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
Graphs become large!

• Graphs *grow rapidly* at an unprecedented pace

25 million (2013) 157.4 million (2022) 0.6 billion (2013) 1.9 billion (2022)
Graphs become large!

- Graphs *grow rapidly* at an unprecedented pace

<table>
<thead>
<tr>
<th>Users</th>
<th>Web pages</th>
</tr>
</thead>
<tbody>
<tr>
<td>25 million (2013)</td>
<td>0.6 billion (2013)</td>
</tr>
<tr>
<td>157.4 million (2022)</td>
<td>1.9 billion (2022)</td>
</tr>
</tbody>
</table>
Graphs become large!

- Graphs *grow rapidly* at an unprecedented pace

![Amazon](image)

<table>
<thead>
<tr>
<th>Users</th>
<th>2013</th>
<th>2022</th>
</tr>
</thead>
<tbody>
<tr>
<td>25 million</td>
<td></td>
<td>157.4 million</td>
</tr>
</tbody>
</table>

![Web](image)

<table>
<thead>
<tr>
<th>Web pages</th>
<th>2013</th>
<th>2022</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.6 billion</td>
<td></td>
<td>1.9 billion</td>
</tr>
</tbody>
</table>

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 $\bar{G}$*
  ○ (e.g.) Manhattan distance between adjacency matrices

---

**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)

Wrong!

Restoration process:
described in the paper
Graph summarization

• **Given**: input graph $G$ and a budget $k$
• **Find**: summary graph $\bar{G}$
• **To minimize**: the difference between $G$ and $\bar{G}$
  ◦ (e.g.) Manhattan distance between adjacency matrices
• **Subject to**: size of summary graph $\bar{G} \leq k$
  ◦ (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)

Wrong!
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 $\tilde{G} = (S, P)$
  
  *personalized to* $T$

• **To minimize**: error *personalized to* $T$

• **Subject to**: $\text{Size}(\tilde{G}) = \# \text{ of bits to encode } \tilde{G} \leq k$
Personalized error

- Personalized error is the \textit{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 \( T \).
Personalized weight

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

\[ W_{ij}^{(T)} \propto \alpha^{-\left(D(i,T)+D(j,T)\right)} \]

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

(e.g.) Personalized weights are
Road map

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

- *Personalized* Graph *S*ummarization 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 $\tilde{G}$

• Procedure
  ➢ initializing step
  ➢ repeat $t_{max}$ times or until $\text{Size}(\tilde{G}) > k$
  ➢ dividing step & merging step
  ➢ If $\text{Size}(\tilde{G}) > k$, then sparsifying step
Initializing step

➢ Initialize a summary graph $\tilde{G}$, and a threshold $\theta_0$
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 $Saving^{(T)} > \theta(0)$, merge supernodes

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

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

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

➢ Divide supernodes into groups $U$ by MinHashing

* MinHashing gives different partitions in each iteration

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

➢ For each group of $U$, if $\text{Saving}^{(T)} > \theta_{(1)}$, *merge* supernodes.
Sparsifying step

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

---

Diagram showing the sparsification process in a graph with nodes $A \cup B \cup C \cup D \cup E \cup F \cup G$. The process involves isolating superedges and reducing the graph's complexity.
Adaptive threshold: motivation

➢ In the merging step,
➢ If $\text{Saving}^{(T)} > \theta(.)$, merge supernodes
➢ ...

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

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

$\theta_{(t+1)} = \text{Top 10% entry}$
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*
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 \textit{loaded} on each worker.

\begin{itemize}
  \item $V_1$ : Load to worker $M_1$
  \item $V_2$ : Load to worker $M_2$
  \item $V_3$ : Load to worker $M_3$
\end{itemize}
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 $u$. 

Diagram:
- A user queries which node is most similar to a node $u$.
- The query is directed to the master.
- The master routes the query to the worker $M_1$.
- Worker $M_1$ processes the query and sends the answer back to the master.
- The master then answers the user.

Which node is most similar to a 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

Who are the neighbors of a node $w$?
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)

• 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

- Queries were \textit{answered} up to \textbf{3.86X more accurately} on personalized summary graphs

- **RWR**
  - SMAPE:
    - Ideal: 1.22X
    - O.o.t: 2.73X

- **HOP**
  - SMAPE:
    - Ideal: 1.37X
    - O.o.t: 2.31X

- **PHP**
  - SMAPE:
    - Ideal: 1.24X
    - O.o.t: 3.86X

\[ |T| = 100 \]

Dataset: \textbf{amazon}
PeGaSus is ...

- Effective in personalization
- Useful for applications
- Scalable to large graphs
Q3. Applicable: settings

- Eight personalized summary graphs on eight workers

$V_1$, Test nodes

$V_8$

Summary graph $\bar{G}_1$

Query for node $u$

Summary graph $\bar{G}_8$

$: Summarize for $V_i$  : Load to worker $M_i$
Q3. Applicable: results

• Queries were *answered* up to **3.22X more accurately** on personalized summary graphs.
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](https://github.com/ShinhwanKang/ICDE22-PeGaSus)
References

References

Personalized Graph Summarization: Formulation, Scalable Algorithms, and Applications

Shinhwan Kang    Kyuhan Lee    Kijung Shin

KAIST AI