**
- Further sparsification phase

Merging and Sparsification Phase (cont.)

Summary graph $\bar{G}$

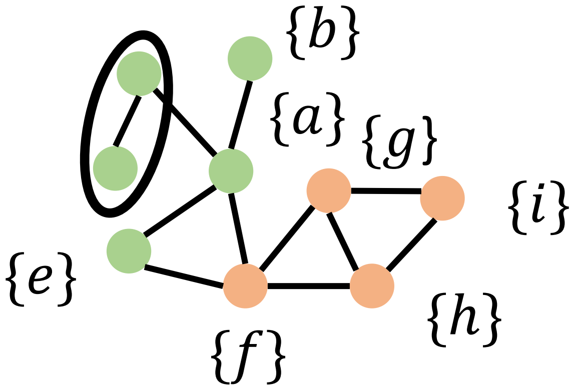


Procedure

- Initialization phase
- $t = 1$
- While t++ ≤ $\boldsymbol{T}$ and $\boldsymbol{K}$ < size of $\bar{\boldsymbol{G}}$ in bits
  - Candidate generation phase
  - **Merge and sparsification phase <<**
- Further sparsification phase

Merging and Sparsification Phase (cont.)

Summary graph $\bar{G}$

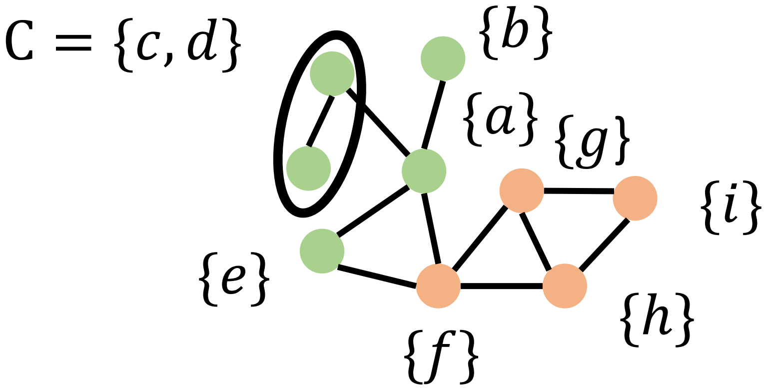


Procedure

- Initialization phase
- $t = 1$
- While t++ ≤ $\boldsymbol{T}$ and $\boldsymbol{K}$ < size of $\bar{\boldsymbol{G}}$ in bits
  - Candidate generation phase
  - **Merge and sparsification phase <<**
- Further sparsification phase

Merging and Sparsification Phase (cont.)

Summary graph $\bar{G}$

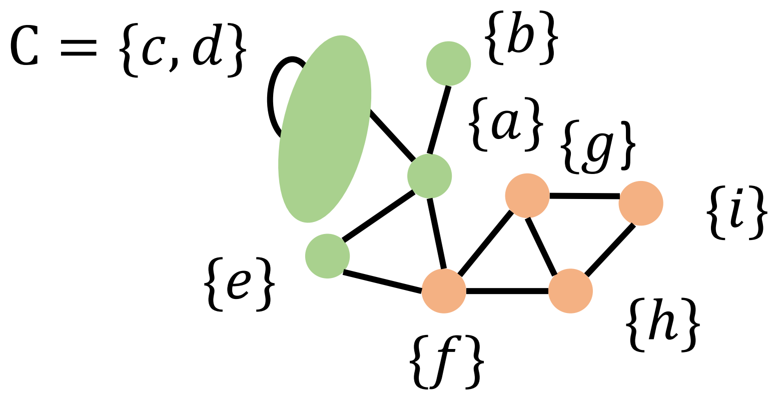


Procedure

- Initialization phase
- $t = 1$
- While t++ ≤ $\boldsymbol{T}$ and $\boldsymbol{K}$ < size of $\bar{\boldsymbol{G}}$ in bits
  - Candidate generation phase
  - **Merge and sparsification phase <<**
- Further sparsification phase

Merging and Sparsification Phase (cont.)

Summary graph $\overline{G}$

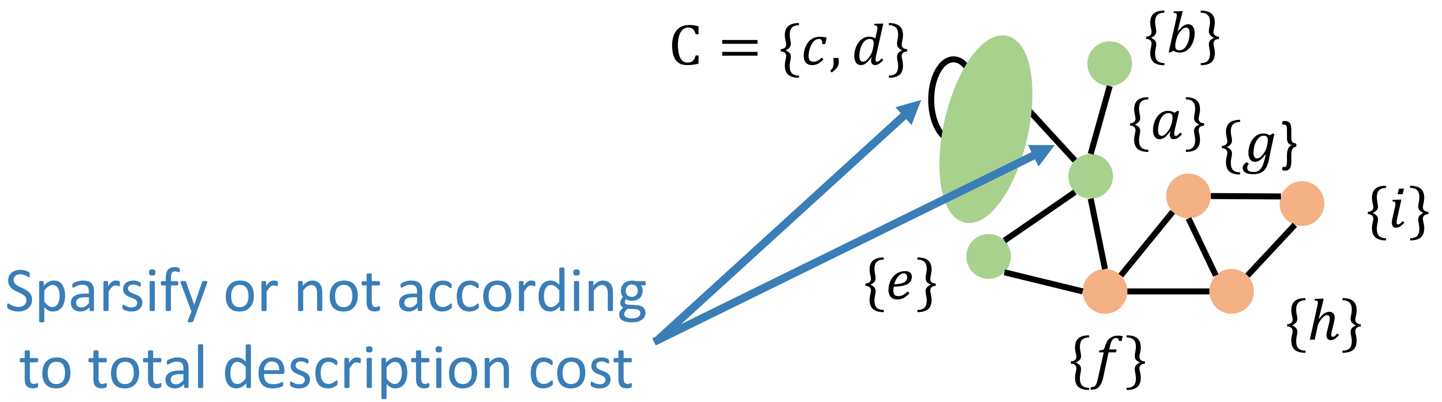


Procedure

- Initialization phase
- $t = 1$
- While t++ ≤ $\boldsymbol{T}$ and $\boldsymbol{K}$ < size of $\overline{\boldsymbol{G}}$ in bits
  - Candidate generation phase
  - **Merge and sparsification phase <<**
- Further sparsification phase

Merging and Sparsification Phase (cont.)

Summary graph $\bar{G}$



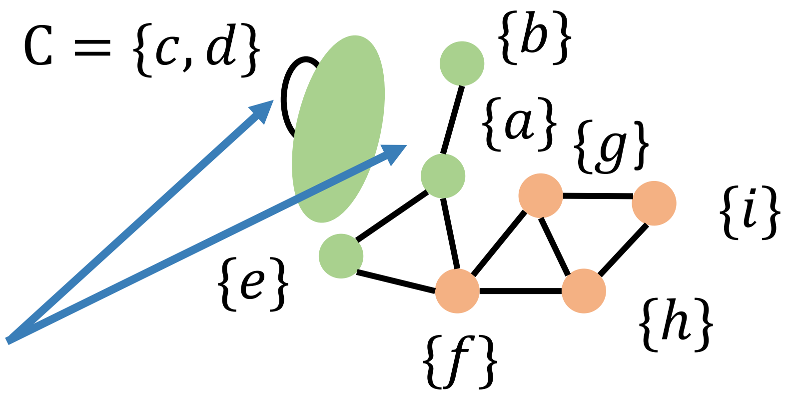
Procedure

- Initialization phase
- $t = 1$
- While t++ ≤ $\boldsymbol{T}$ and $\boldsymbol{K}$ < size of $\bar{\boldsymbol{G}}$ in bits
  - Candidate generation phase
  - **Merge and sparsification phase <<**
- Further sparsification phase

Merging and Sparsification Phase (cont.)

Summary graph $\overline{G}$

Sparsify or not according to total description cost

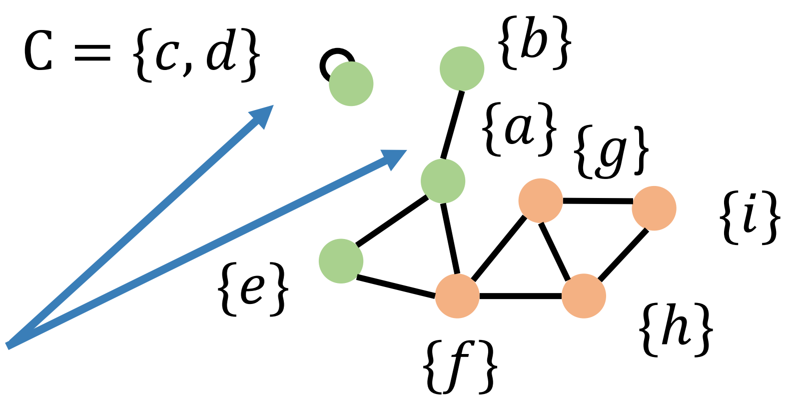


Procedure

- Initialization phase
- $t = 1$
- While t++ $\leq \boldsymbol{T}$ and $\boldsymbol{K}$ < size of $\overline{\boldsymbol{G}}$ in bits
  - Candidate generation phase
  - **Merge and sparsification phase <<**
- Further sparsification phase

Merging and Sparsification Phase (cont.)

Summary graph $\overline{G}$



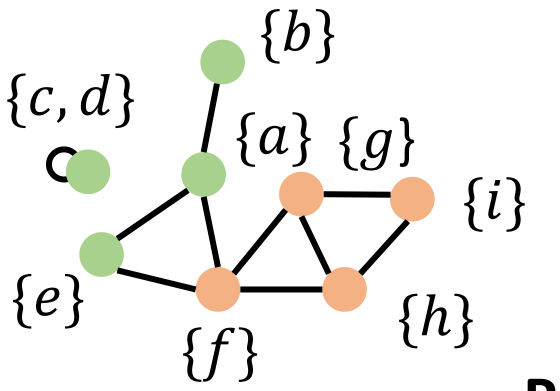
Sparsify or not according to total description cost

Procedure

- Initialization phase
- $t = 1$
- While t++ $\leq \boldsymbol{T}$ and $\boldsymbol{K}$ < size of $\overline{\boldsymbol{G}}$ in bits
  - Candidate generation phase
  - **Merge and sparsification phase <<**
- Further sparsification phase

Repetition

Summary graph $\bar{G}$

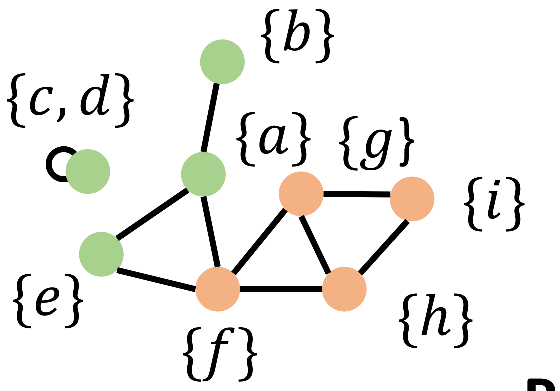


Procedure

- Initialization phase
- $t = 1$
- **While t++ $\leq T$ and K < size of $\bar{G}$ in bits**
  - Candidate generation phase
  - Merge and sparsification phase
- Further sparsification phase

Repetition

Summary graph $\bar{G}$



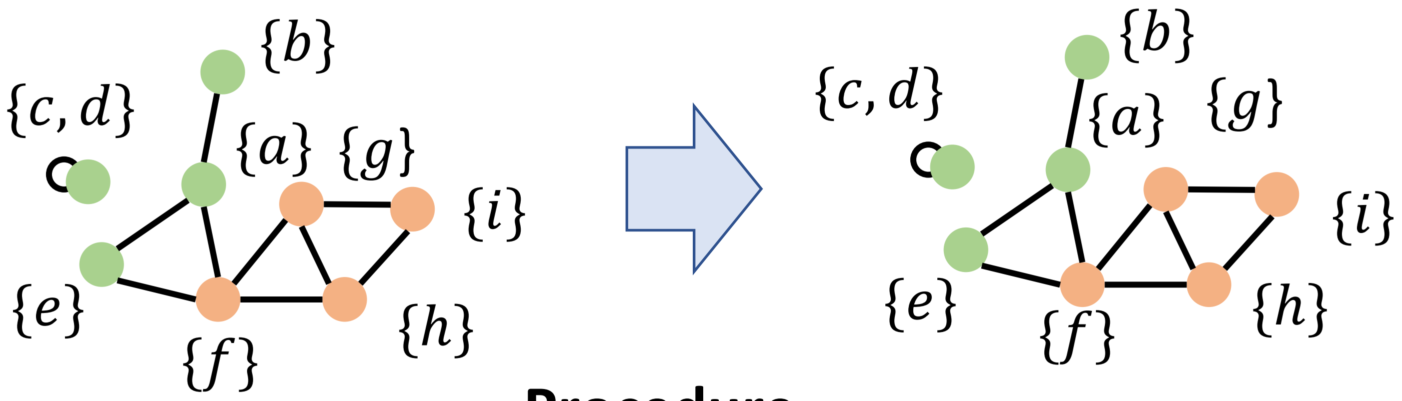
Different candidate sets and decreasing threshold θ over iteration

Procedure

- Initialization phase
- $t = 1$
- **While t++ ≤ T and K < size of $\bar{G}$ in bits**
  - Candidate generation phase
  - Merge and sparsification phase
- Further sparsification phase

Repetition

Summary graph $\overline{G}$ **Summary graph $\overline{G}$**



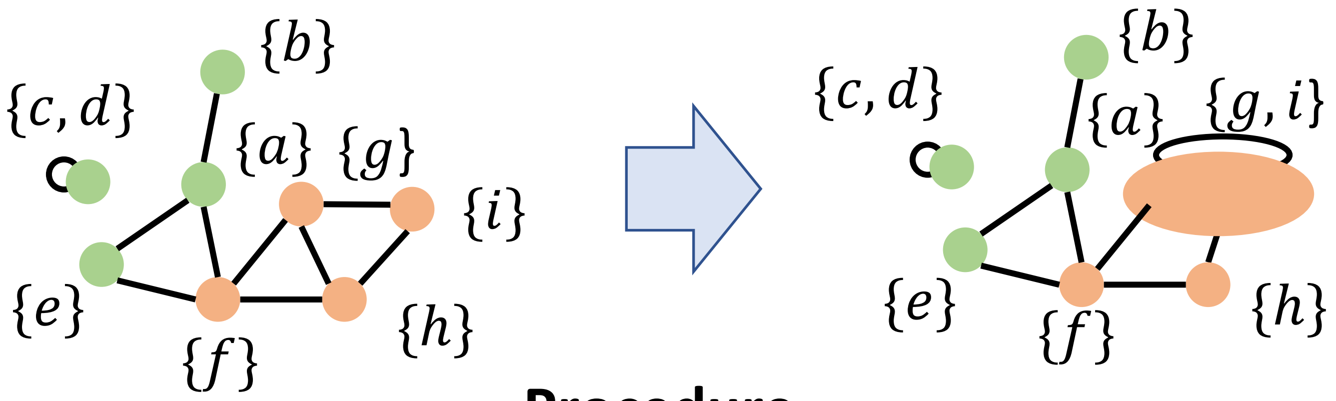
Procedure

- Initialization phase
- $t = 1$
- **While t++ ≤ T and K < size of $\overline{G}$ in bits**
  - Candidate generation phase
  - Merge and sparsification phase
- Further sparsification phase

Different candidate sets and decreasing threshold θ over iteration

Repetition

Summary graph $\bar{G}$ **Summary graph $\bar{G}$**



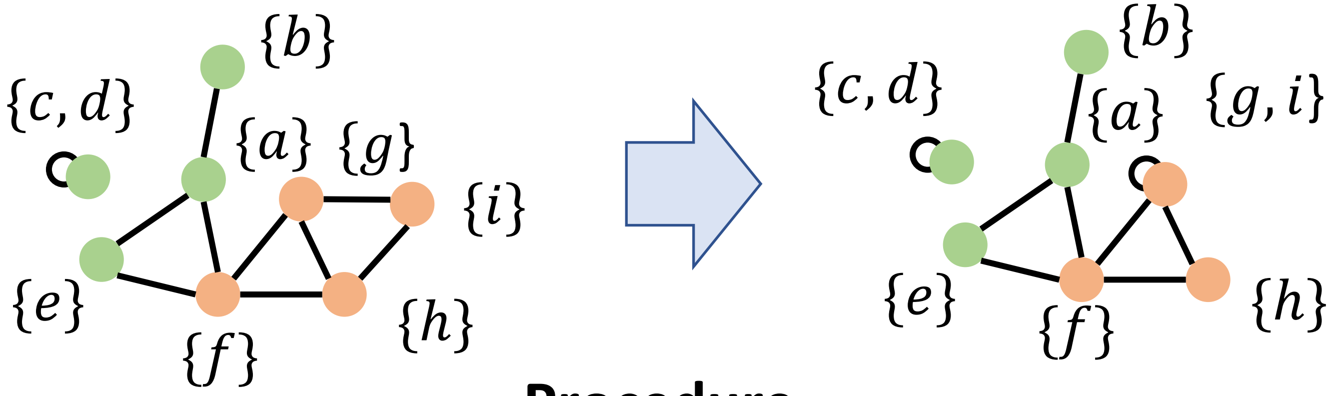
Procedure

- Initialization phase
- $t = 1$
- **While t++ ≤ T and K < size of $\bar{G}$ in bits**
  - Candidate generation phase
  - Merge and sparsification phase
- Further sparsification phase

Different candidate sets and decreasing threshold θ over iteration

Repetition

Summary graph $\bar{G}$ **Summary graph $\bar{G}$**



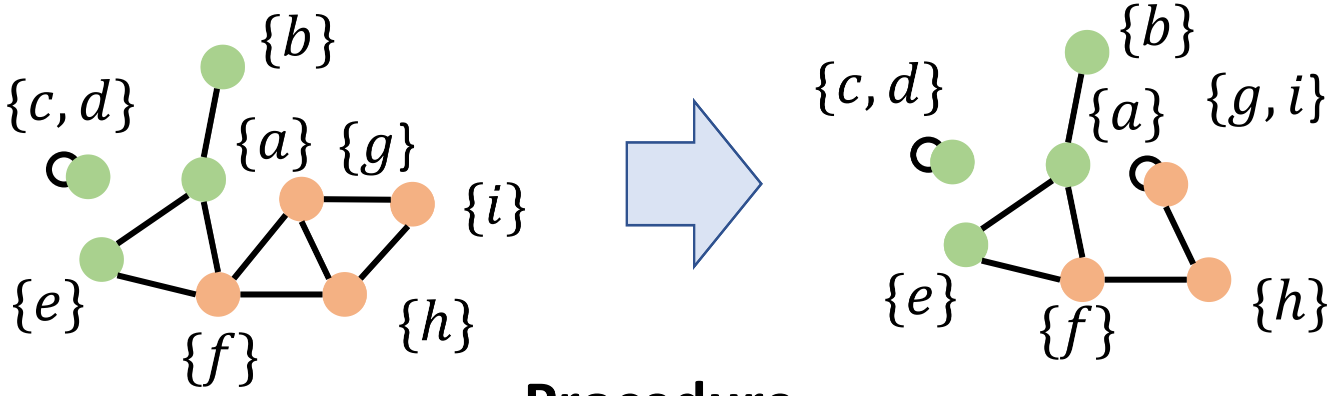
Different candidate sets and decreasing threshold θ over iteration

Procedure

- Initialization phase
- $t = 1$
- **While t++ ≤ T and K < size of $\bar{G}$ in bits**
  - Candidate generation phase
  - Merge and sparsification phase
- Further sparsification phase

Repetition

Summary graph $\bar{G}$ **Summary graph $\bar{G}$**



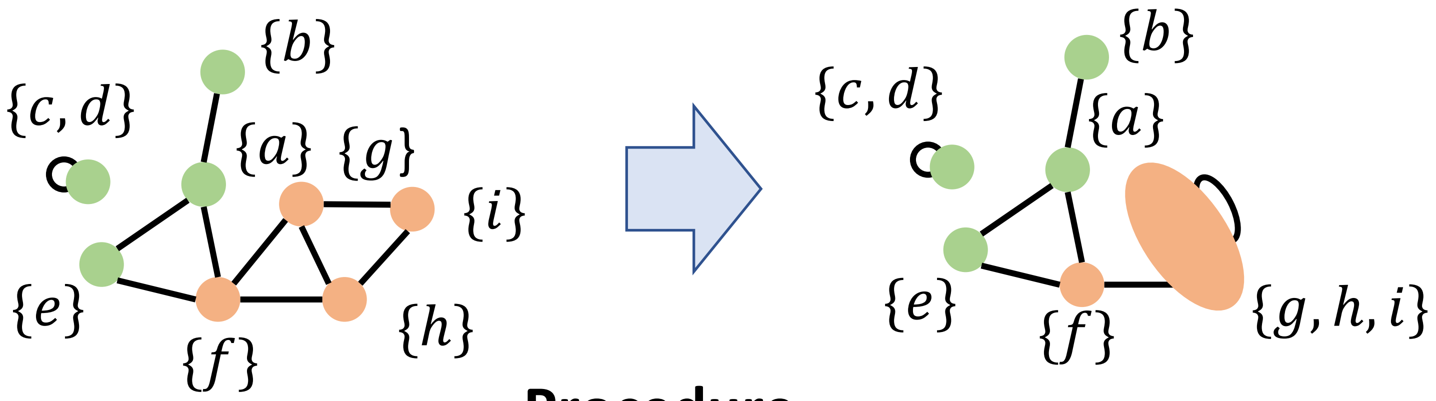
Different candidate sets and decreasing threshold θ over iteration

Procedure

- Initialization phase
- $t = 1$
- **While t++ ≤ T and K < size of $\bar{G}$ in bits**
  - Candidate generation phase
  - Merge and sparsification phase
- Further sparsification phase

Repetition

Summary graph $\bar{G}$ **Summary graph $\bar{G}$**



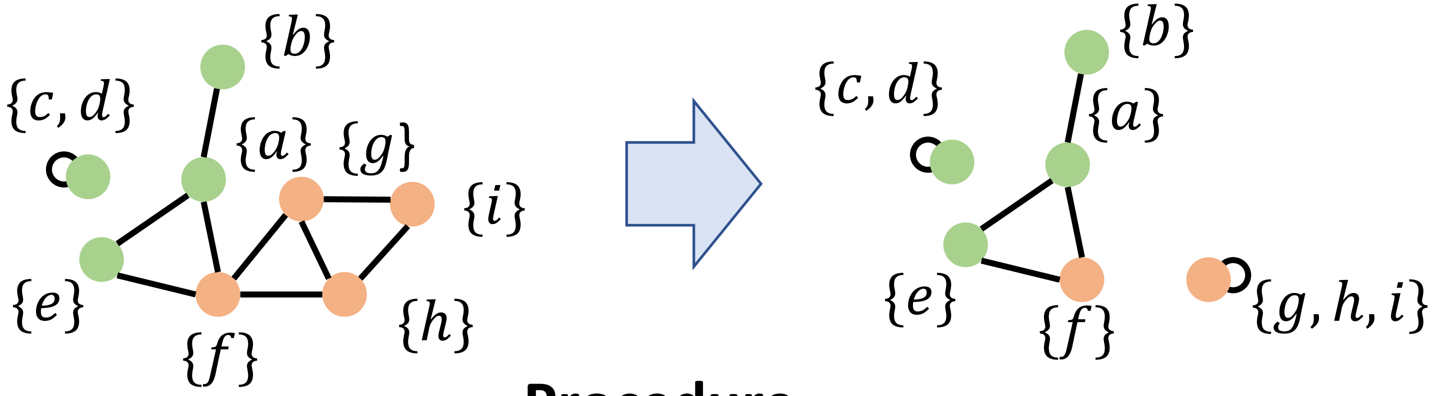
Different candidate sets and decreasing threshold θ over iteration

Procedure

- Initialization phase
- $t = 1$
- **While t++ $\leq T$ and K < size of $\bar{G}$ in bits**
  - Candidate generation phase
  - Merge and sparsification phase
- Further sparsification phase

Repetition

Summary graph $\bar{G}$ **Summary graph $\bar{G}$**

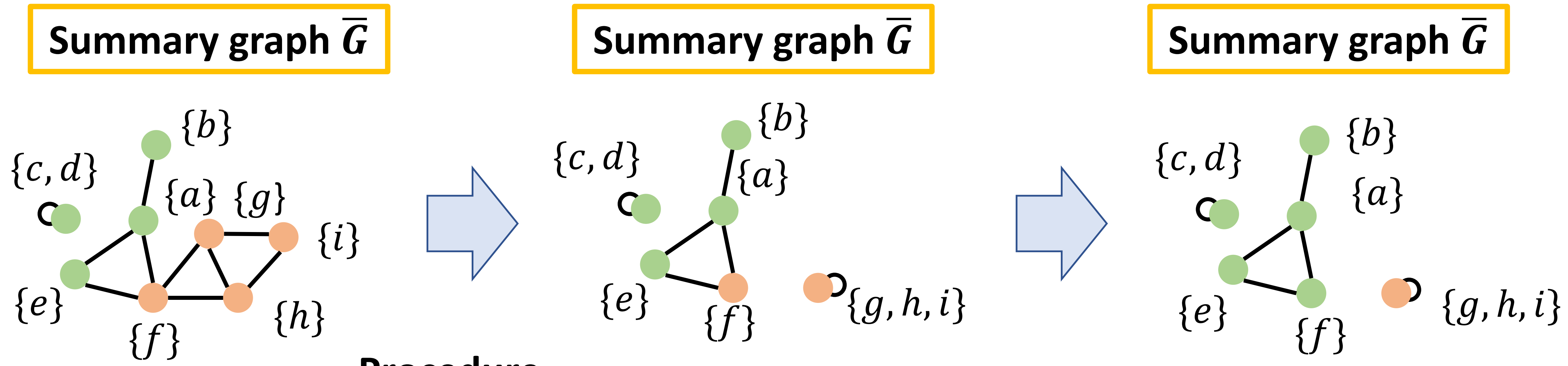


Procedure

- Initialization phase
- $t = 1$
- **While t++ ≤ T and K < size of $\bar{G}$ in bits**
  - Candidate generation phase
  - Merge and sparsification phase
- Further sparsification phase

Different candidate sets and decreasing threshold θ over iteration

Repetition

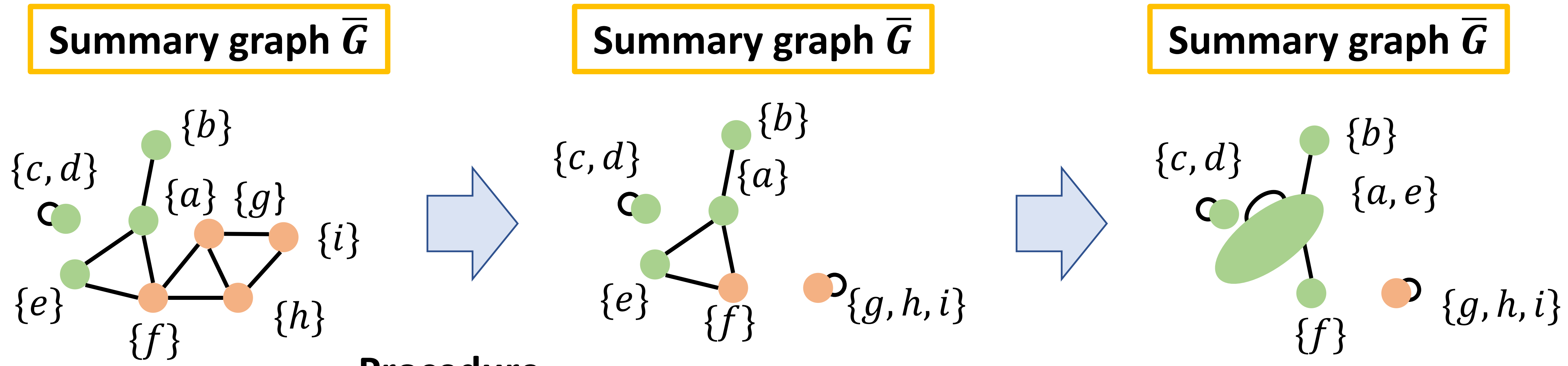


Procedure

- Initialization phase
- $t = 1$
- **While t++ ≤ T and K < size of $\bar{G}$ in bits**
  - Candidate generation phase
  - Merge and sparsification phase
- Further sparsification phase

Different candidate sets and decreasing threshold θ over iteration

Repetition



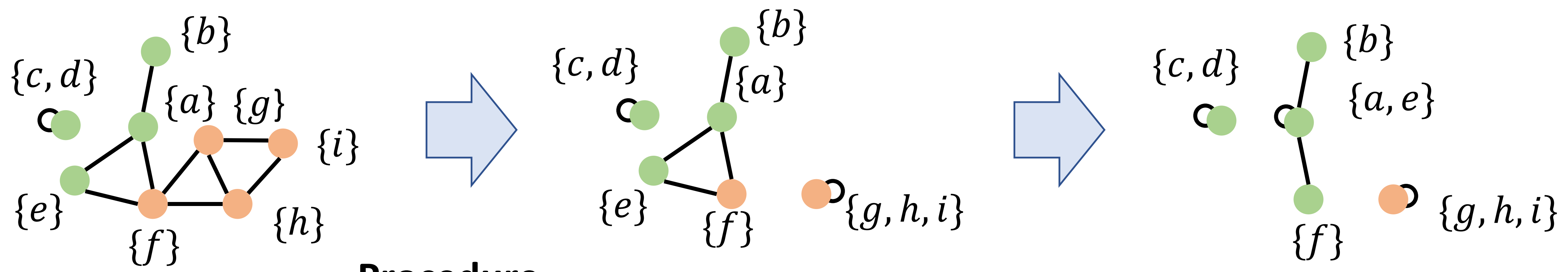
Different candidate sets and decreasing threshold θ over iteration

Procedure

- Initialization phase
- $t = 1$
- **While t++ ≤ T and K < size of $\bar{G}$ in bits**
  - Candidate generation phase
  - Merge and sparsification phase
- Further sparsification phase

Repetition

Summary graph $\bar{G}$ **Summary graph $\bar{G}$** **Summary graph $\bar{G}$**



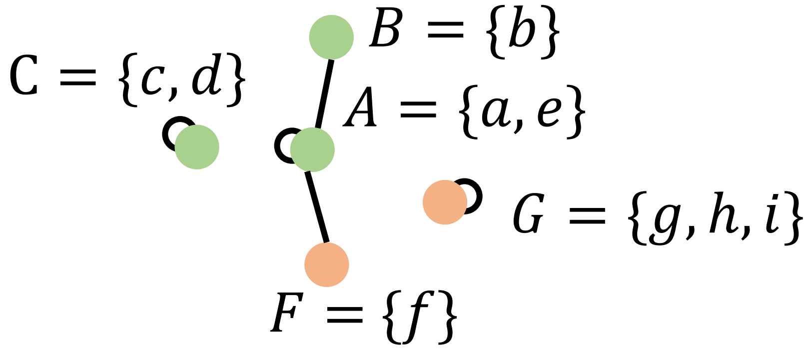
Procedure

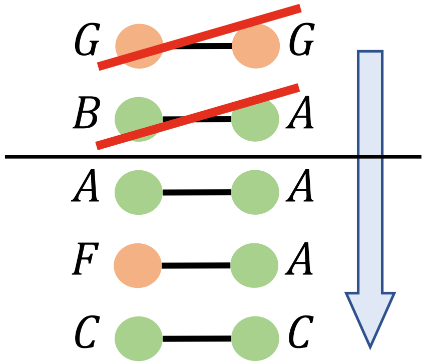
- Initialization phase
- $t = 1$
- **While t++ $\leq T$ and K < size of $\bar{G}$ in bits**
  - Candidate generation phase
  - Merge and sparsification phase
- Further sparsification phase

Different candidate sets and decreasing threshold θ over iteration

Further Sparsification Phase

Summary graph $\bar{G}$





Size of $\bar{\boldsymbol{G}}$ in bits $\leq \boldsymbol{K}$

Superedges sorted by ΔRE_p

Procedure

- Initialization phase
- $t = 1$
- While $t{+}{+} \leq \boldsymbol{T}$ and $\boldsymbol{K} <$ size of $\bar{\boldsymbol{G}}$ in bits
  - Candidate generation phase
  - Merge and sparsification phase
- **Further sparsification phase <<**

Road Map

- Introduction
- Problem
- Proposed Algorithm: SSumM
- **Experimental Results <<**
- Conclusions

Experiments Settings

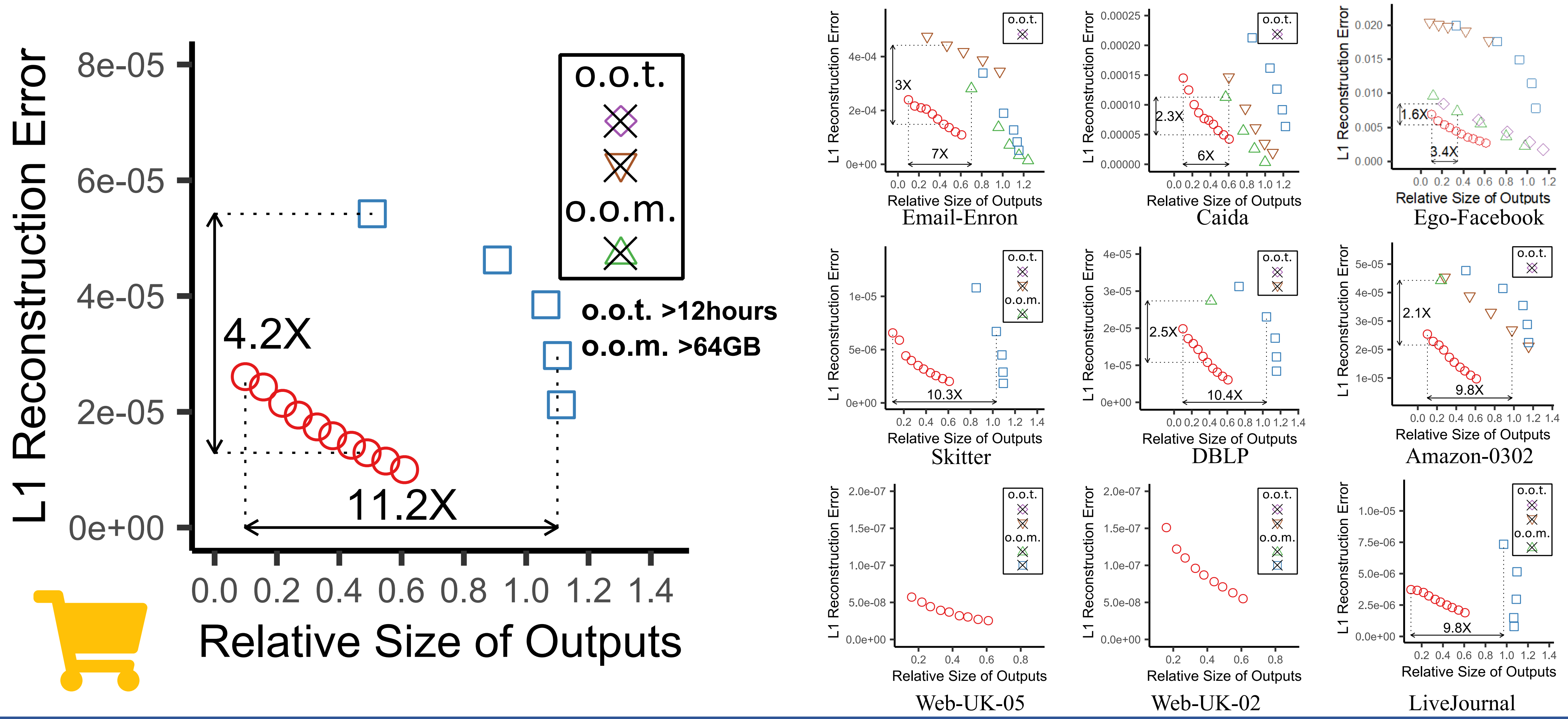
- 10 datasets from 6 domains (up to 0.8B edges)

Social Internet Email Co-purchase Collaboration Hyperlinks

- Three competitors for graph summarization
  - k-Gs [LT10]
  - S2L [RSB17]
  - SAA-Gs [BAZK18]

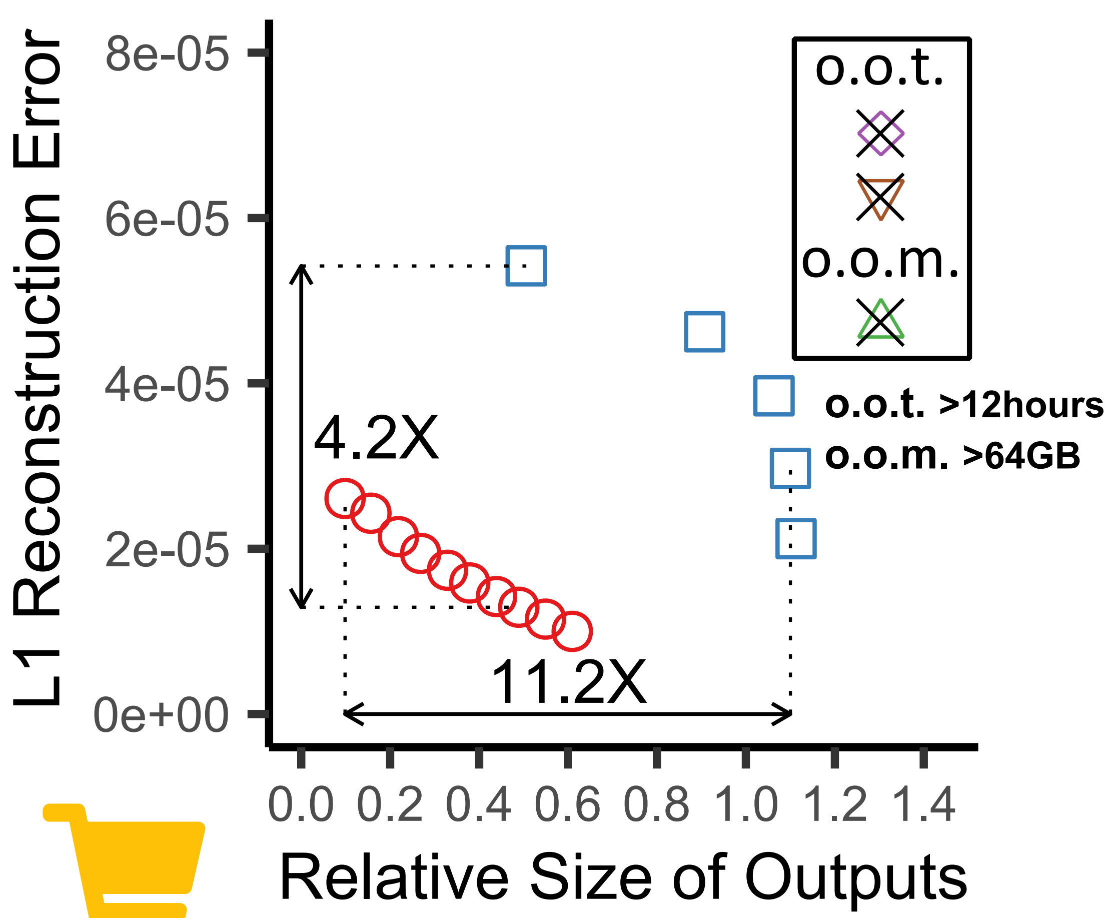
SSumM Gives Concise and Accurate Summary

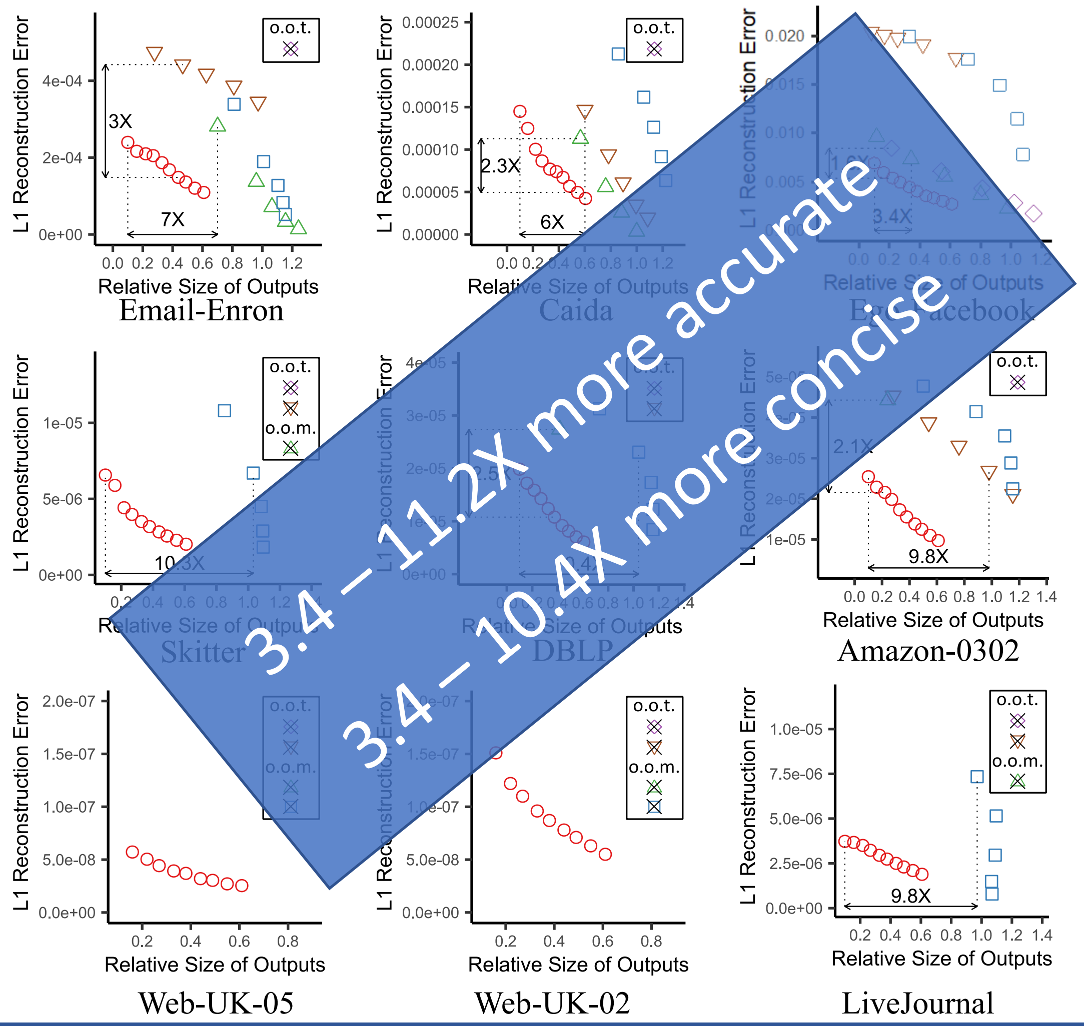
SSumM SAA-Gs SAA-Gs (linear sample) S2L k-Gs



SSumM Gives Concise and Accurate Summary

SSumM SAA-Gs SAA-Gs (linear sample) S2L k-Gs



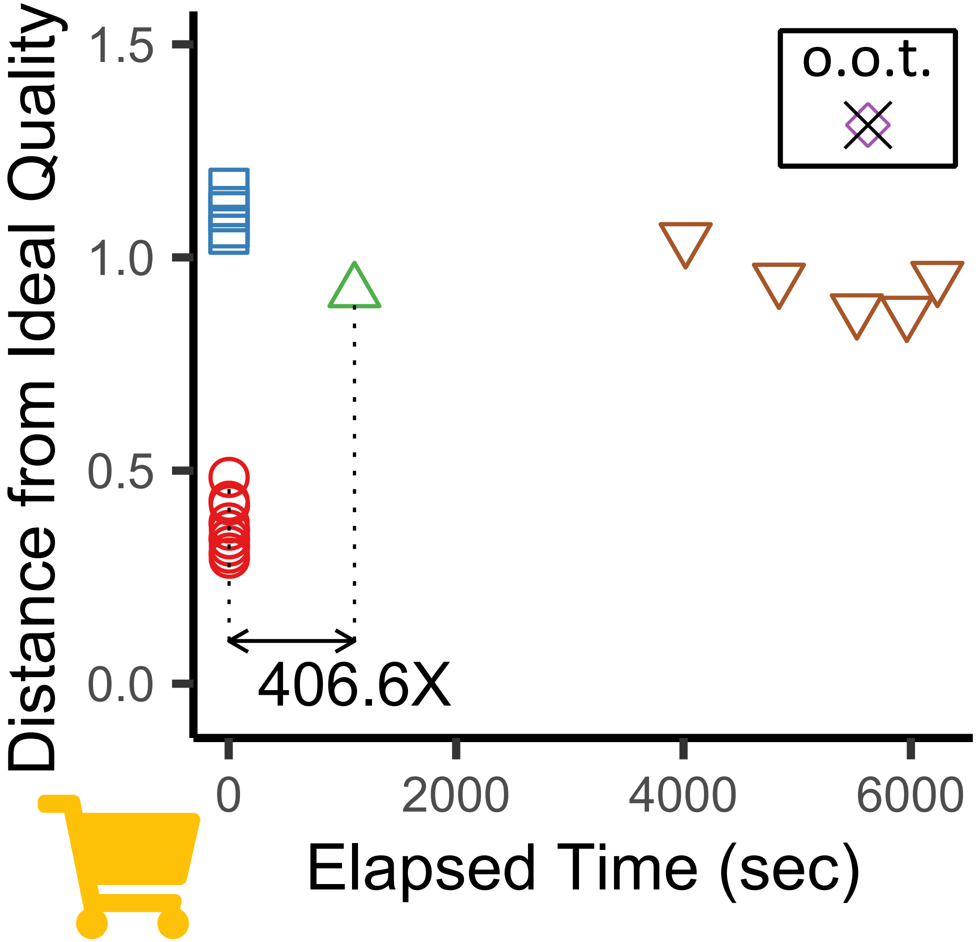


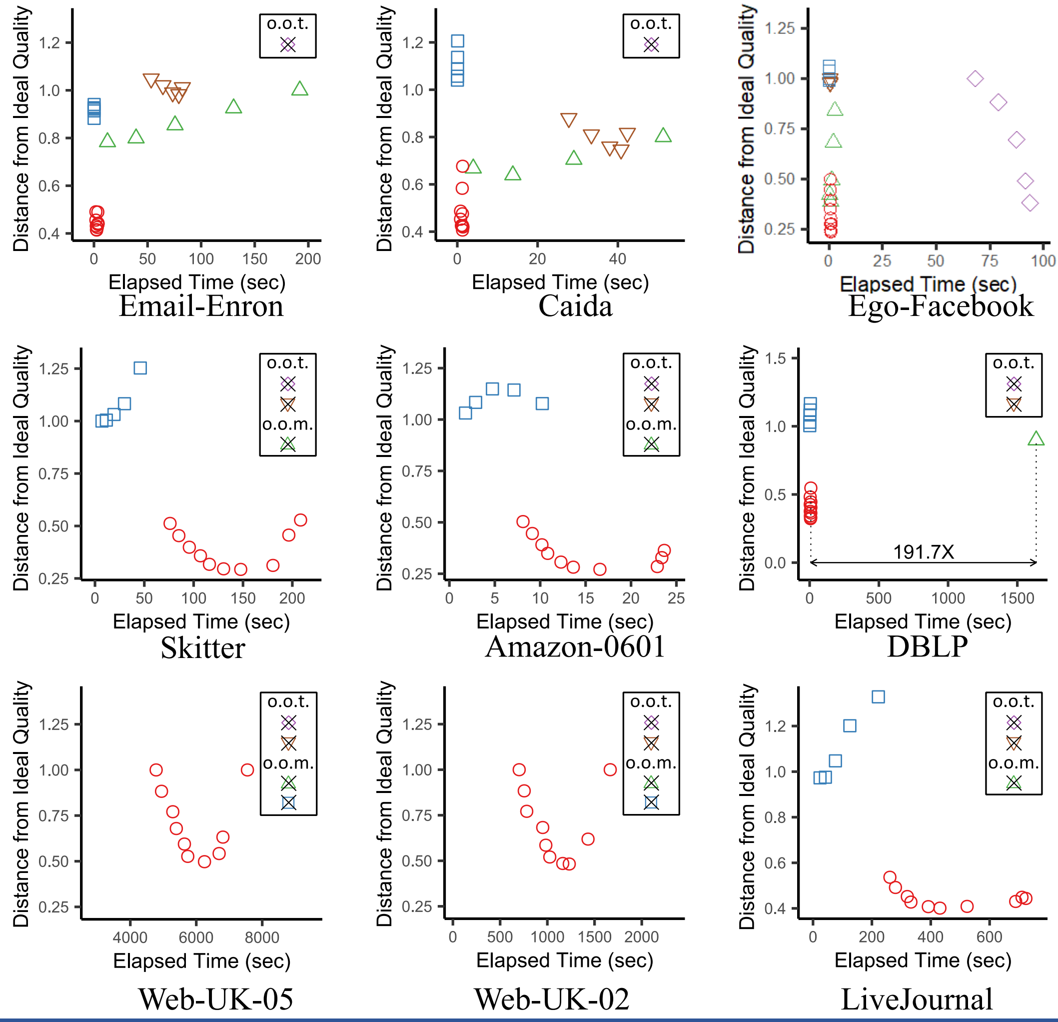
3.4 – 11.2X more accurate

3.4 – 10.4X more concise

SSumM is Fast

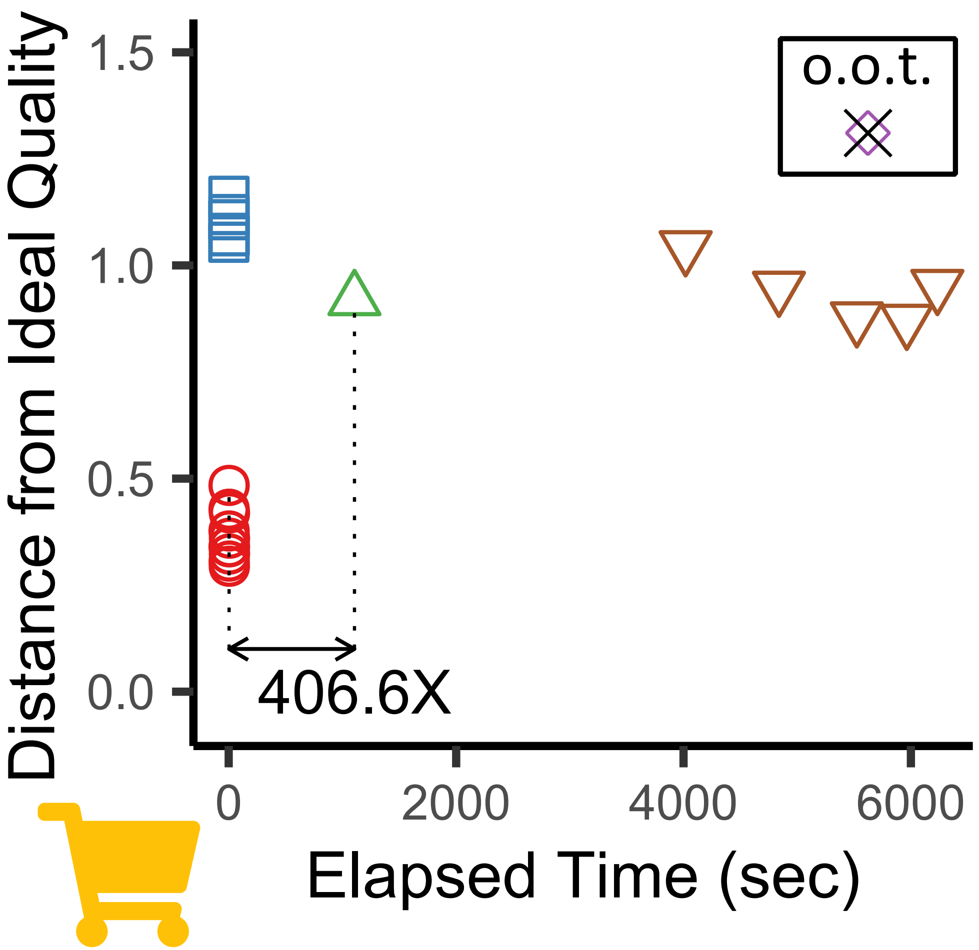
SSumM   SAA-Gs   SAA-Gs (linear sample)   S2L   k-Gs

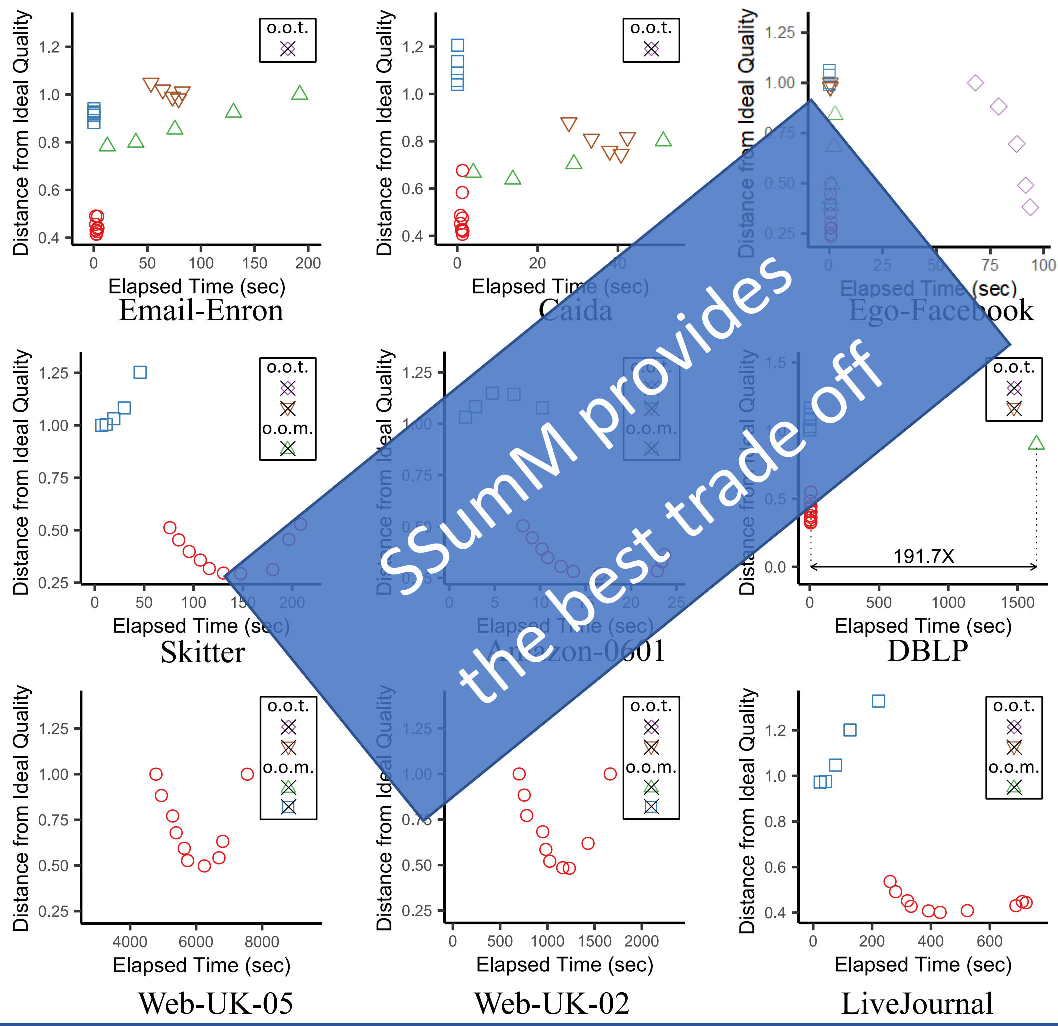




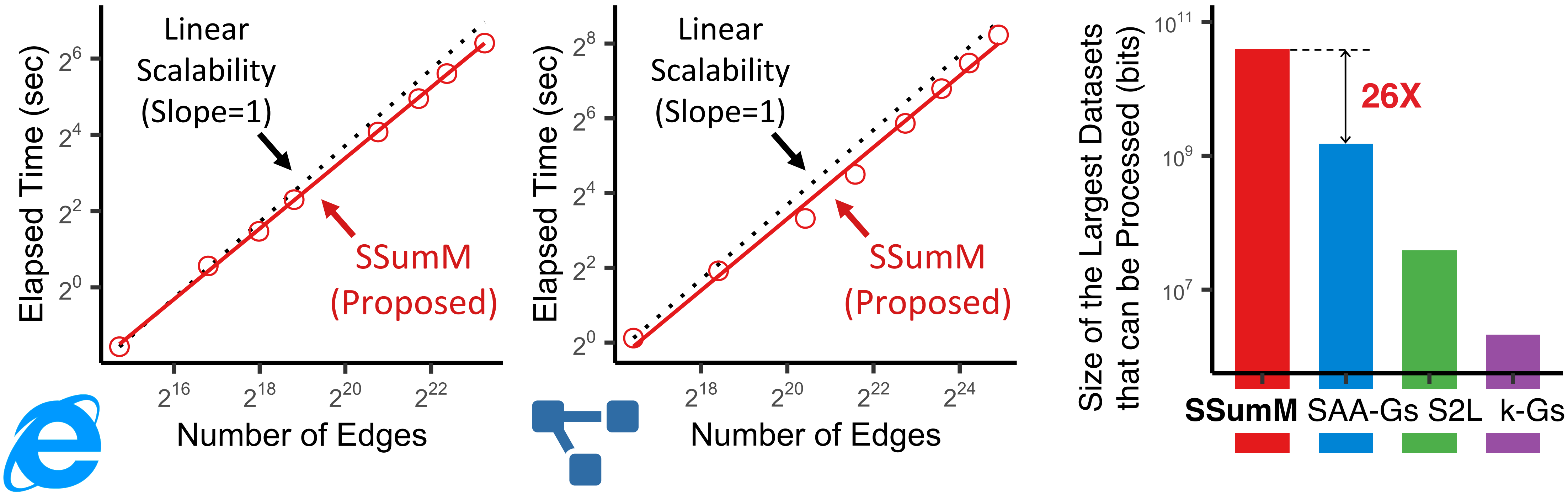
SSumM is Fast

SSumM SAA-Gs SAA-Gs (linear sample) S2L k-Gs

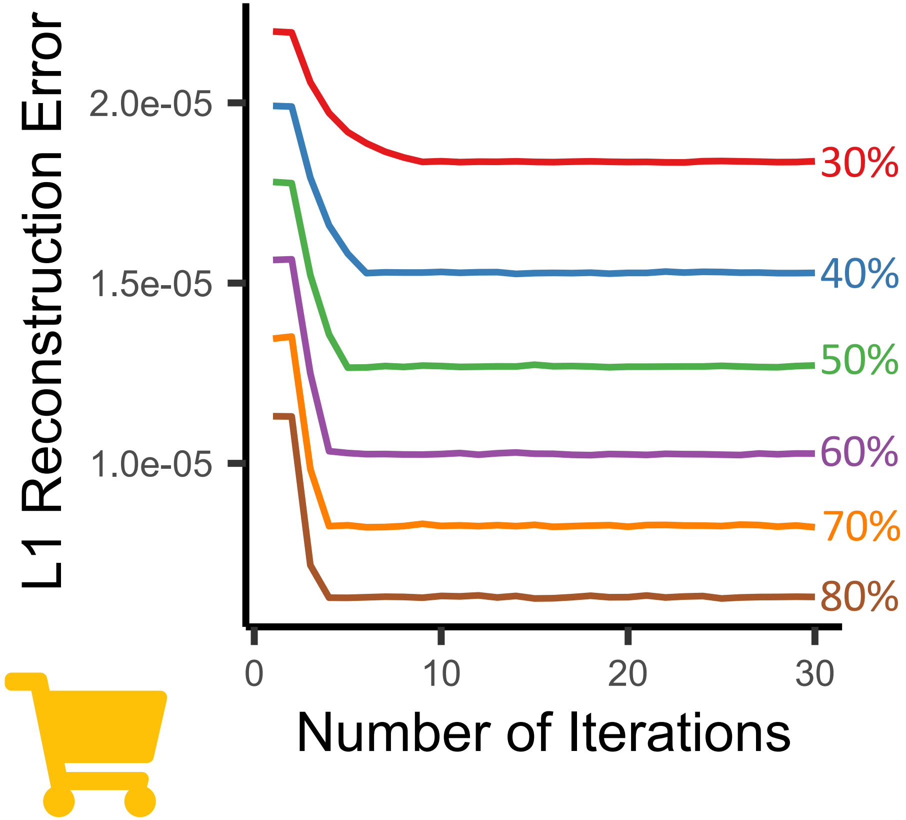


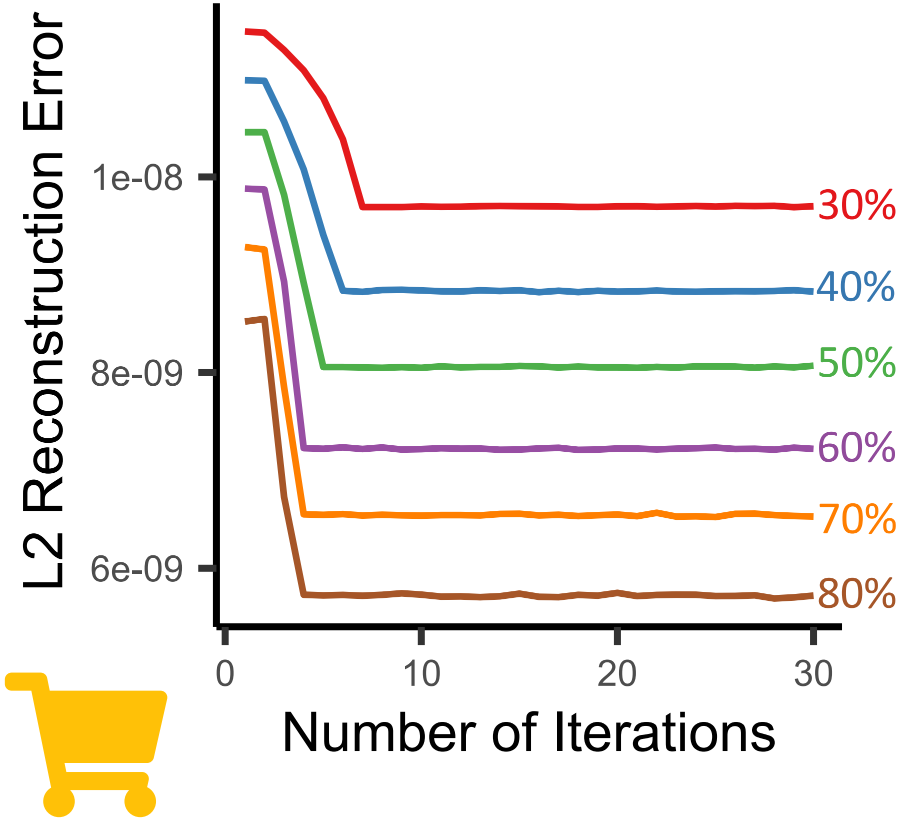


SSumM is Scalable



SSumM Converges Fast





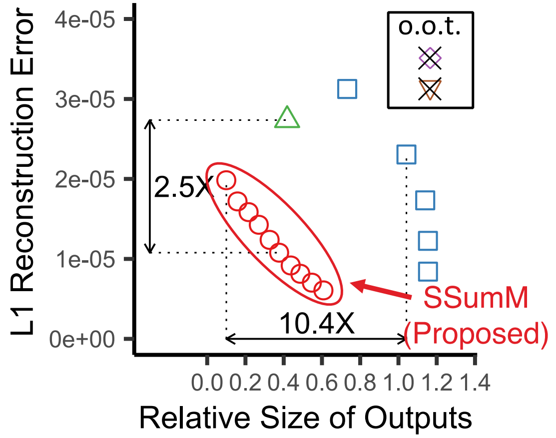
Road Map

- Introduction
- Problem
- Proposed Algorithm: SSumM
- Experimental Results
- **Conclusions <<**

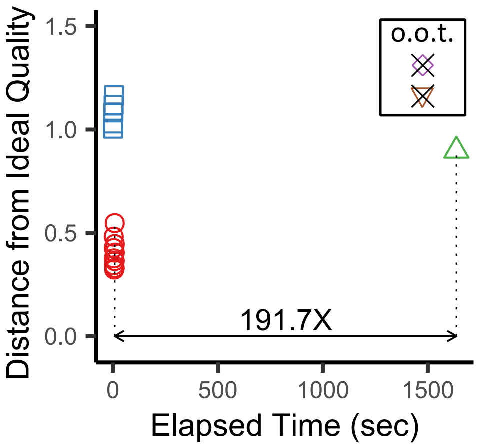
Conclusions

- Practical Problem Formulation
- Scalable and Effective Algorithm Design
- Extensive Experiments on 10 real world graphs

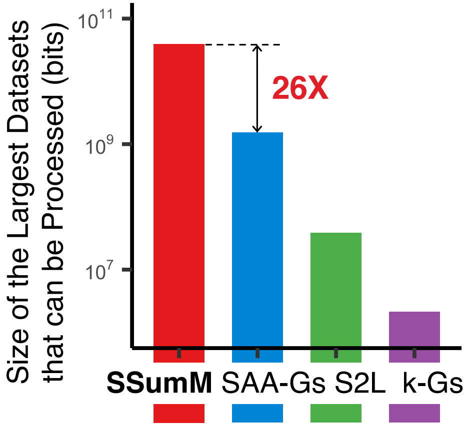
Concise & Accurate



Fast



Scalable

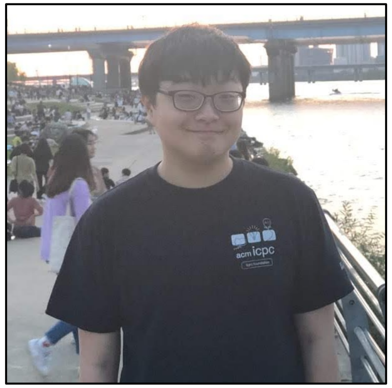


Code available at https://github.com/KyuhanLee/SSumM

SSumM : Sparse Summarization of Massive Graphs


Kyuhan Lee*


Hyeonsoo Jo*


Jihoon Ko


Sungsu Lim


Kijung Shin