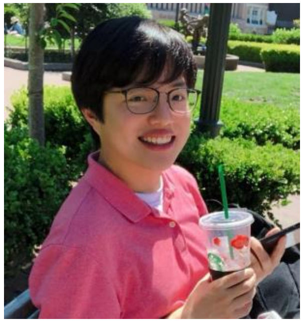
Personalized Graph Summarization: Formulation, Scalable Algorithms, and Applications

Shinhwan Kang   Kyuhan Lee   Kijung Shin

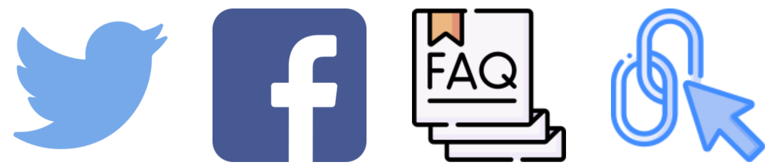
KAIST AI

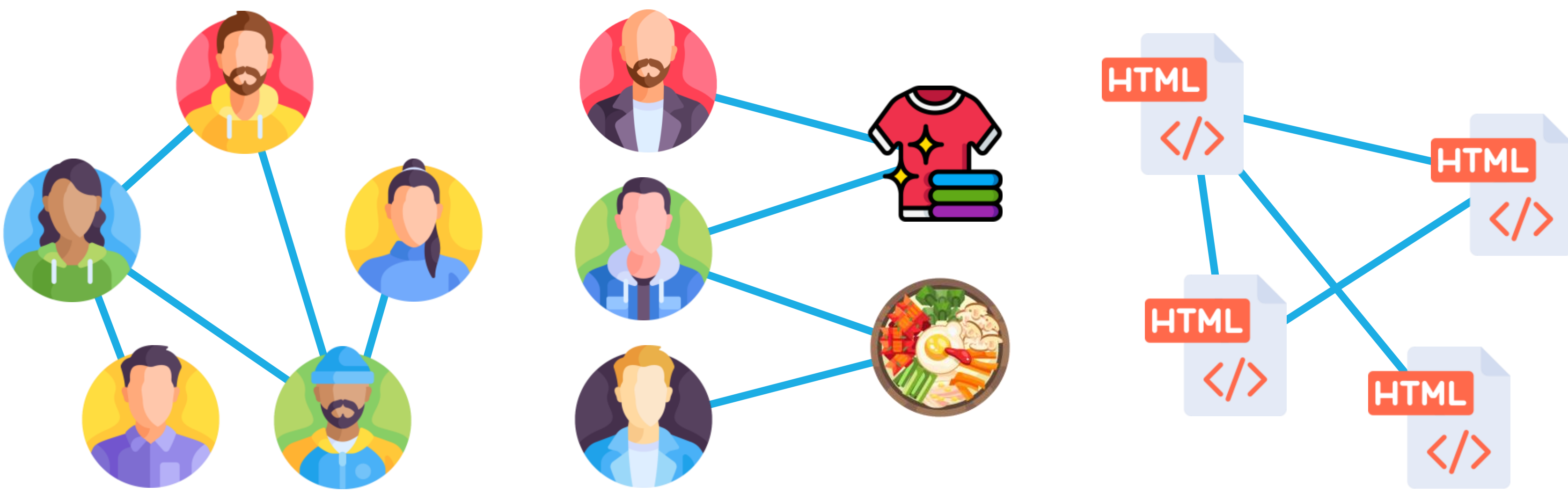




Graphs are everywhere!

- ***Graphs represent relationships*** such as
  - Friends in social networks
  - Purchase history
  - Hyperlinks between web pages





Social network **Purchase history** **Web graph**

Icon made by Freepik from www.flaticon.com

Graphs become large!

- Graphs ***grow rapidly*** at an unprecedented pace

25 million (2013)

157.4 million (2022)

Users

0.6 billion (2013)

1.9 billion (2022)

Web pages

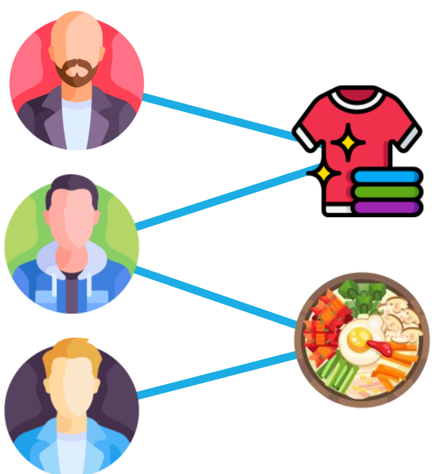
Graphs become large!

- Graphs ***grow rapidly*** at an unprecedented pace

25 million (2013)

157.4 million (2022)

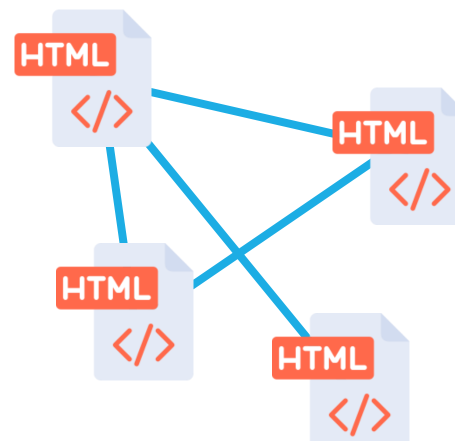
Users



0.6 billion (2013)

1.9 billion (2022)

Web pages



Graphs become large!

- Graphs ***grow rapidly*** at an unprecedented pace

25 million (2013)

157.4 million (2022)

Users

0.6 billion (2013)

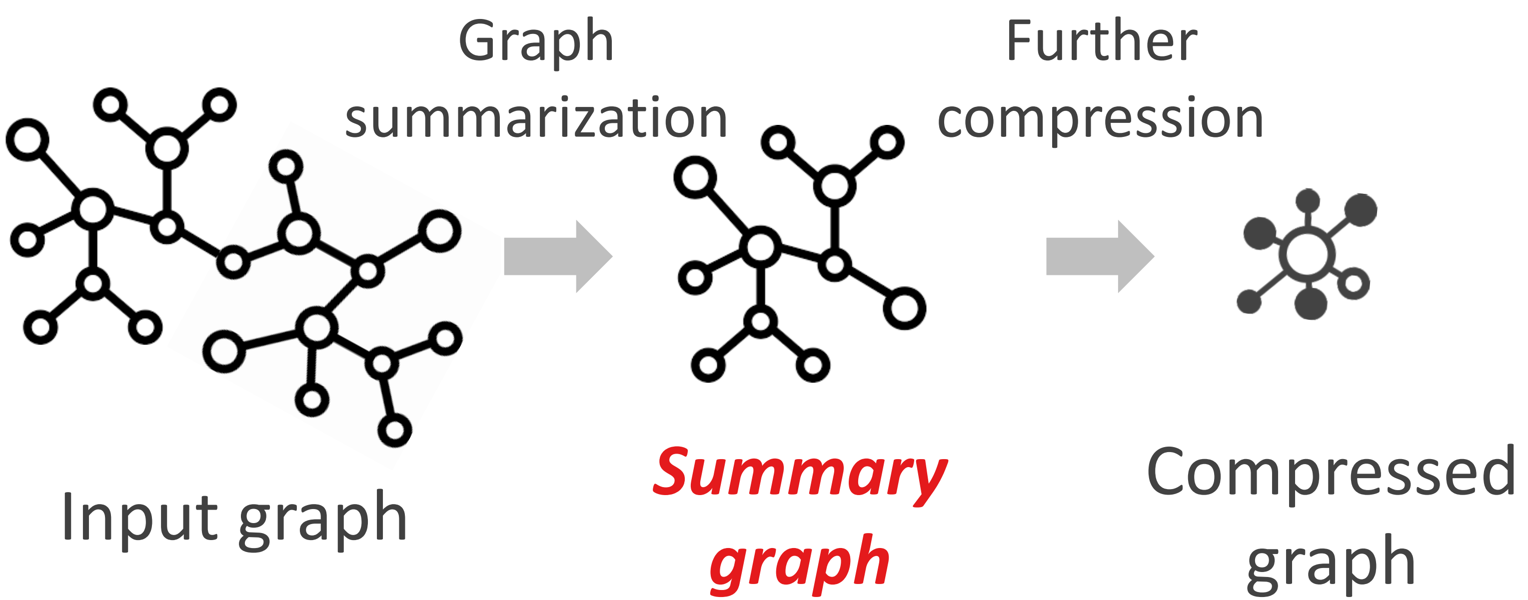
1.9 billion (2022)

Web pages

How do we efficiently utilize such large graphs?

Graph summarization

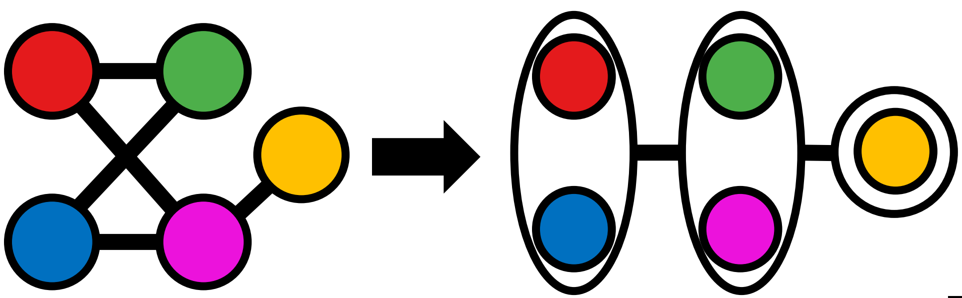
- A ***lossy*** graph compression technique [1, 2, 3, 4]
- A ***summary graph*** is ***in the form of a graph***
  - Directly query processing without restoration
  - Application of other graph compression techniques



Graph summarization

- **Given**: input graph G
- **Find**: summary graph $\bar{G}$

What should be the objective & constraint?

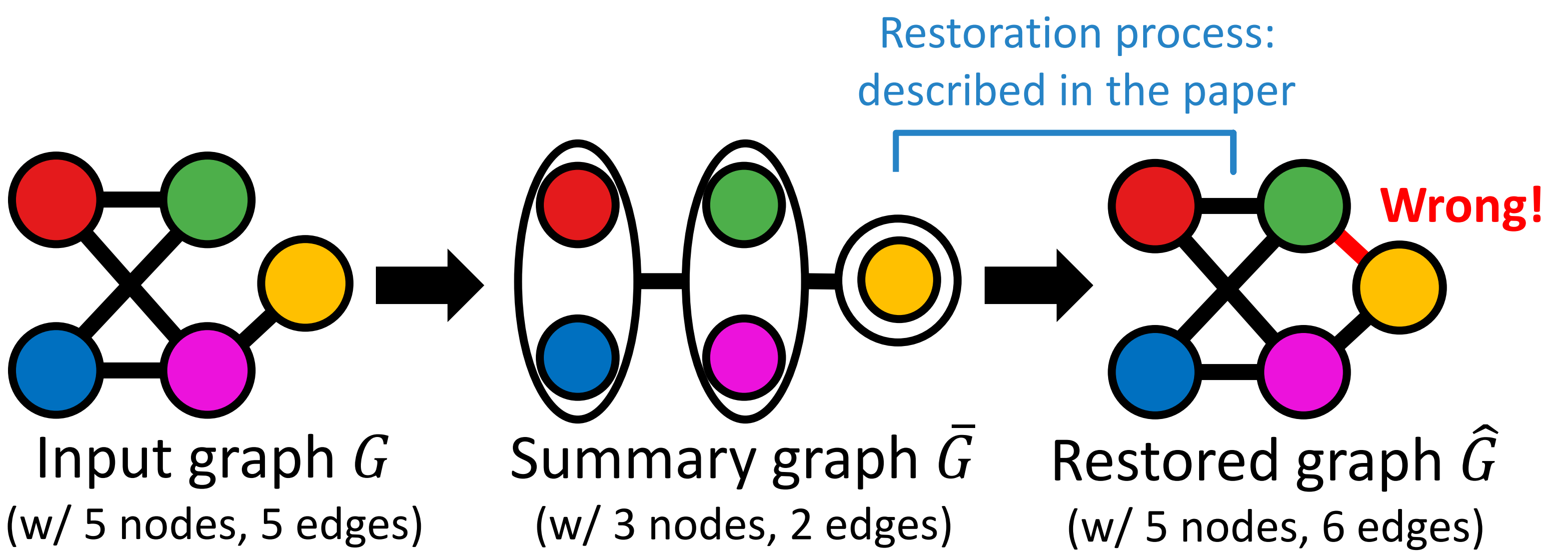


Input graph G
(w/ 5 nodes, 5 edges)

Summary graph $\bar{G}$
(w/ 3 nodes, 2 edges)

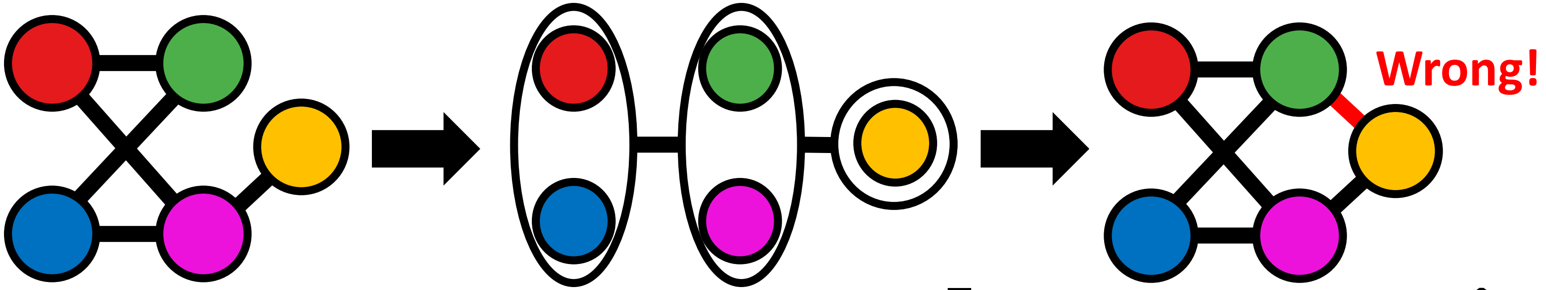
Graph summarization

- **Given**: input graph G
- **Find**: summary graph $\bar{G}$
- **To minimize**: ***the difference between G and $\hat{G}$***
  - (e.g.) Manhattan distance between adjacency matrices



Graph summarization

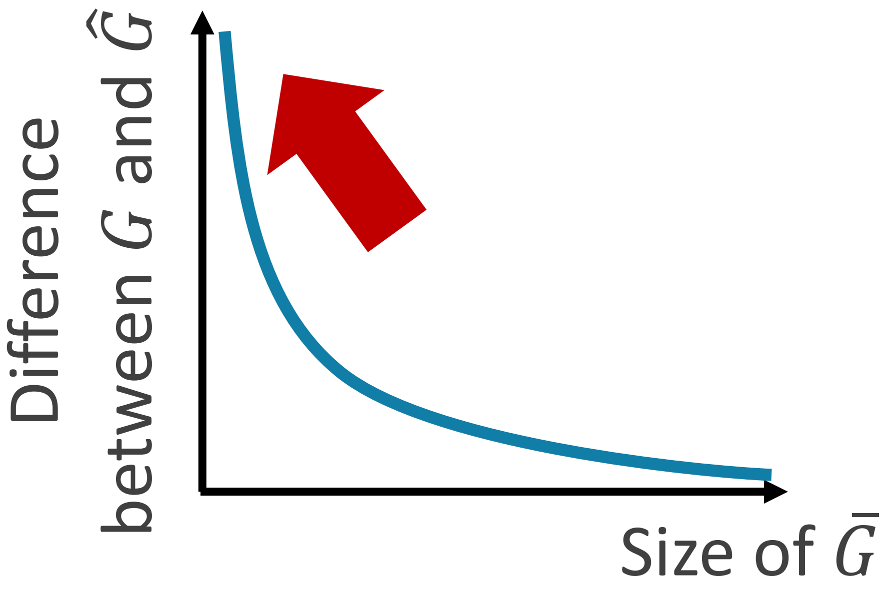
- **Given**: input graph G ***and a budget k***
- **Find**: summary graph $\bar{G}$
- **To minimize**: the difference between G and $\hat{G}$
  - (e.g.) Manhattan distance between adjacency matrices
- **Subject to**: size of summary graph $\bar{G} \leq \boldsymbol{k}$
  - (e.g.) # of nodes in $\bar{G}$, # of bits to encode $\bar{G}$



Input graph G (w/ 5 nodes, 5 edges) | Summary graph $\bar{G}$ (w/ 3 nodes, 2 edges) | Restored graph $\hat{G}$ (w/ 5 nodes, 6 edges)

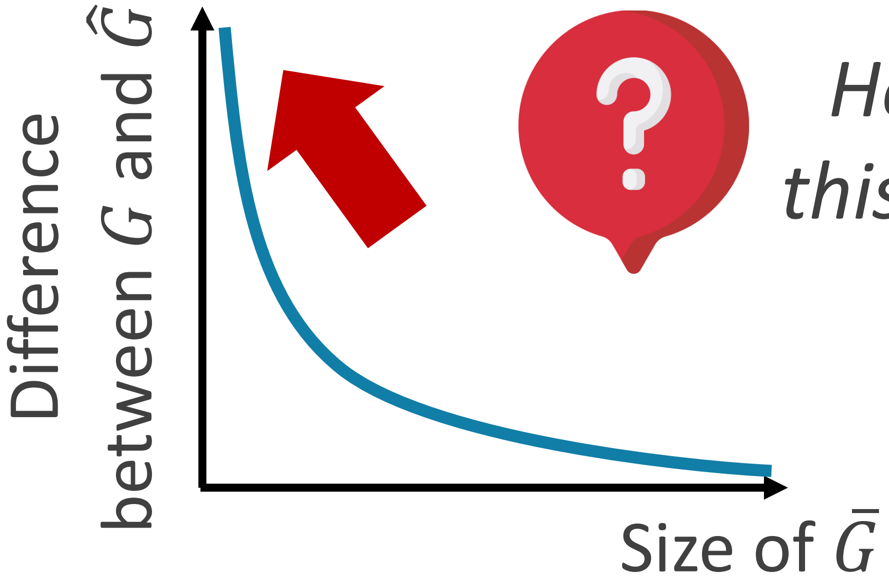
Limitation of graph summarization

- ***Information loss increases*** inevitably as a graph is more compressed



Limitation of graph summarization

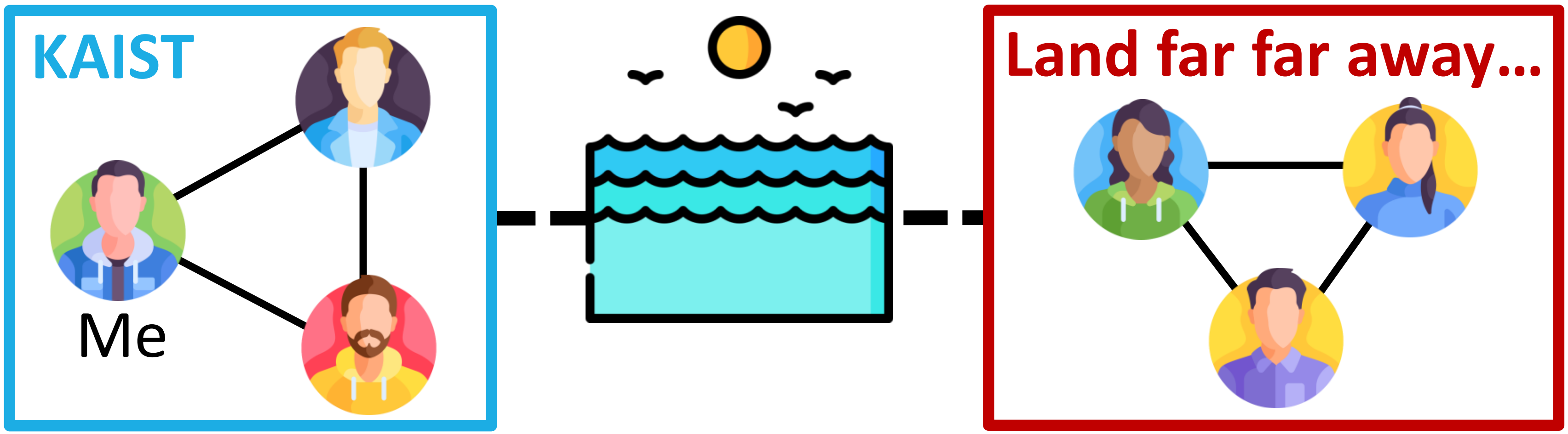
- ***Information loss increases*** inevitably as a graph is more compressed



How can we mitigate this inherent limitation?

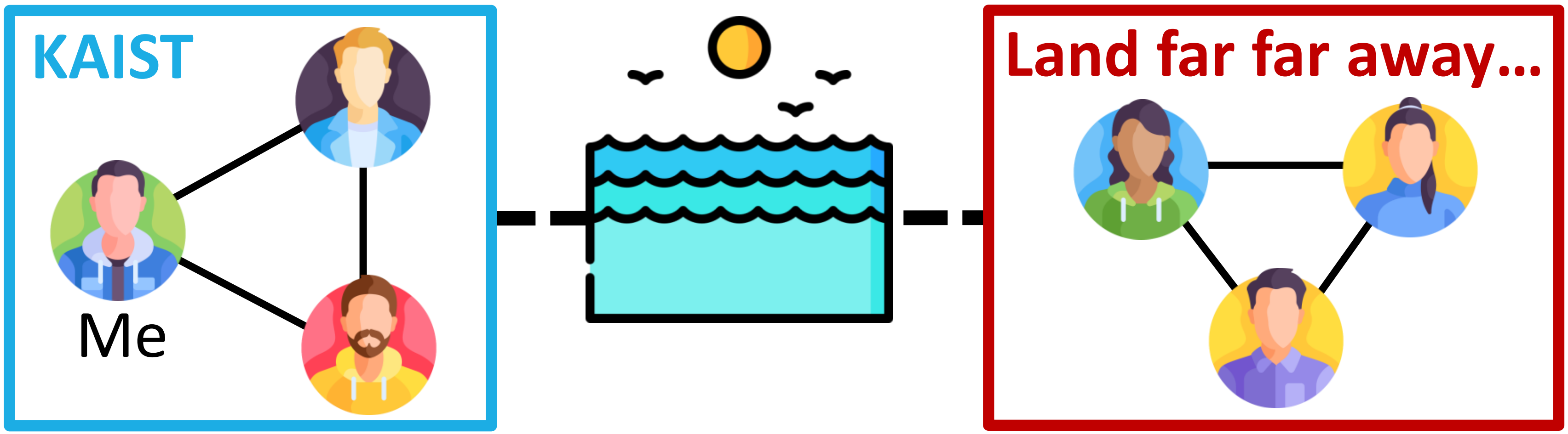
Motivation: example

- We often have ***different levels of interest*** in ***different parts*** of a graph



Motivation: example

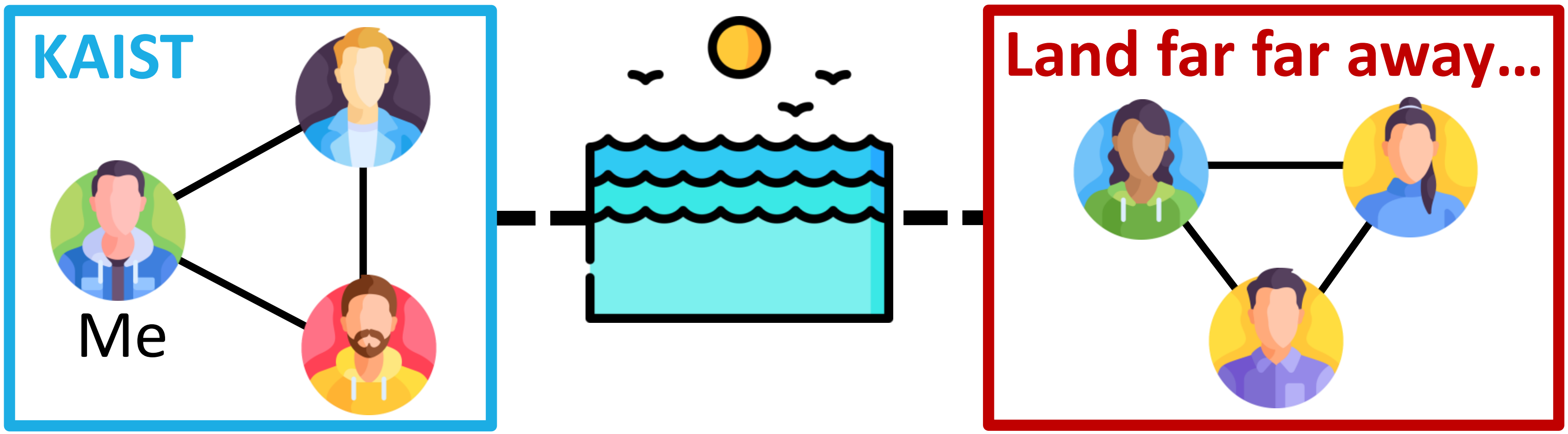
- We often have ***different levels of interest*** in ***different parts*** of a graph



For lossy compression, which connections do "I" prioritize to better preserve?

First law of geography

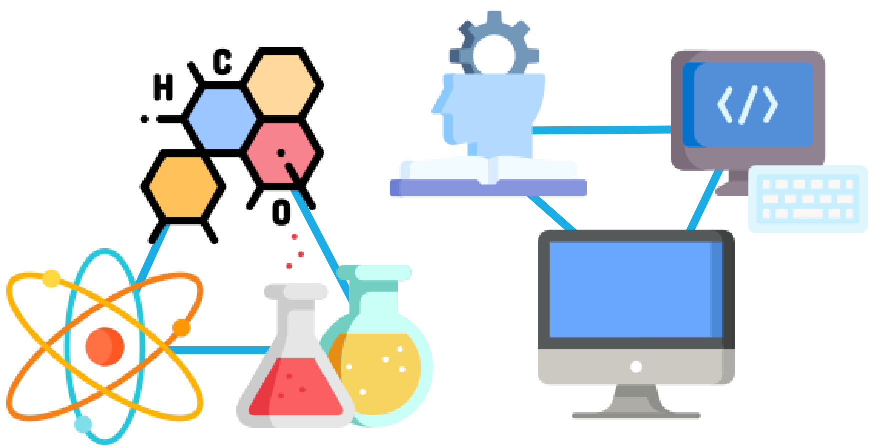
- We often have ***different levels of interest*** in ***different parts*** of a graph



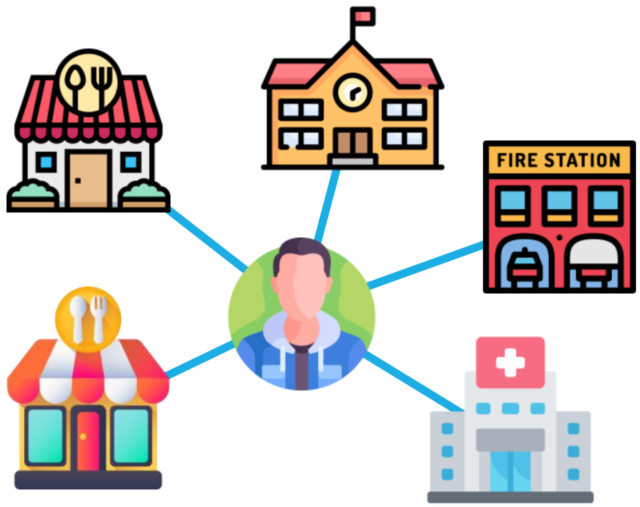
"Everything is related to everything else, but near things are more related than distant things" [5]
- Waldo Tobler (the 1st law of geography) -

First law of geography

- Other examples of the 1st law of geography



Citation network



Road network

Road map

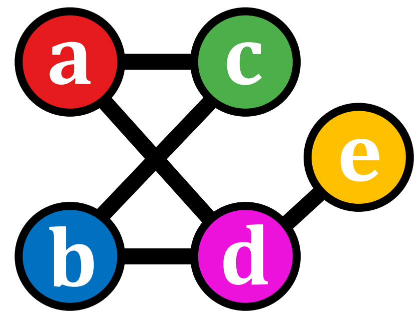
- ✓ Introduction
- ✓ **Problem formulation <<**
- ✓ Optimization: PeGaSus
- ✓ Application
- ✓ Experiments
- ✓ Conclusion

Personalized graph summarization

- **Given**: input graph: $G = (V, E)$

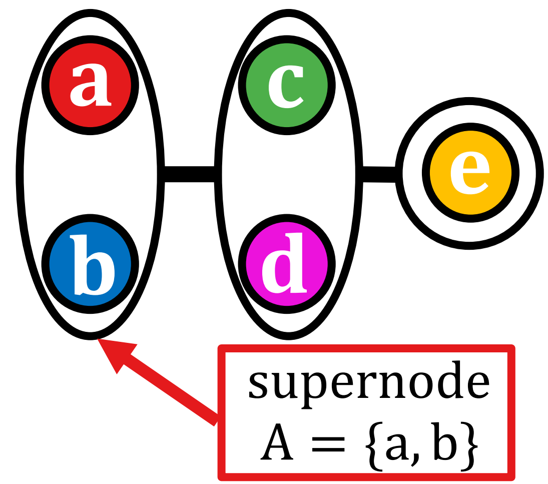
  set of target nodes: $T (\subseteq V)$

  and a budget: k



- **Find**: summary graph $\bar{G} = (S, P)$

  personalized to T



- **To minimize**: error ***personalized to T***

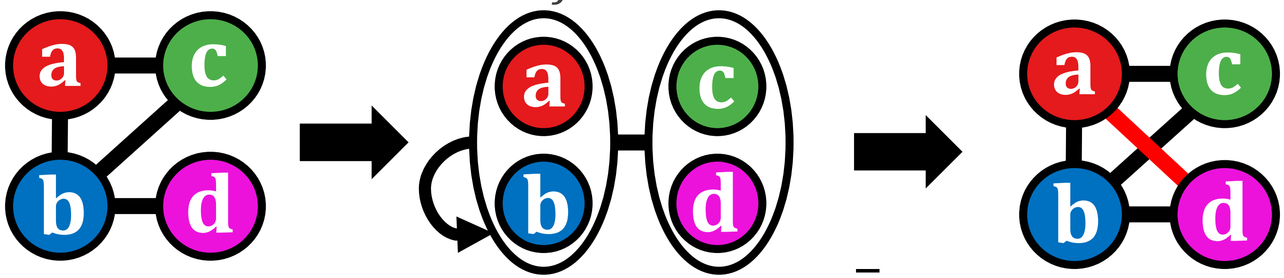
- **Subject to**: $\mathrm{Size}(\bar{G})$ = # of bits to encode $\bar{G} \leq k$

Personalized error

- Personalized error is the ***weighted sum of errors***

$$\sum_{i=1}^{|V|}\sum_{j=1}^{|V|} W_{ij}^{(T)} \left|A_{ij}^{(G)} - A_{ij}^{(\hat{G})}\right|,$$

  - where each weight $W_{ij}^{(T)}$ is personalized to target nodes T



Input graph G Summary graph $\bar{G}$ Restored graph $\hat{G}$

$A_{ij}^{(G)}$

	a	b	c	d
a	0	1	1	0
b	1	0	1	1
c	1	1	0	0
d	0	1	0	0

$A_{ij}^{(\hat{G})}$

	a	b	c	d
a	0	1	1	**1**
b	1	0	1	1
c	1	1	0	0
d	**1**	1	0	0

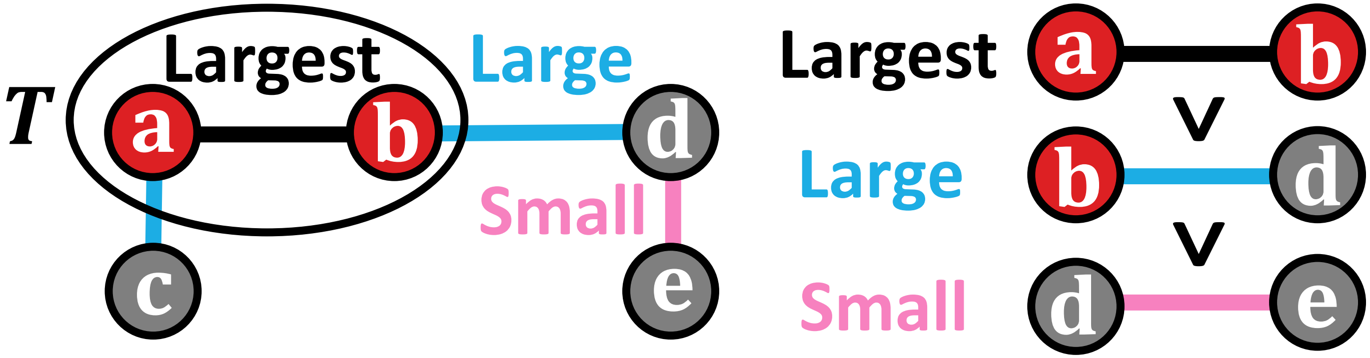
Personalized weight

- Personalized weight on a node pair *depends on* ***their distance from target nodes***

$$W_{ij}^{(T)} \propto \alpha^{-(D(i,T)+D(j,T))}$$

  - where $D(i,T) = \min_{t \in T}(\#of\ hops(i,t))$ and α is a constant

(e.g.) Personalized weights are



Road map

- ✓ Introduction
- ✓ Problem formulation
- ✓ **Optimization: PeGaSus <<**
- ✓ Application
- ✓ Experiments
- ✓ Conclusion

Optimization: PeGaSus

- ***Pe***rsonalized ***G***r***a***ph ***Su***mmarization with ***S***calability

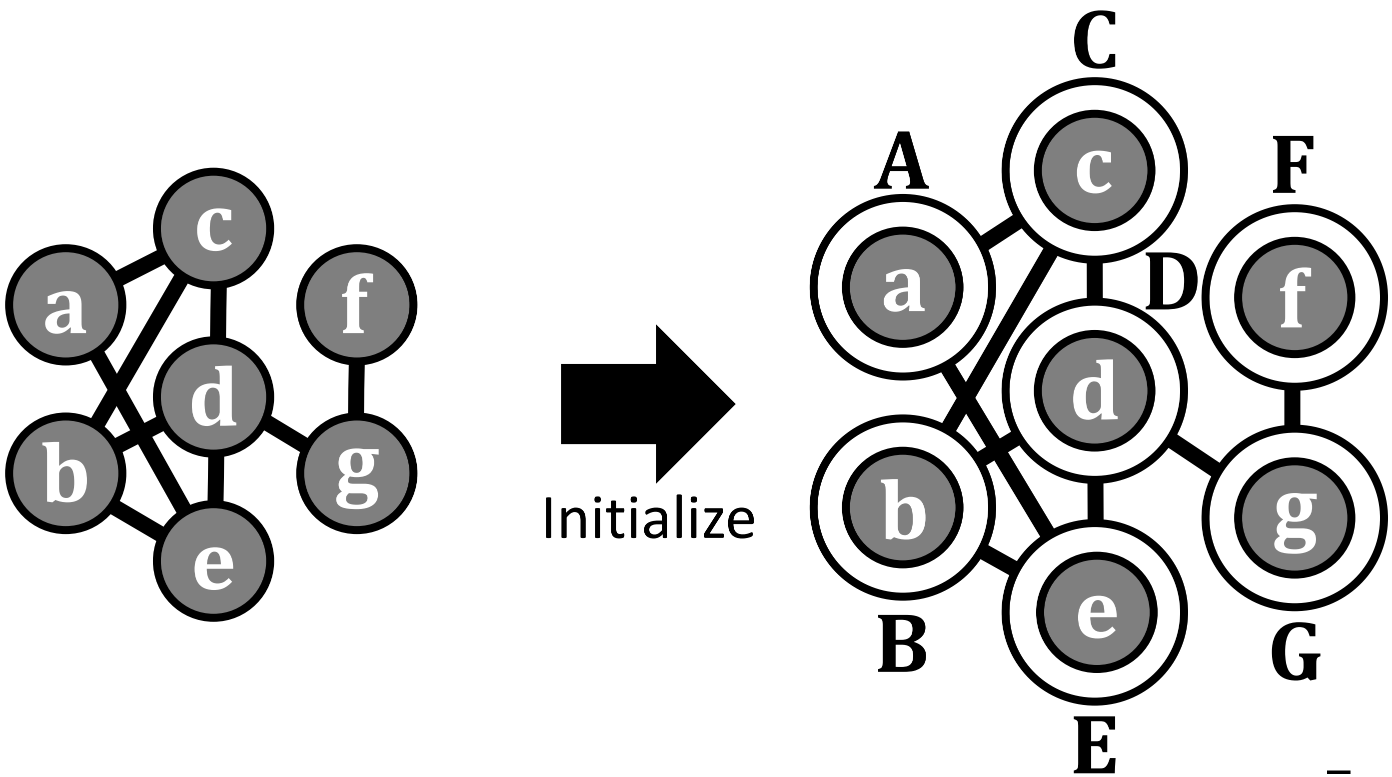
- [ ] **Effective in personalization**
- [ ] **Useful for applications**
- [ ] **Scalable to large graphs**

Overview: PeGaSus

- PeGaSus is largely based on SSumM [1]
- **Inputs**
  - input graph G
  - size budget k
  - set of target nodes T
  - max. number of iterations t_{max}
- **Output**
  - ***personalized*** summary graph $\bar{G}$
- **Procedure**
  - ➢ initializing step
  - ➢ repeat t_{max} times or until $\mathrm{Size}(\bar{G}) > k$
    - ➢ dividing step & merging step
  - ➢ If $\mathrm{Size}(\bar{G}) > k$, then sparsifying step

Initializing step

➢ ***Initialize*** a summary graph $\bar{G}$, and a threshold $\theta_{(0)}$

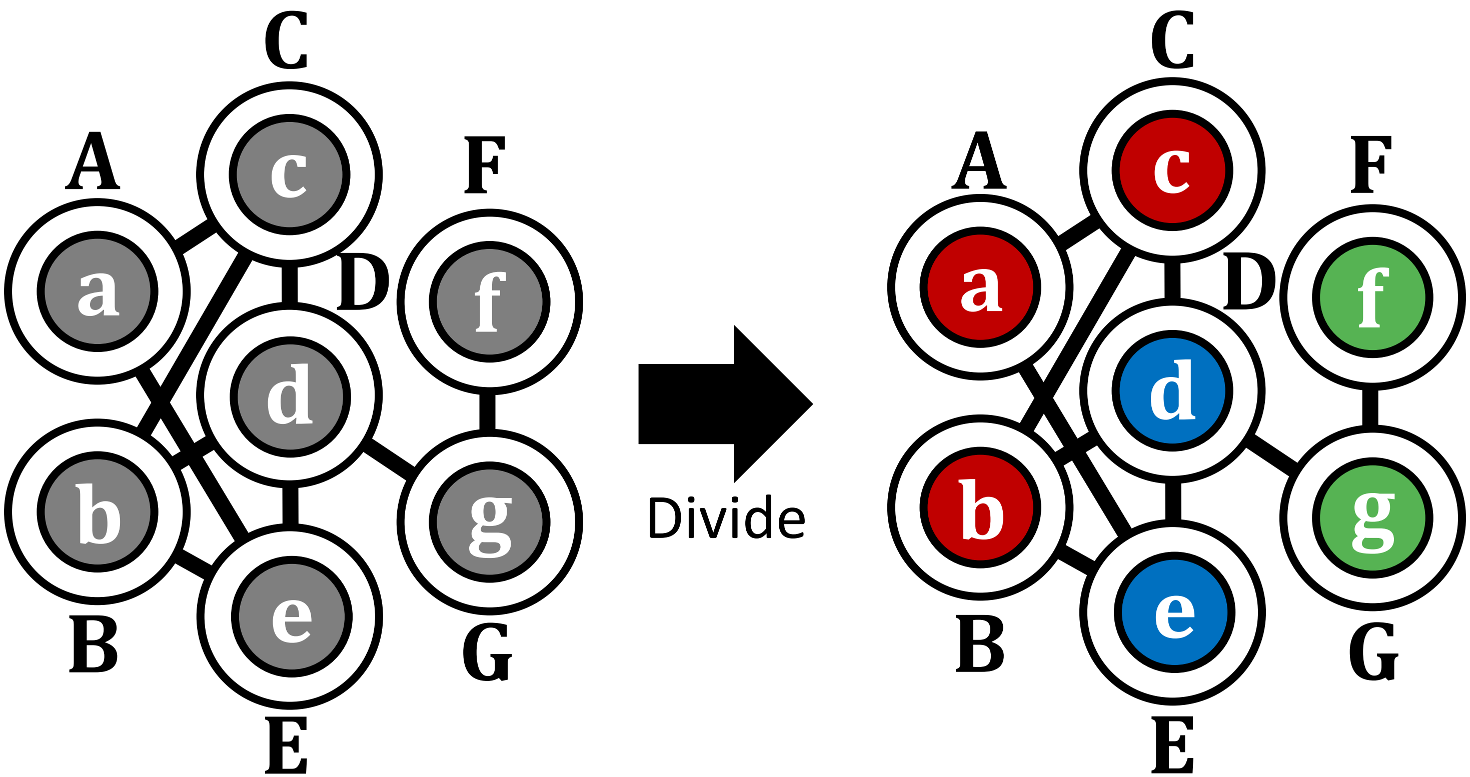


Input graph G

Summary graph $\bar{G}$

Dividing step

- ***Divide*** supernodes into groups U by MinHashing

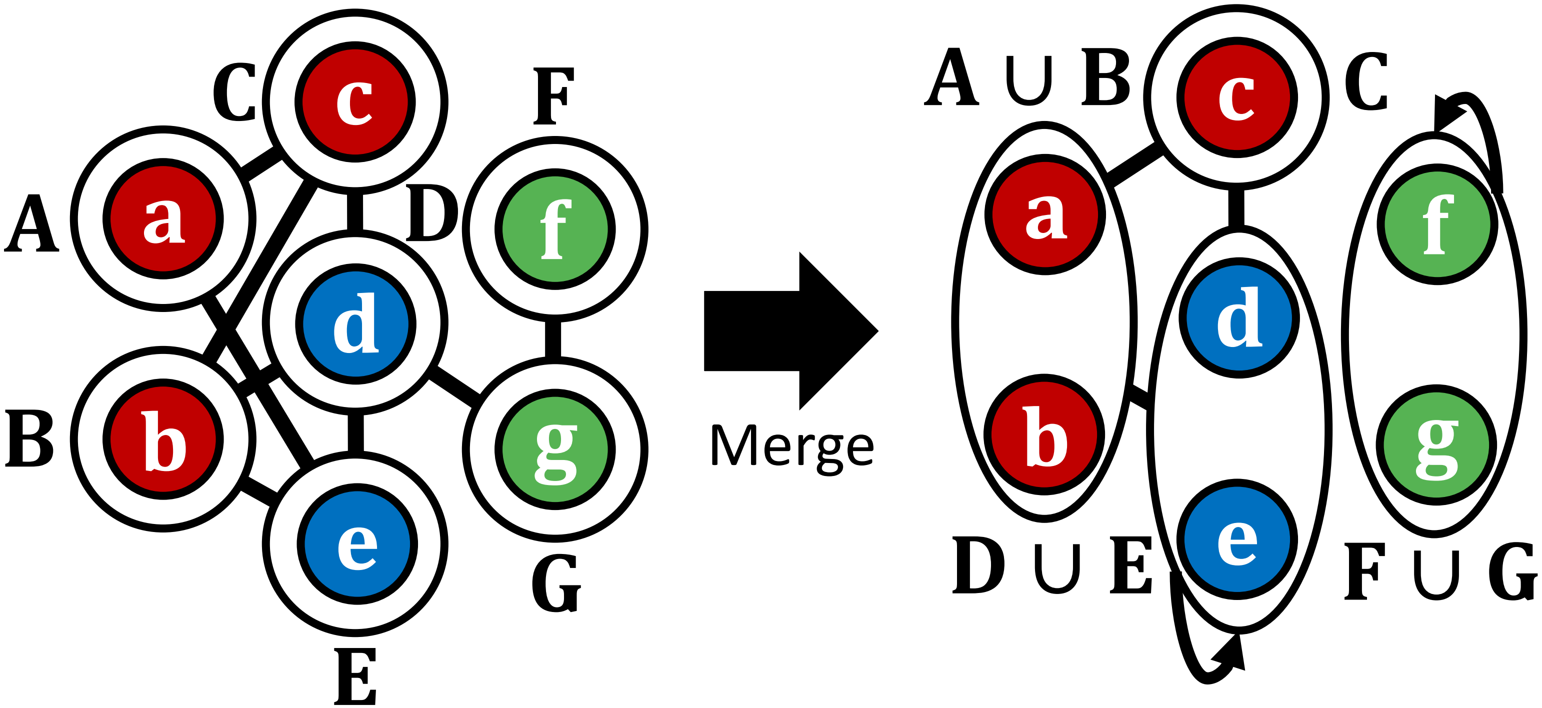


U: $\{\{A, B, C\}, \{D, E\}, \{F, G\}\}$

Merging step

- For each group of U, if $\boldsymbol{Saving}^{(T)} > \theta_{(0)}$, **merge** supernodes

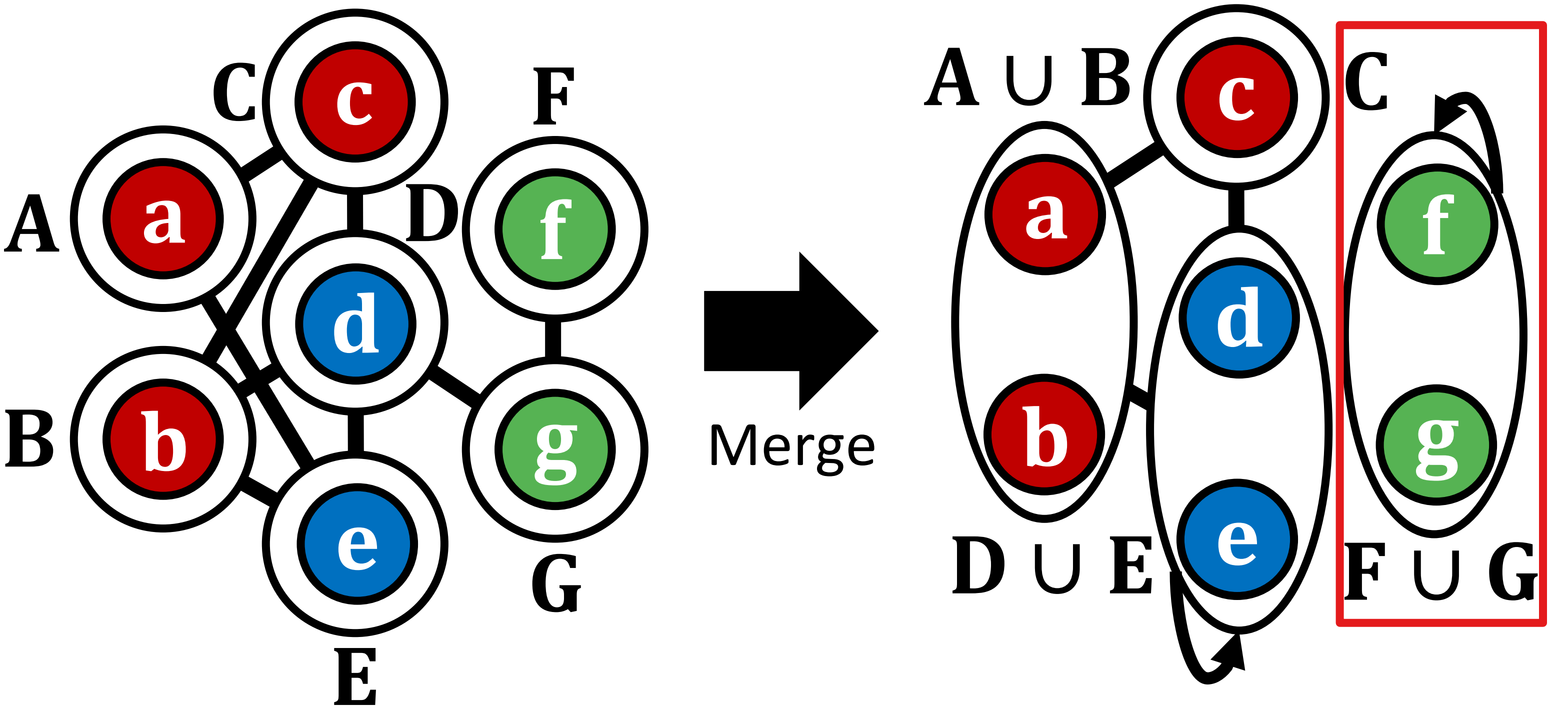
* $\boldsymbol{Saving}^{(T)} \approx$ saving (in bits) in personalized error + size



Merging step

➢For each group of U, if ***Saving***$^{(T)} > \theta_{(0)}$, ***merge*** supernodes

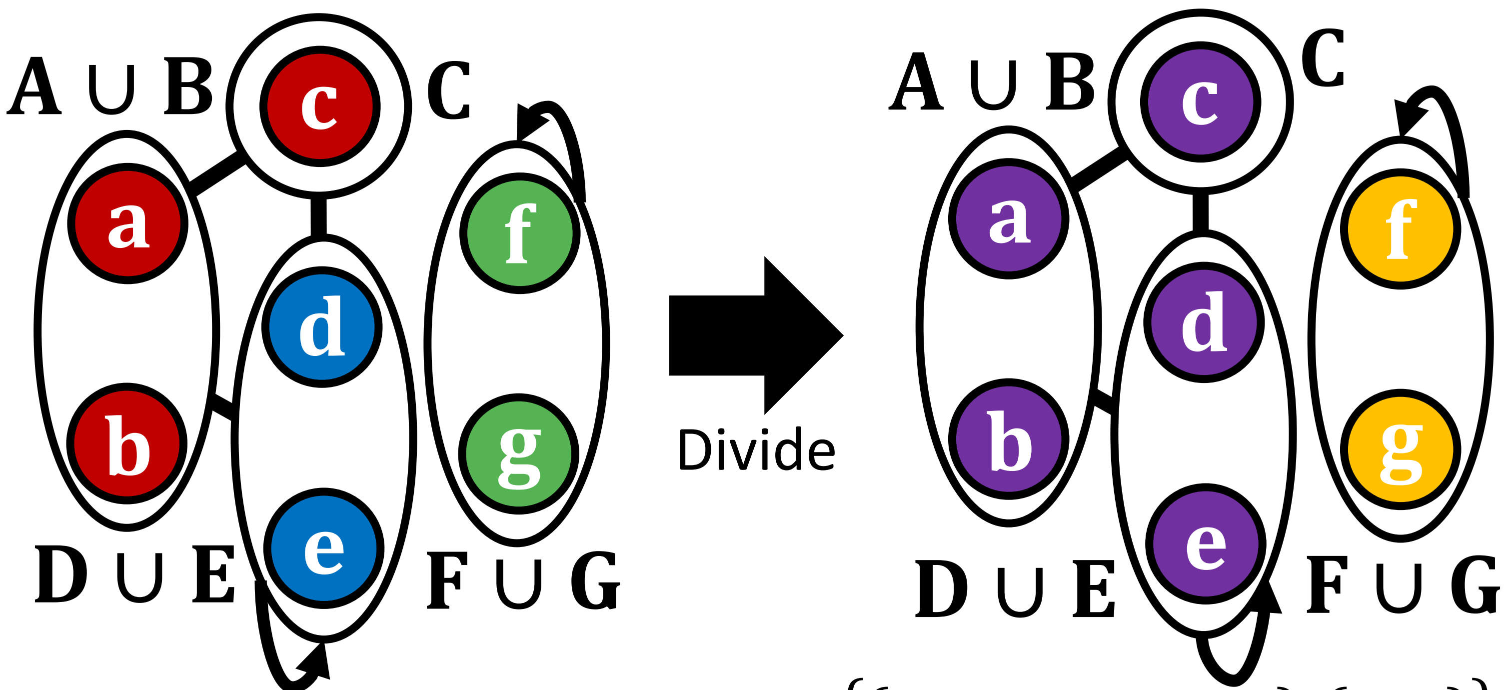
* ***Saving***$^{(T)}$ ≈ saving (in bits) in personalized error + size



Dividing step

➢ ***Divide*** supernodes into groups U by MinHashing

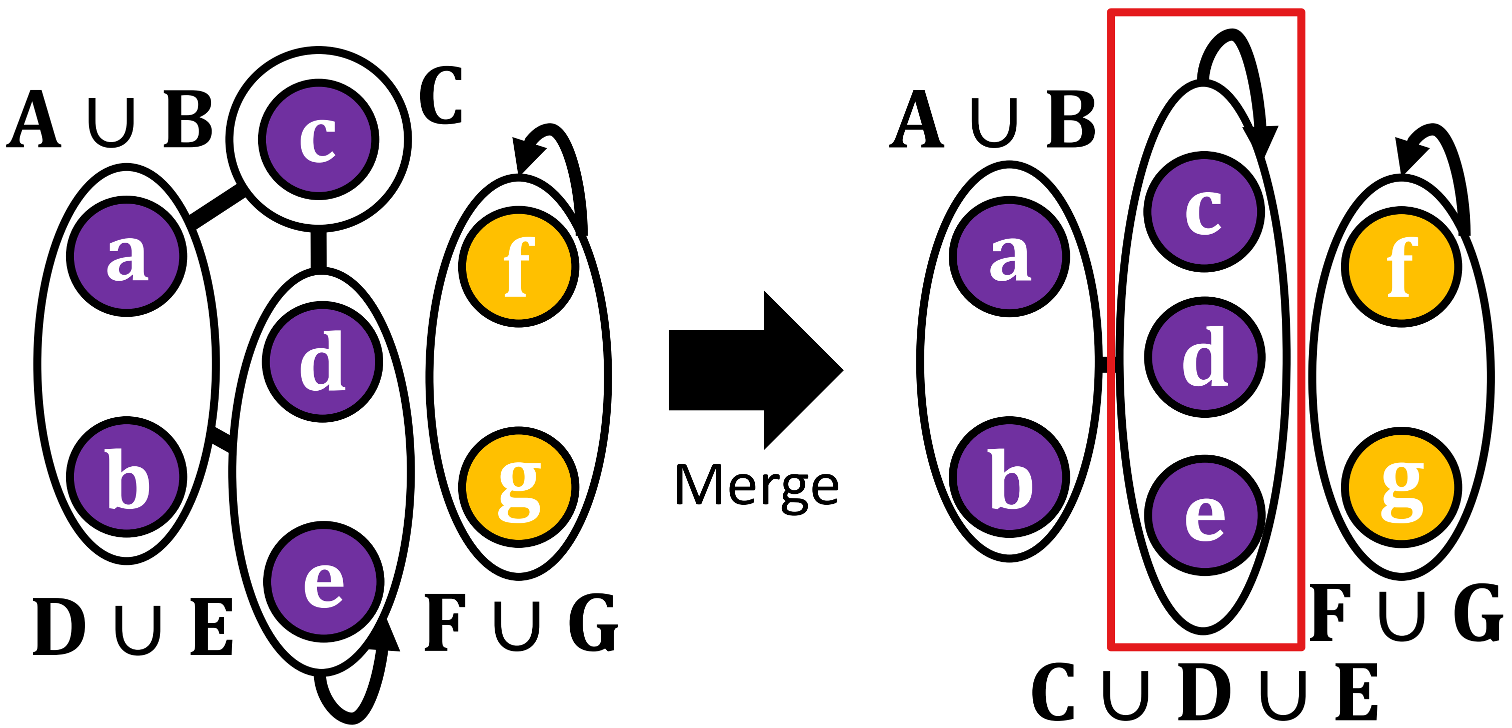
* MinHashing gives different partitions in each iteration



U: {{A ∪ B, C, D ∪ E}, {F, G}}

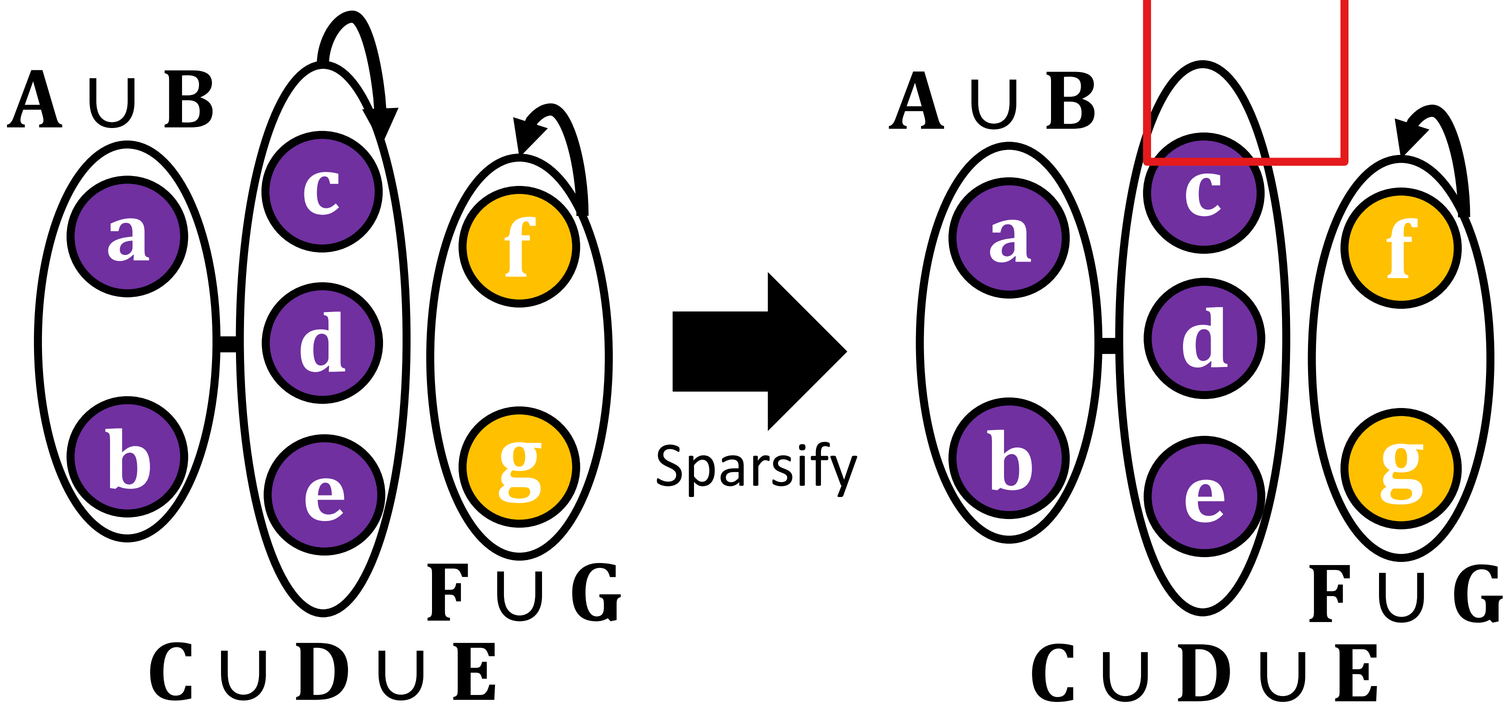
Merging step

- For each group of U, if $Saving^{(T)} > \theta_{(1)}$, ***merge*** supernodes



Sparsifying step

- After t_{max} iterations, if $\text{Size}(\bar{G}) > k$, ***drop*** superedges to maximize $Saving^{(T)}$



Adaptive threshold: motivation

- In the merging step,
  - If $\boxed{Saving^{(T)} > \theta_{(\cdot)},}$ merge supernodes
  - ...

- ***Controlling θ*** is important for output quality [6]
  - Small θ: supernodes are merged myopically even when better pairs can be found later
  - Large θ: supernodes remain without being merged
  - A ***fixed rule*** was used to reduce θ over iterations [1,6]
- PeGaSus ***controls θ adaptively*** based on past savings

Adaptive threshold: details

- PeGaSus ***controls θ adaptively*** based on past savings
- θ is set to ***top 10%*** of $Saving^{(T)}$ at "unsuccessful" searches in the previous iteration
- θ always decreases over iterations
  - $Saving^{(T)}$ at unsuccessful searches is at most the current θ

$Saving^{(T)}$ at unsuccessful searches

Small ⟷ **Large**

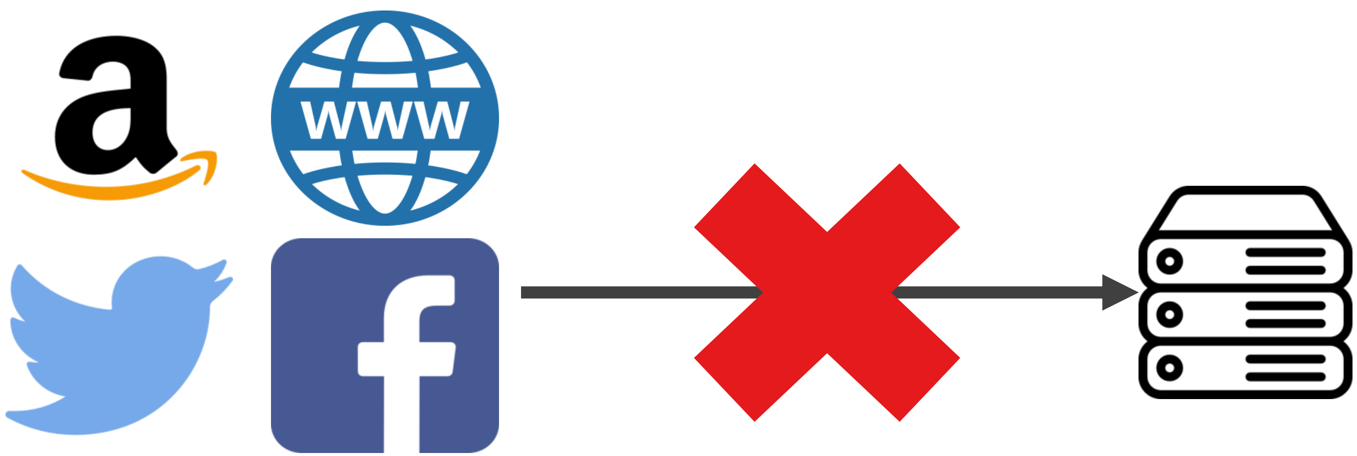
$\theta_{(t+1)} =$ Top 10% entry

Road map

- ✓ Introduction
- ✓ Problem formulation
- ✓ Optimization: PeGaSus
- ✓ **Application <<**
- ✓ Experiments
- ✓ Conclusion

Motivation: storing big graphs

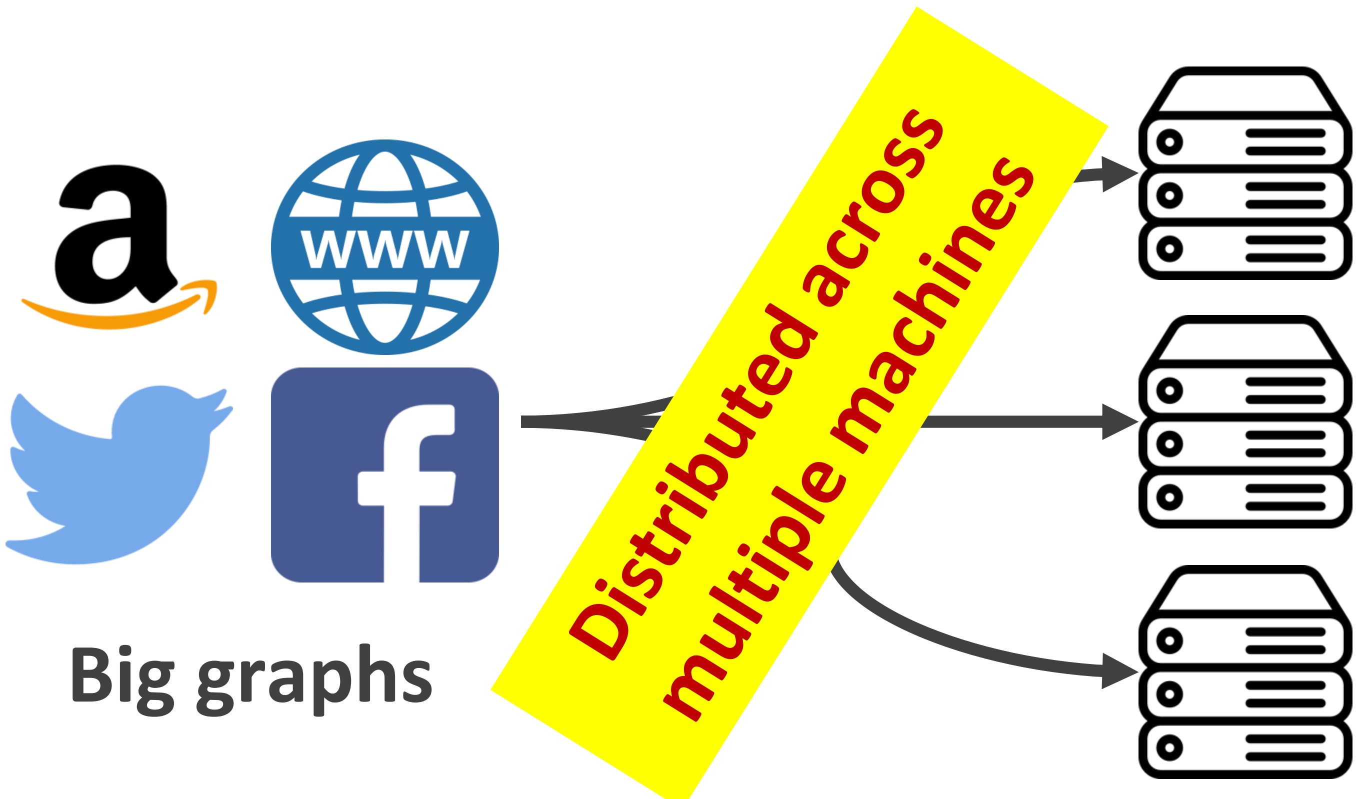
- Real-world graphs are often ***too large to be stored in a single machine***



Big graphs

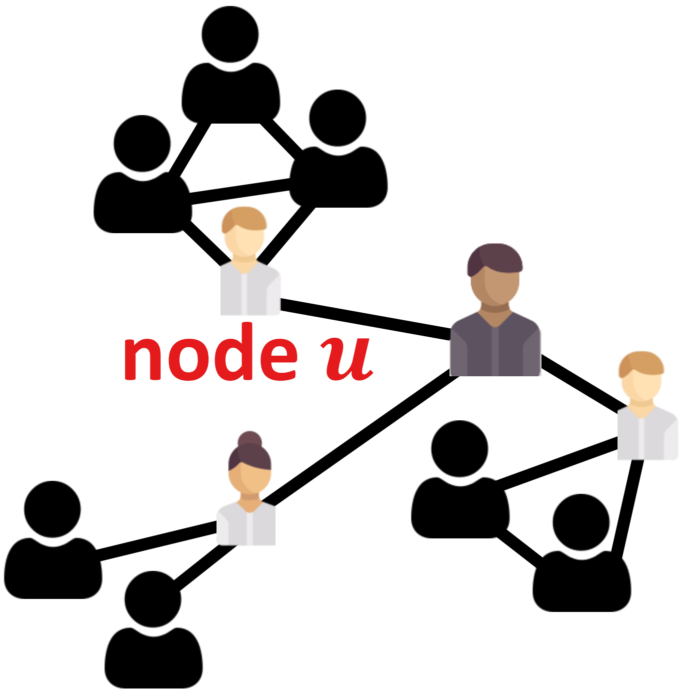
Motivation: storing big graphs

- Thus, real-world graphs are typically ***distributed across multiple machines***



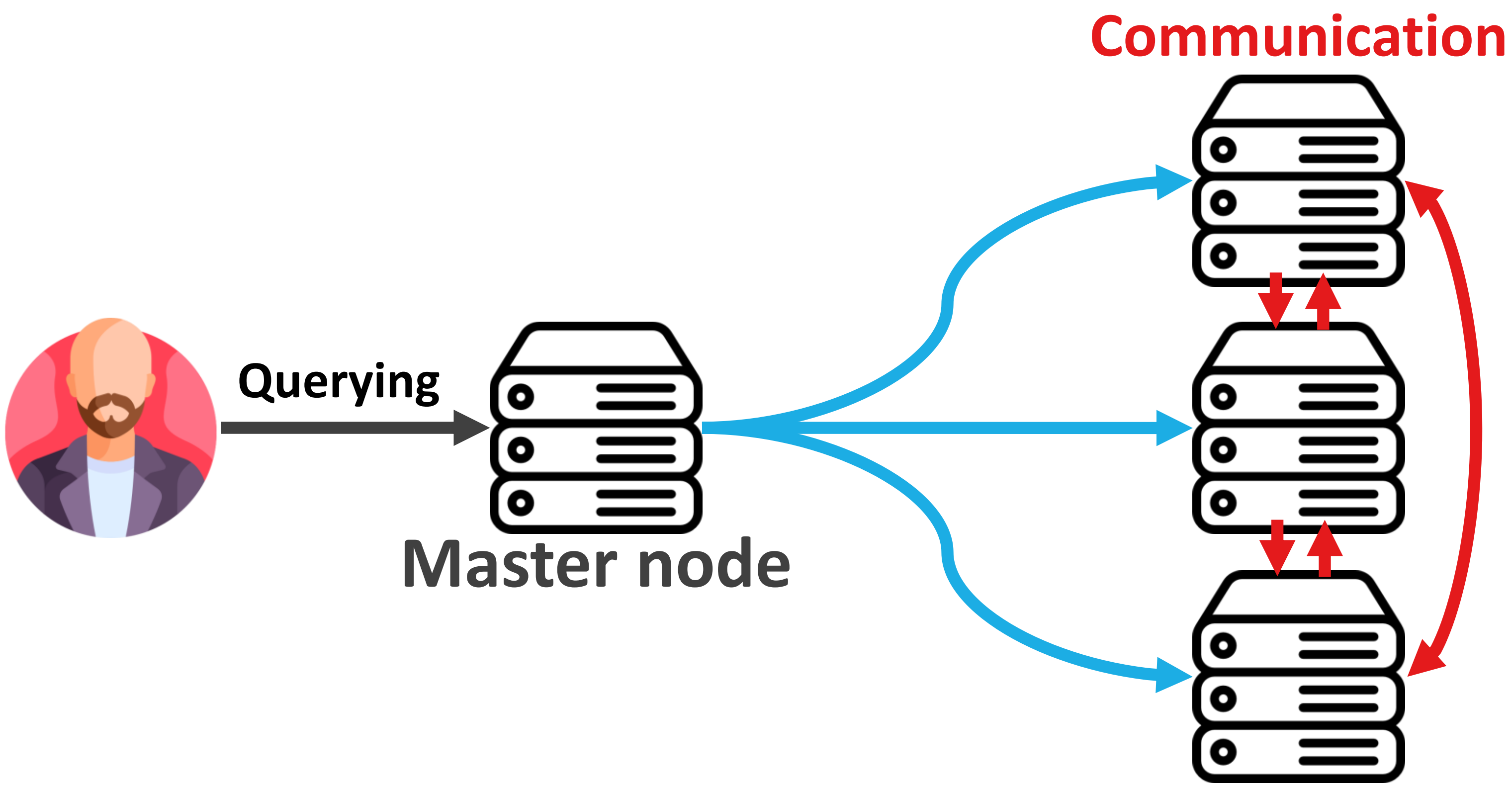
Motivation: query answering

- How are queries answered on distributed graphs?
  - E.g., Which node is the *most similar* to a ***node u***?
  - E.g., Who are the *neighbors* of a ***node u***?



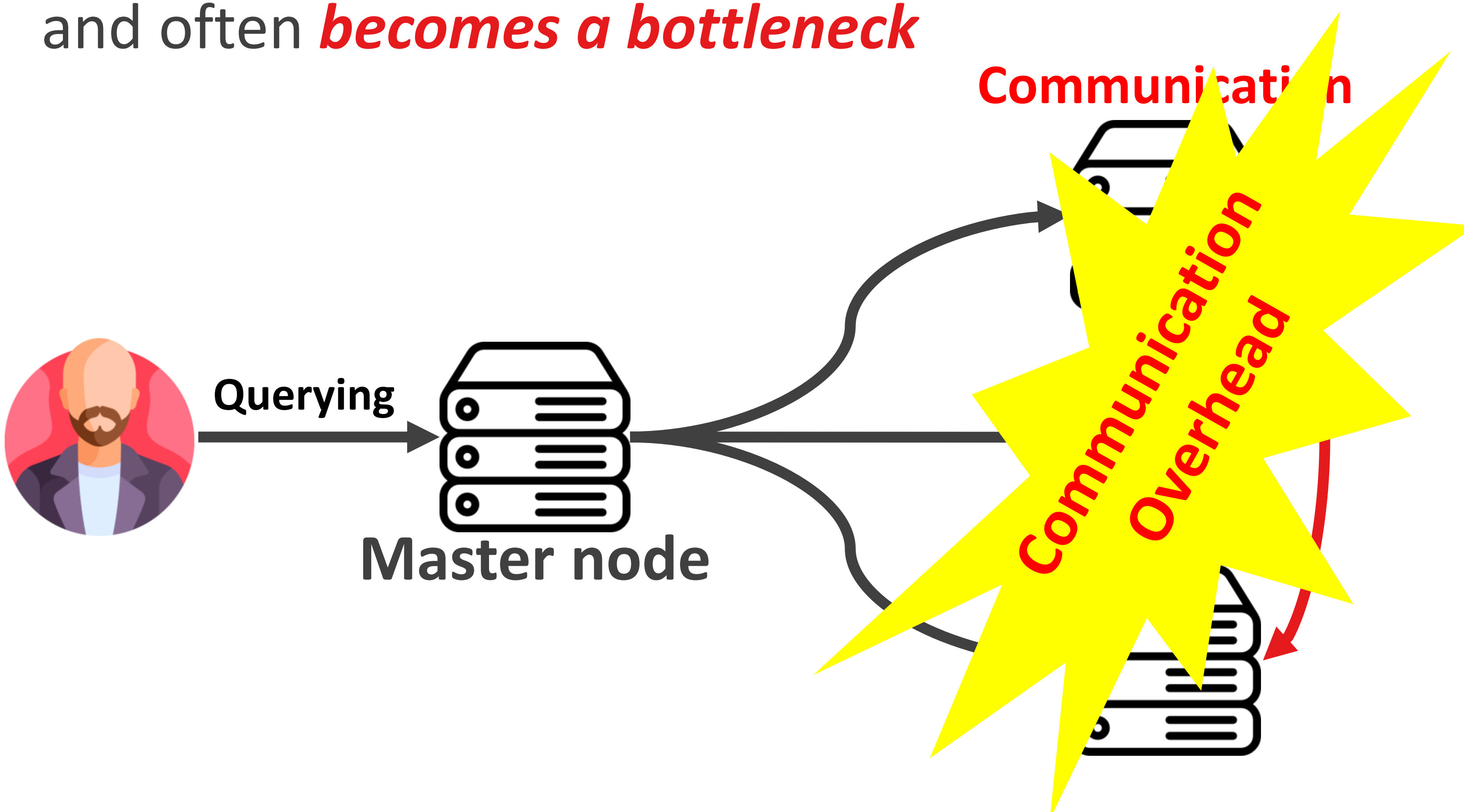
Motivation: query answering

- Given a query, multiple ***workers communicate with each other*** to answer it



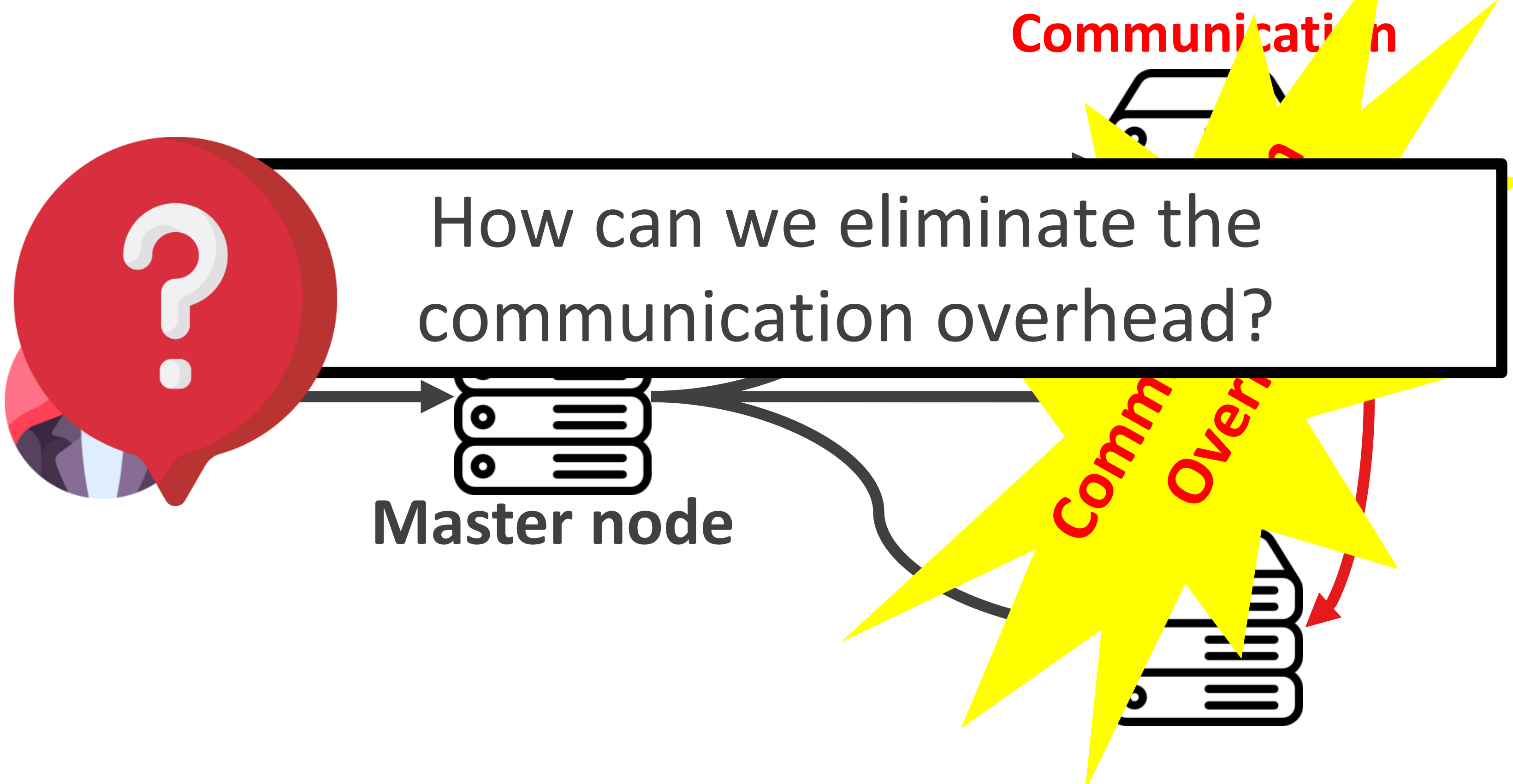
Motivation: bottleneck

- Such ***communication*** causes a significant overhead and often ***becomes a bottleneck***



Motivation: bottleneck

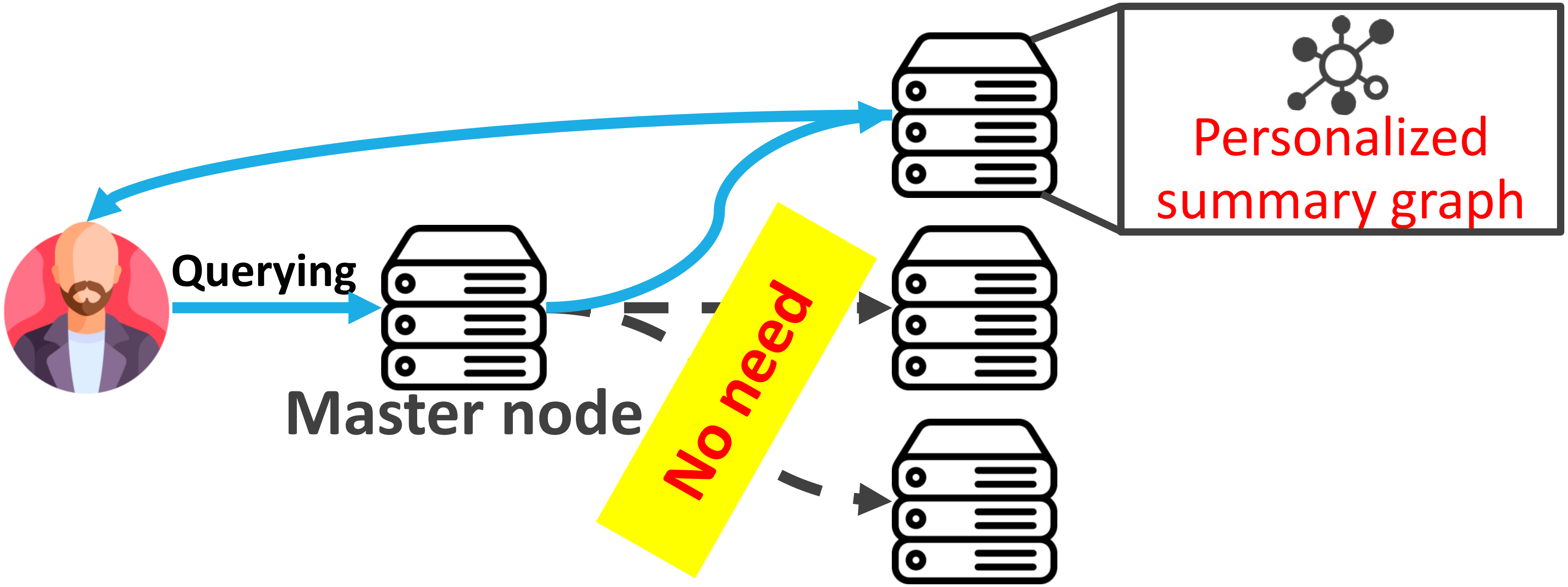
- Such ***communication*** causes a significant overhead and often ***becomes a bottleneck***



How can we eliminate the communication overhead?

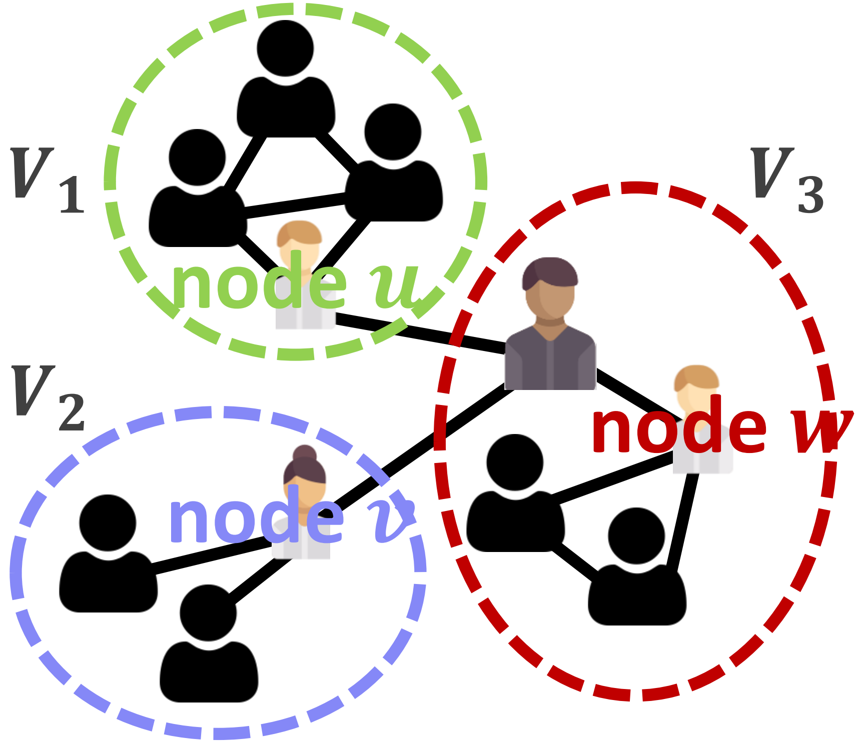
Application: overview

- Get multiple ***summary graphs with different targets***
  - Each summary graph ***fit in main memory*** *of a worker*
- Each query is answered by a worker with a "proper" summary graph ***without communications***



Application: preprocessing

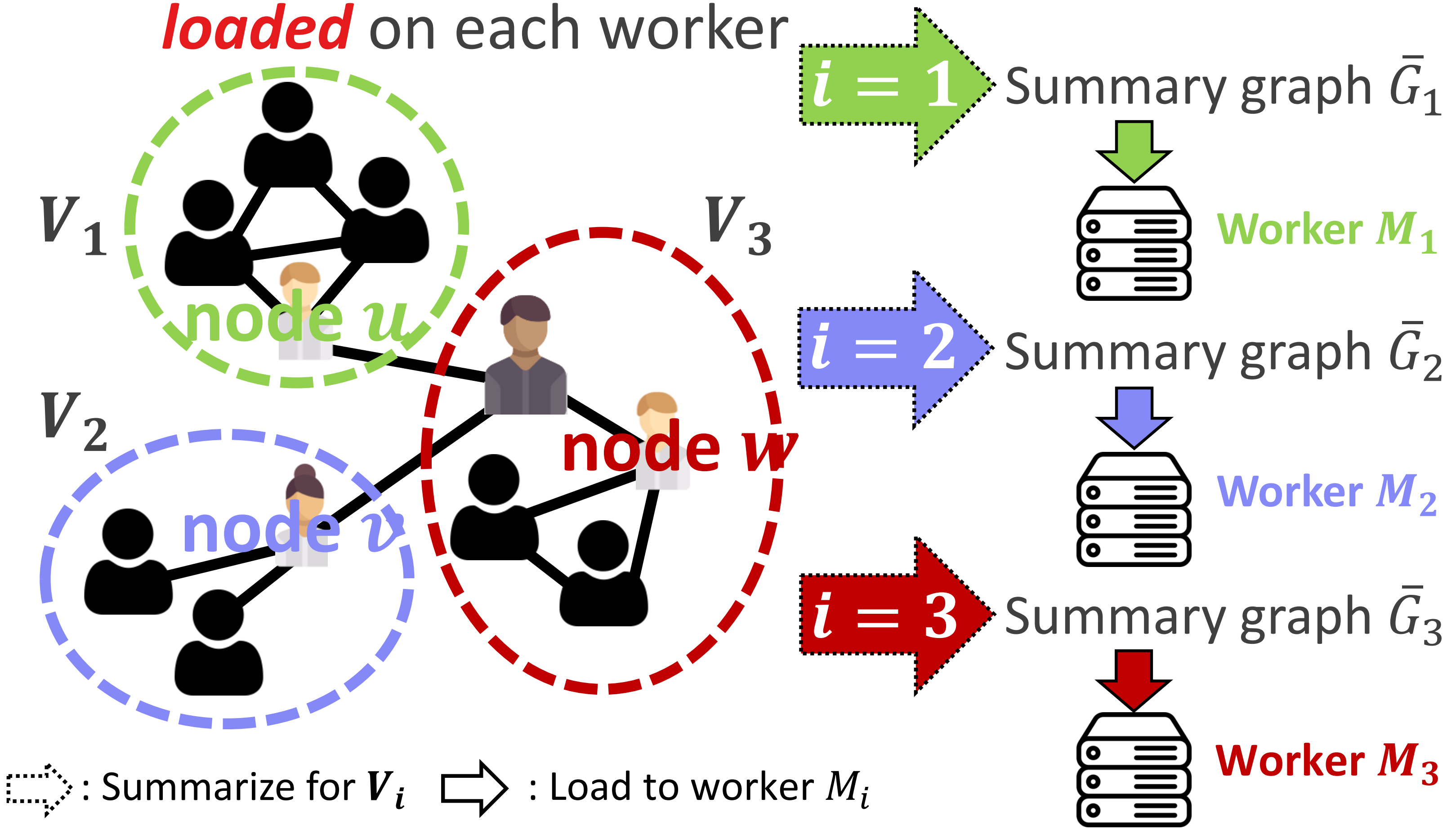
- ***Divide*** nodes into m subsets via graph partitioning
  - E.g., the Louvain method [7]



Input graph $G = (V, E)$

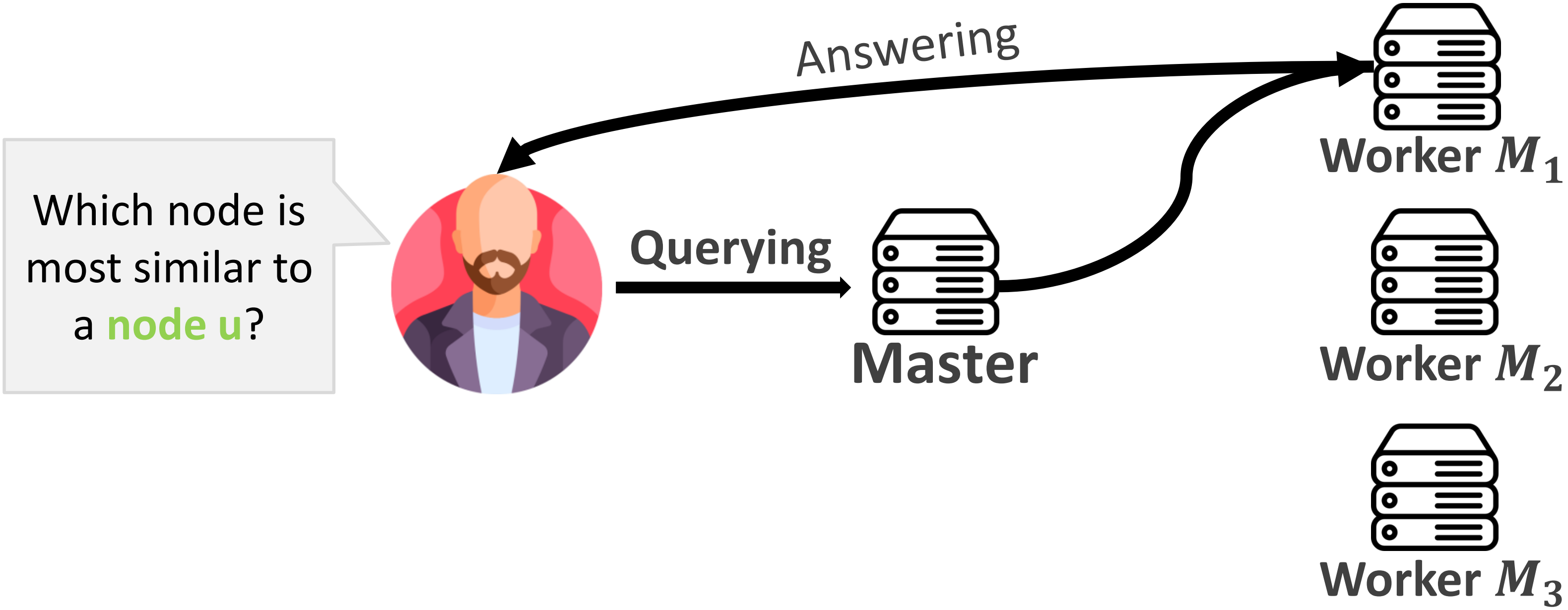
Application: preprocessing

- A summary graph personalized to each subset is ***loaded*** on each worker



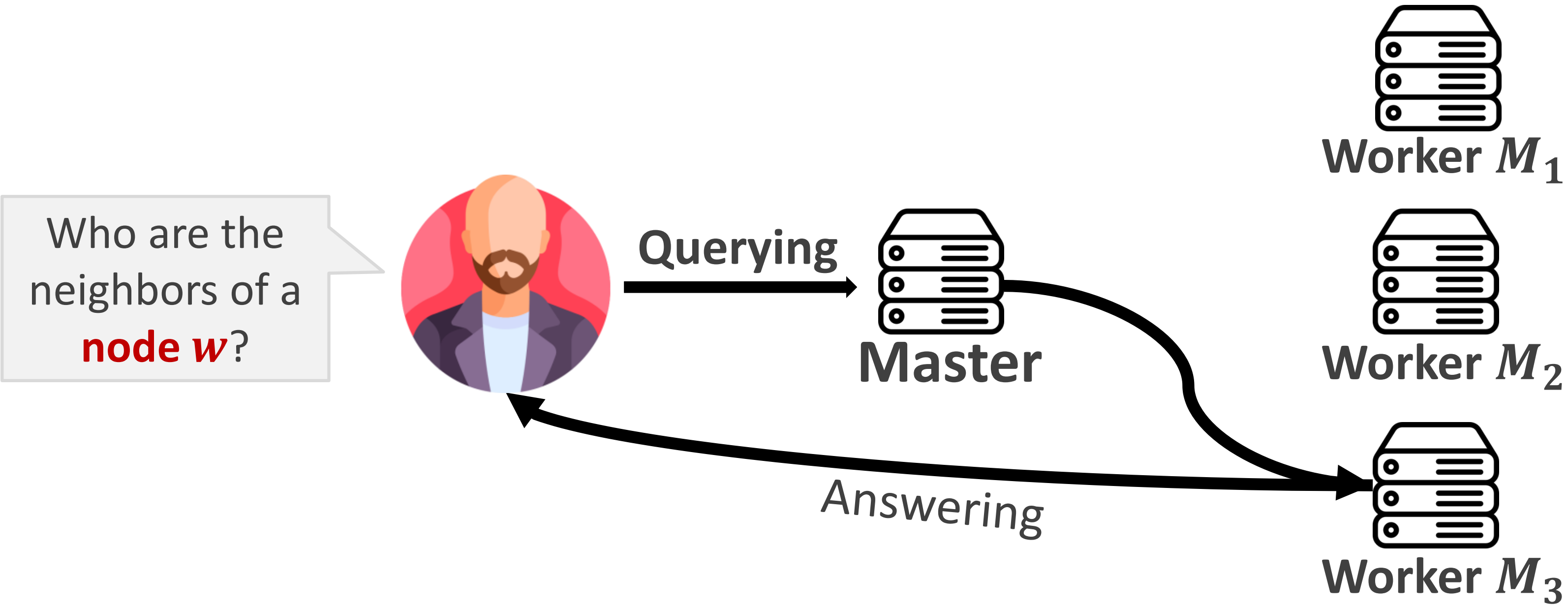
Application: query answering

- Each query is answered by ***a single worker without communications***
- Queries about ***node u*** are answered by the worker with the ***summary graph personalized to the subset with node u***



Application: query answering

- Answers are ***approximate but accurate***
  - Summary graphs used have abundant information about query nodes
- Multiple queries can be answered ***in parallel***
  - Workers perform independently



Road map

- ✓ Introduction
- ✓ Problem formulation
- ✓ Optimization: PeGaSus
- ✓ Application
- ✓ **Experiments <<**
- ✓ Conclusion

Experiments: settings

- Datasets
  - 6 Real-world graphs (27K - 0.1B edges)
  - 10 Synthetic graphs (up to ***1B*** edges)

Social | Collaboration | Internet | Co-purchase | Hyperlinks

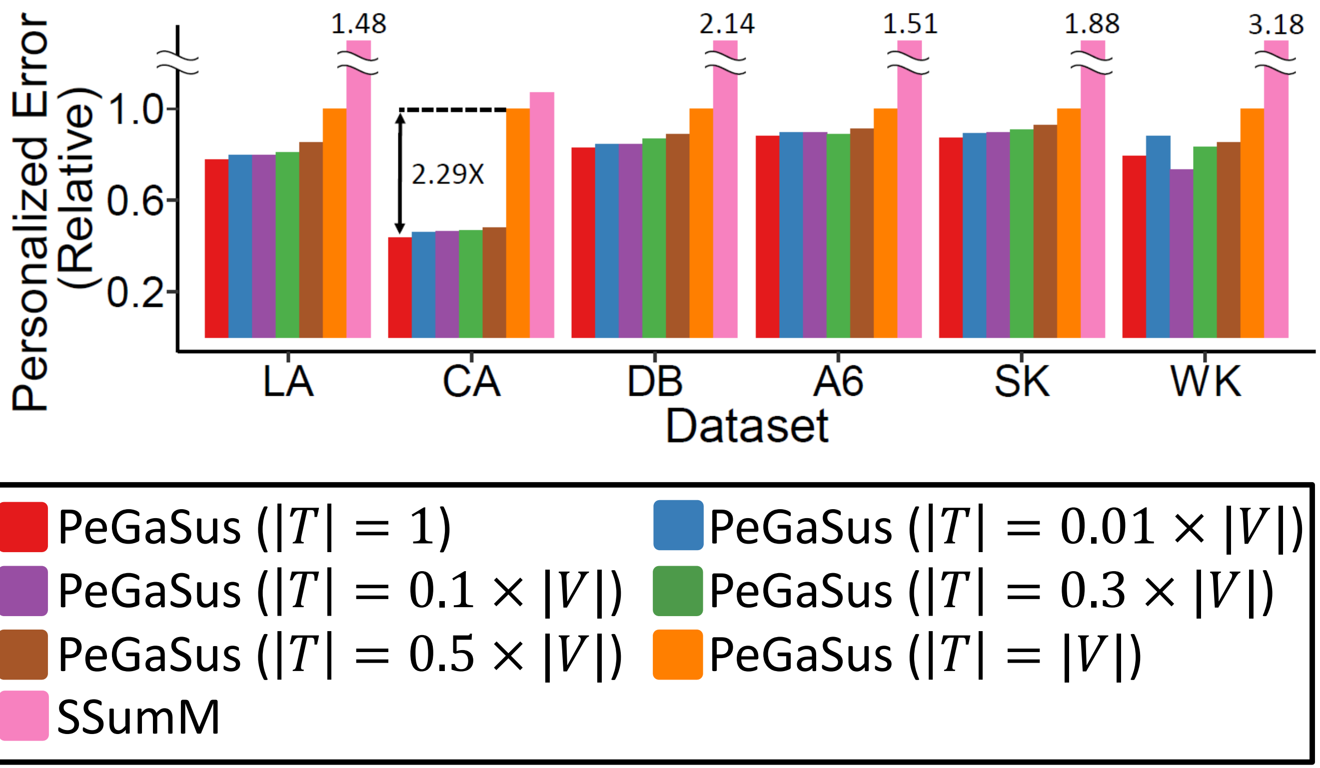
- Graph summarization methods
  - SSumM [1], k-Grass [2], SAAGs [3], S2L [4]
- Graph partitioning methods
  - Louvain [7], SHP [8], BLP [9]

Experiments: settings & metrics

- Node similarity queries
  - Random Walk with Restart (RWR) [10]
  - Length of shortest path (HOP)
  - Penalized Hitting Probability (PHP) [11, 12]
- Evaluation measures
  - Symmetric Mean Absolute Percentage Error (SMAPE) [13]
  - Spearman' correlation coefficients (Spearman Corr.) [14]
- Set of target nodes: T
  - Sample $|T|$ nodes uniformly at random

Q1. Personalization

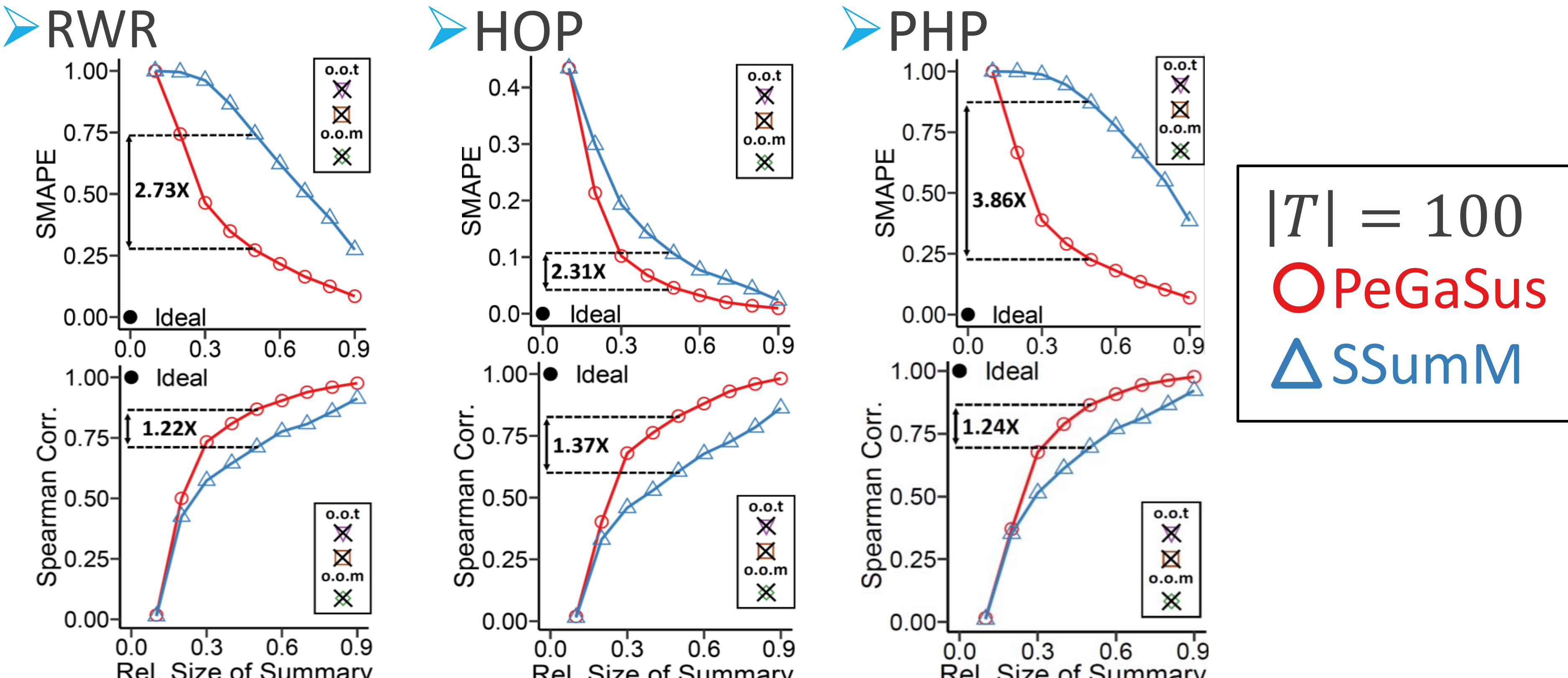
- PeGaSus provides "personalized" summary, ***well preserving the information*** close to target nodes T



Q2. Effectiveness

Dataset: amazon

- Queries were ***answered*** up to ***3.86X more accurately*** on personalized summary graphs

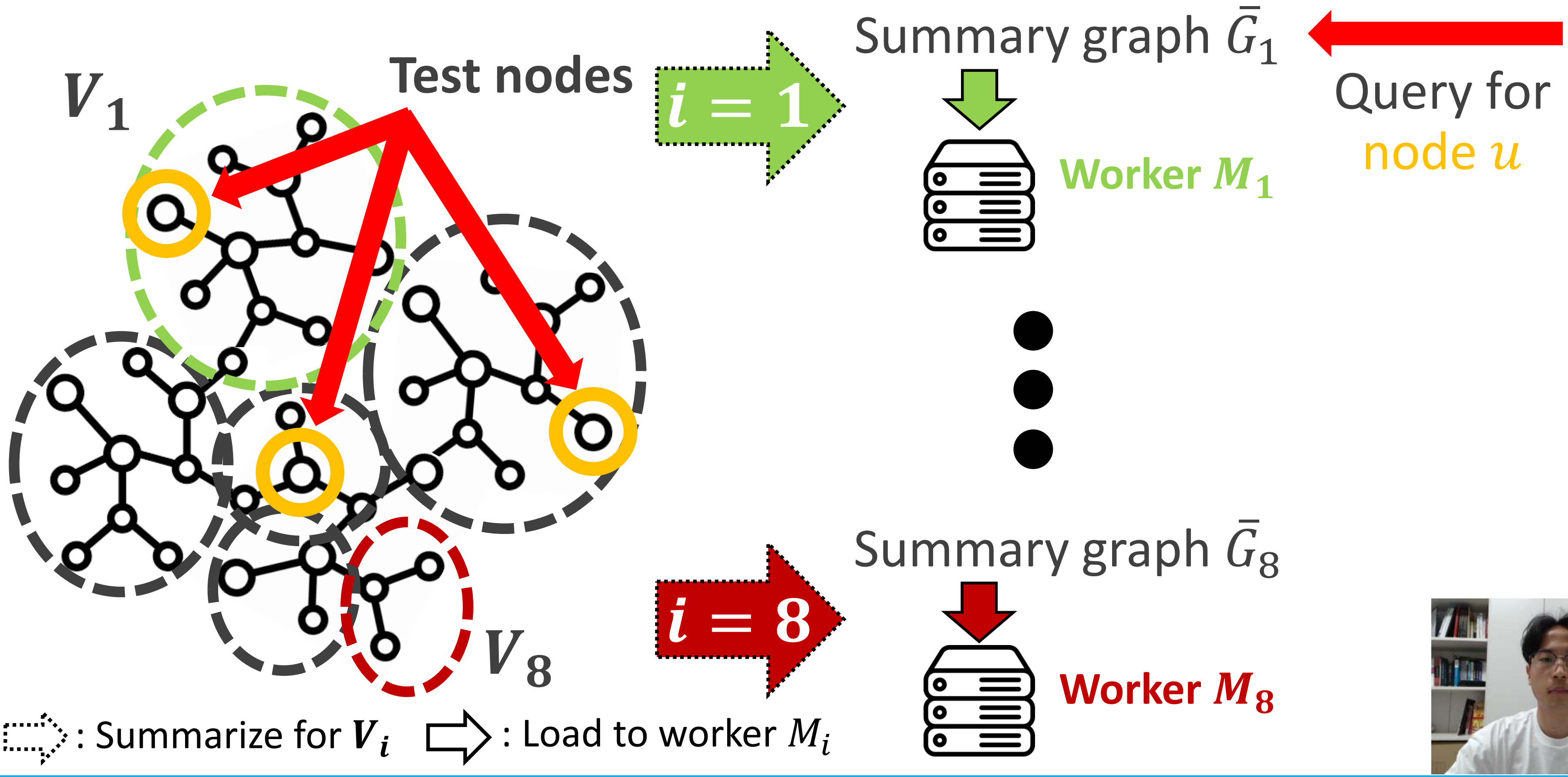


PeGaSus is ...

- [x] Effective in personalization
- [ ] Useful for applications
- [ ] Scalable to large graphs

Q3. Applicable: settings

- ***Eight*** personalized summary graphs on ***eight*** workers

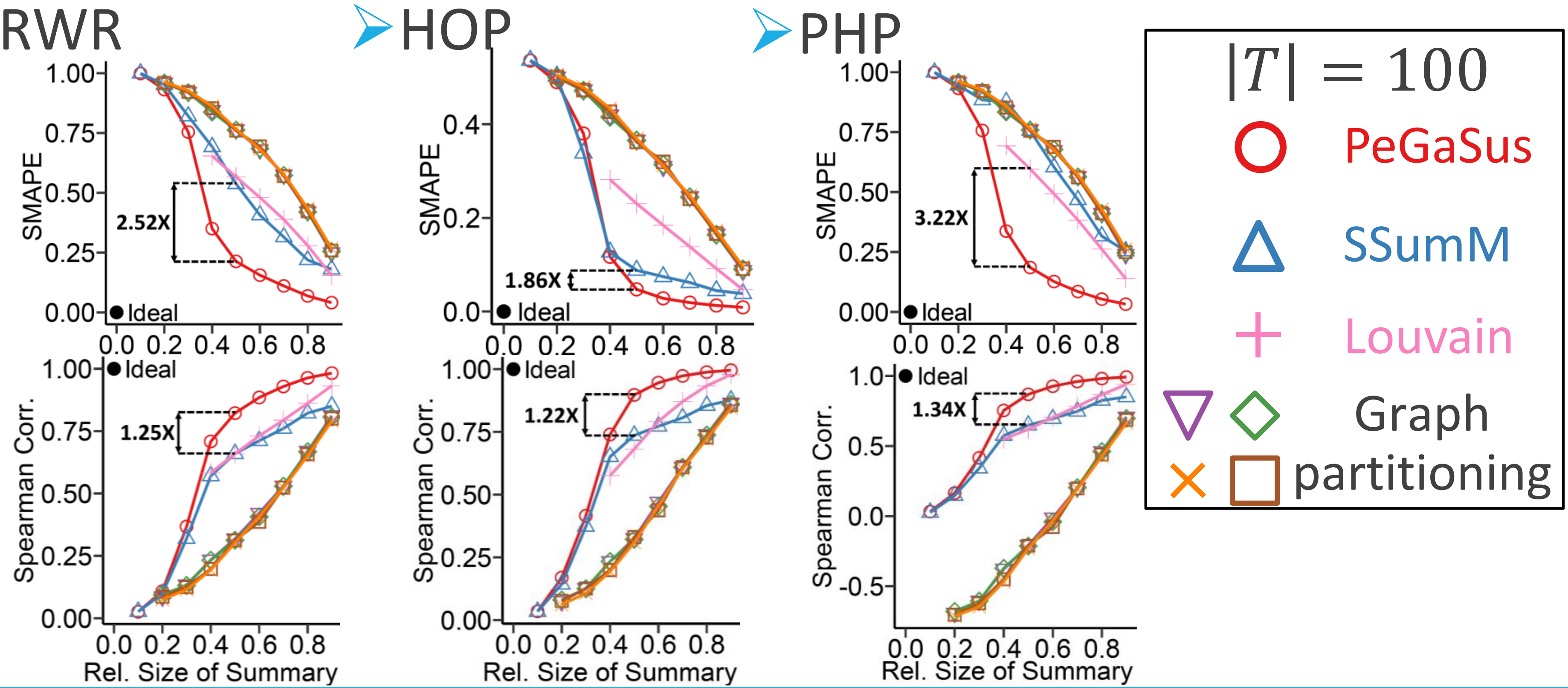


Q3. Applicable: results

Dataset: caida

- Queries were ***answered*** up to ***3.22X more accurately*** on personalized summary graphs

➢ RWR ➢ HOP ➢ PHP

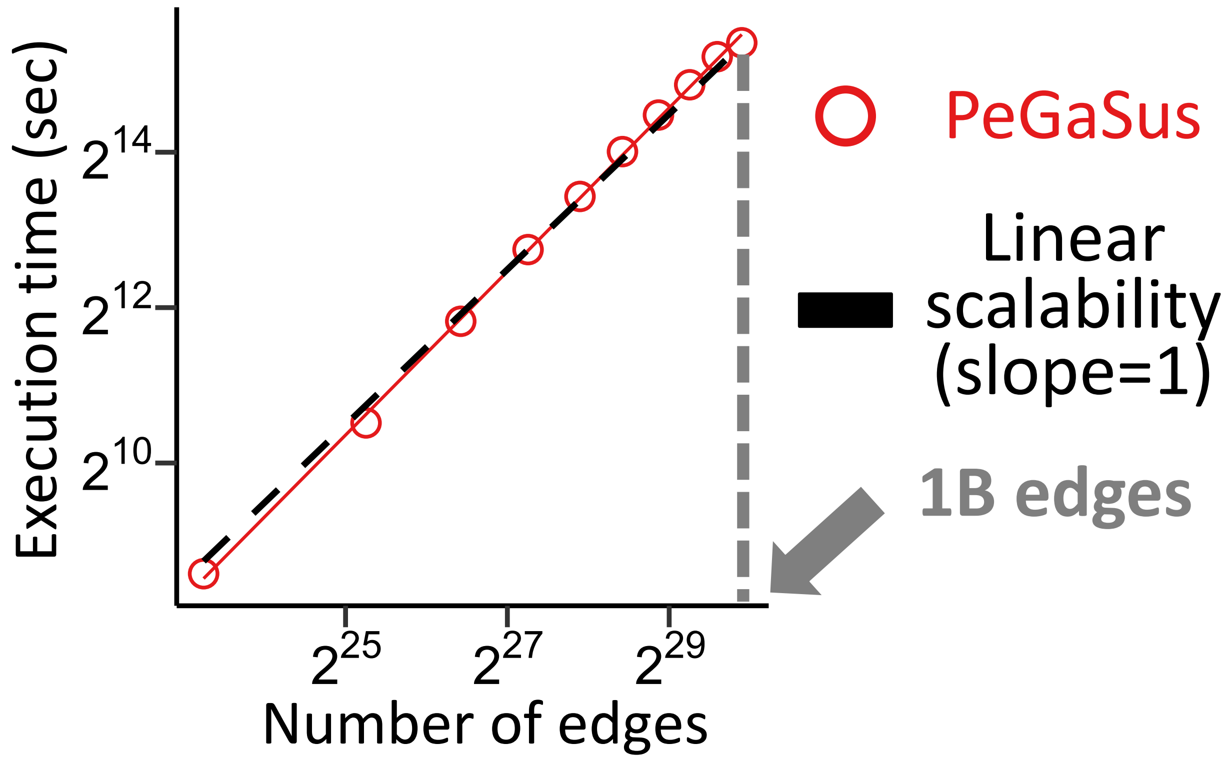


PeGaSus is …

- ☑ Effective in personalization
- ☑ Useful for applications
- ☐ Scalable to large graphs

Q4. Scalable

- PeGaSus ***scales linearly*** with the number of edges, to about 1B edges
  - Consistent with our theoretical analysis (Theorem 1)



PeGaSus is …

- ☑ Effective in personalization
- ☑ Useful for applications
- ☑ Scalable to large graphs

Road map

- ✓ Introduction
- ✓ Problem formulation
- ✓ Optimization: PeGaSus
- ✓ Application
- ✓ Experiments
- ✓ **Conclusion <<**

Conclusion

- We introduce a novel problem, ***personalized graph summarization***
- We propose ***PeGaSus***, an optimization algorithm for the problem

☑ **Effective in personalization**

☑ **Useful for applications**

☑ **Scalable to large graphs**

Github Link: https://github.com/ShinhwanKang/ICDE22-PeGaSus

References

- [1] K. Lee, H. Jo, J. Ko, S. Lim, and K. Shin, “Ssumm: Sparse summarization of massive graphs,” in KDD, 2020
- [2] K. LeFevre and E. Terzi, “Grass: Graph structure summarization,” in SDM, 2010
- [3] M. A. Beg, M. Ahmad, A. Zaman, and I. Khan, “Scalable approximation algorithm for graph summarization,” in PAKDD, 2018
- [4] M. Riondato, D. Garćıa-Soriano, and F. Bonchi, “Graph summarization with quality guarantees,” DMKD, vol. 31, no. 2, pp. 314–349, 2017
- [5] Tobler W., (1970) "A computer movie simulating urban growth in the Detroit region". *Economic Geography*, 46 (Supplement): 234–240
- [6] K. Shin, A. Ghoting, M. Kim, and H. Raghavan, “Sweg: Lossless and lossy summarization of web-scale graphs,” in WWW, 2019
- [7] V. D. Blondel, J.-L. Guillaume, R. Lambiotte, and E. Lefebvre, “Fast unfolding of communities in large networks,” JSTAT, vol. 2008, no. 10, p. P10008, 2008.
- [8] I. Kabiljo, B. Karrer, M. Pundir, S. Pupyrev, A. Shalita, A. Presta, and Y. Akhremtsev, “Social hash partitioner: a scalable distributed hypergraph partitioner,” arXiv preprint arXiv:1707.06665, 2017.

References

- [9] J. Ugander and L. Backstrom, "Balanced label propagation for partitioning massive graphs," in WSDM, 2013.
- [10] H. Tong, C. Faloutsos, and J.-Y. Pan, "Random walk with restart: fast solutions and applications," KAIS, vol. 14, no. 3, pp. 327–346, 2008.
- [11] C. Zhang, L. Shou, K. Chen, G. Chen, and Y. Bei, "Evaluating geo-social influence in location-based social networks," in CIKM, 2012.
- [12] Z. Guan, J. Wu, Q. Zhang, A. Singh, and X. Yan, "Assessing and ranking structural correlations in graphs," in SIGMOD, 2011
- [13] P. Goodwin and R. Lawton, "On the asymmetry of the symmetric mape," International journal of forecasting, vol. 15, no. 4, pp. 405–408, 1999.
- [14] C. Spearman, "The proof and measurement of association between two things." 1961

Personalized Graph Summarization: Formulation, Scalable Algorithms, and Applications

Shinhwan Kang Kyuhan Lee Kijung Shin

KAIST AI