Node Embedding Preserving Graph Summarization

HOUQUAN ZHOU, CAS Key Laboratory of AI Security, Institute of Computing Technology, Chinese Academy of Sciences, Haidian District, China and University of Chinese Academy of Sciences, Huairou District, China
SHENGHUA LIU, CAS Key Laboratory of AI Security, Institute of Computing Technology, Chinese Academy of Sciences, Haidian District, China and University of Chinese Academy of Sciences, Huairou District, China
HUAWEI SHEN, CAS Key Laboratory of AI Security, Institute of Computing Technology, Chinese Academy of Sciences, Haidian District, China and University of Chinese Academy of Sciences, Huairou District, China
XUEQI CHENG, CAS Key Laboratory of AI Security, Institute of Computing Technology, Chinese Academy of Sciences, Haidian District, China and University of Chinese Academy of Sciences, Huairou District, China

Graph summarization is a useful tool for analyzing large-scale graphs. Some works tried to preserve original node embeddings encoding rich structural information of nodes on the summary graph. However, their algorithms are designed heuristically and not theoretically guaranteed. In this article, we theoretically study the problem of preserving node embeddings on summary graph. We prove that three matrix-factorization-based node embedding methods of the original graph can be approximated by that of the summary graph, and we propose a novel graph summarization method, named HCSumm, based on this analysis. Extensive experiments are performed on real-world datasets to evaluate the effectiveness of our proposed method. The experimental results show that our method outperforms the state-of-the-art methods in preserving node embeddings.

CCS Concepts: • **Information systems** → **Data mining**; *Social networks;*

Additional Key Words and Phrases: Graph summarization, hierarchical clustering, node embedding

ACM Reference Format:
Houquan Zhou, Shenghua Liu, Huawei Shen, and Xueqi Cheng. 2024. Node Embedding Preserving Graph Summarization. *ACM Trans. Knowl. Discov. Data.* 18, 6, Article 145 (April 2024), 19 pages. https://doi.org/10.1145/3649505

This article is partially supported by the National Science Foundation of China under Grants No. U21B2046 and No. 6237075198.
Authors' address: H. Zhou, S. Liu (Corresponding author), H. Shen, and X. Cheng, CAS Key Laboratory of AI Security, Institute of Computing Technology, Chinese Academy of Sciences, 6 Zhongguancun South Street, Haidian District, Beijing, China, 100190 and University of Chinese Academy of Sciences, No. 1 Yanqihu East Road, Huairou District, Beijing, China; e-mails: zhouhouquan18z@ict.ac.cn, liushenghua@ict.ac.cn, shenhuawei@ict.ac.cn, cxq@ict.ac.cn.

ACM 1556-4681/2024/04-ART145
https://doi.org/10.1145/3649505

1 INTRODUCTION

Graphs are widely used to represent various objects in real-world and relationships among them, including social networks, computer networks, and transportation networks, and so on. And graph-related applications have been widely studied in various fields [21, 38]. Recent years have witnessed the explosive growth of data size and such large scale brings great challenge to processing, analyzing and understanding graph data. To tackle this problem, some researchers resort to **graph summarization**. Given a graph $\mathcal{G}$, graph summarization finds a compact representation of it. The typical form is a *summary graph* by grouping nodes in $\mathcal{G}$ into *supernodes* and aggregating edges in $\mathcal{G}$ into *superedges*. Figure 1 shows a small example of graph summarization. The original graph with nine nodes are summarized into a summary graph with three supernodes and six superedges. The summary graph is smaller and easier to process and analyze than the original graph and thus can be used to analyze the original graph [8, 23, 26, 29, 36].

Generally, a good summary graph is expected to keep the properties of the original graph. Most graph summarization methods aims to preserve the adjacency matrix. However, the adjacency matrix is only the most fundamental representation of a graph and fails to represent the high-order properties of a graph. However, node embedding methods have shown great power in capturing the structural properties and have become a fundamental tool in graph mining. Typically, node embedding methods learn low-dimensional representations of nodes in a graph, which can be used for various downstream tasks, such as link prediction, node classification, and anomaly detection. Moreover, graph summarization may capture high-order relations and help learn high-quality node embeddings [3, 22]. Thus, it is important to preserve the node embeddings of the original graph in the summary graph.

Several studies have attempted to learn node embeddings for large-scale graphs by combining graph summarization and node embedding methods. They first summarize input graphs into smaller summary graphs, and then learn summary embeddings on them, which are subsequently recovered to approximate the original node embeddings. The main objective of these approaches is to preserve the node embeddings of the original graph in the summary graph. Despite the empirical success of these methods, there is a key limitation that they summarize input graphs heuristically and do not investigate the theoretical connection between input graphs and summary graphs.

In this work, we study the theoretical connection between them in node embedding methods. We analyze three matrix-factorization-based node embedding methods, namely, NetMF [33], DeepWalk [32], and LINE [40]. These three methods learn node embeddings by factorizing the proximity matrix [33] of the input graph. By showing that the proximity matrix in these methods can be approximated by the one of the summary graph, we provide theoretical foundation for learning node embeddings via summary graphs. We further analyze the error raised by the summarization and relate it to a trace optimization problem. Based on the analysis, propose a novel graph **Summ**arization method based on **H**ierarchical **C**lustering, named HCSumm, to minimize the error. We conduct extensive experiments on several real-world datasets and show that our method outperforms several state-of-the-art methods with better summary.

In summary, our contributions include:

- **Theory:** We reveal the theoretical connection between the proposed scheme and three node-embedding learning methods, which provides theoretical foundation for learning node embeddings via summary graphs.
- **Method:** Based on the theoretical analysis, we propose a graph summarization method HCSumm based on hierarchical clustering.

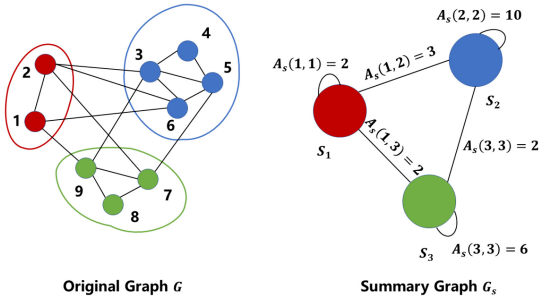


Fig. 1. Example of graph summarization.

— **Effectiveness:** We perform extensive experiments on several real-world datasets and the experimental results show that our HCSumm algorithm outperforms the state-of-the-art methods with better node embedding preservation.
— **Scalability:** Our HCSumm algorithm runs fast and scales linearly in the size of graphs.

2 RELATED WORK

2.1 Graph Summarization

Graph summarization methods can be categorized based on many aspects. Here, we categorize them according to their objectives. See the comprehensive survey [25] for more knowledge about this topic.

Error of adjacency matrix: These methods try to minimize some error metrics between the original and reconstructed adjacency matrices and are the main focus of this article. k-Gs [20] aimed to find a summary graph with at most k supernodes, such that the L1 reconstruction error is minimized. Riondato et al. [35] revealed the connection between the geometric clustering problem and the graph summarization problem under multiple error metrics (including L1 error, L2 error and cut-norm error), and they proposed a polynomial-time approximate graph summarization method based on geometric clustering algorithms. Beg et al. [2] developed a randomized algorithm SAA-Gs using weighted sampling and count-min sketch [4] techniques to find promising node pairs efficiently. SpecSumm [27] reformulate the graph summarization problem as a trace optimization problem and propose a spectral algorithm based on k-means clustering on the eigenvectors of the adjacency matrix.

Total edge number: In this kind of method, the objective function is defined as number of edges in summary graph plus edge corrections. In Reference [28], Navlakha et al. proposed two algorithms: Greedy and Randomized. The former considers all possible node pairs at each step, and merges the best pair (u, v), which results in the greatest decrease of the total edge number. The latter samples a supernode as u randomly at each step, checks all other supernodes, finds the best v and merges them together. This process continues until the summary graph becomes smaller than a given size. However, both algorithms are computationally expensive. To address this problem, SWeG [39] reduces the search space by grouping supernodes, according to their shingle values, and only considers merging node pairs in the same groups. Reference [44] further uses weighted LSH and scales to large graphs with tens of billions of edges.

Encoding length: These kinds of methods often adopt the MDL principle and use the total encoding length as the objective function. They typically optimize the total description length under their proposed encoding scheme. LeFevre and Terzi [20] formulated the graph summarization problem Gs based on the MDL principle, and they proposed three algorithms Greedy, SamplePairs, and LinearCheck. Lee et al. [19] designed a dual-encoding scheme and proposed

Table 1. Major Symbols and Definitions

Symbol	Definition
$\mathcal{G}$=($\mathcal{V}, \mathcal{E}$)	Original graph with nodeset $\mathcal{V}$ and edgeset $\mathcal{E}$
$\mathcal{G}_s$=($\mathcal{V}_s, \mathcal{E}_s$)	Summary graph with supernodes $\mathcal{V}_s$ and superedges $\mathcal{E}_s$
$\mathcal{G}_r$=($\mathcal{V}, \mathcal{E}_r$)	Reconstructed graph with nodeset $\mathcal{V}$ and edgeset $\mathcal{E}_r$
v_i	Node i in the original graph $\mathcal{G}$
$\mathcal{S}_k$	Supernode k in the summary graph $\mathcal{G}_s$
d_i, D_k	Degree of node i and supernode k
$\mathbf{A}, \mathbf{A}_s, \mathbf{A}_r$	Adjacency matrix of original, summary, reconstructed graphs
$\mathbf{D}, \mathbf{D}_s$	Degree matrix of original and summary graphs
$\mathbf{L}, \mathbf{L}_s, \mathbf{L}_r$	(Combinatorial) Laplacian matrices of original, summary, reconstructed graphs
$\mathcal{A}, \mathcal{L}$	Normalized adjacency matrix and normalized Laplacian matrix
$\mathbf{P}, \mathbf{Q}$	Membership and reconstruction matrix in summarization
$\mathbf{R}$	Restoration matrix for recovering the original embeddings
$\mathbf{E}, \mathbf{E}_s$	Embeddings of original graph and summary graph

a sparse summarization algorithm SSumM, which reduces the number of node and sparsifies the graph simultaneously. By dropping less important edges and encoding them as errors, SSumM is able to obtain a compact and sparse summary graph. Different from methods mentioned above, VoG [18] adopted a vocabulary-based encoding scheme, which encodes the graph using frequent patterns in real-world graphs, such as cliques, stars, and bipartite cores.

Methods mentioned above mainly focus on static simple graphs. There are works aiming to summarize other types of graphs, including dynamic graphs [1, 34, 37], attributed graphs [11, 16, 42], and streaming graphs [17, 41].

2.2 Graph Summarization Preserving Node Embeddings

There are some existing works that aim to learn node embeddings via summary graphs [25, 43]. The typical approach is to coarsen the original graph into a smaller summary graph and apply representation learning methods on it to obtain intermediate embeddings. The embeddings of the original nodes are then restored with a further refinement step. For example, HARP [3] finds a series of smaller graphs that preserve the global structure of the input graph and learns representations hierarchically. HSRL [9] learns embeddings on multi-level summary graphs, and concatenate them to restore original embeddings. MILE [22] repeatedly coarsens the input graph into smaller ones using a hybrid matching strategy, and finally refines the embeddings via GCN to obtain the original node embeddings. GPA [24] uses METIS [15] to partition the graphs, and smooths the restored embeddings via a propagation process. GraphZoom [5] employs an extra graph fusion step to combine the structural information and feature information, and then uses a spectral coarsening method to merge nodes based on their spectral similarities. Embeddings are then refined by a graph filter to ensure feature smoothness. Reference [7] learns embeddings of the given subset of nodes by coarsening the remaining nodes, which is not capable to learn embeddings of the remaining ones.

3 CR RECONSTRUCTION SCHEME

In this section, we introduce the **configuration-based reconstruction (CR)** scheme after introducing some basic concepts of graph summarization. We list the frequently used symbols in Table 1 for readability.

3.1 Graph Summarization and Reconstruction Scheme

Given an input graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ with $n = |\mathcal{V}|$ nodes, graph summarization aims to find a smaller summary graph $\mathcal{G}_s = (\mathcal{V}_s, \mathcal{E}_s)$ (with $n_s = |\mathcal{V}_s|$ nodes) that preserves the structural information of the original graph. The supernode set $\mathcal{V}_s$ forms a partition of the original node set $\mathcal{V}$ such that every node $v \in \mathcal{V}$ belongs to exactly one supernode $\mathcal{S} \in \mathcal{V}_s$. The supernodes are connected via superedges $\mathcal{E}_s$, which are weighted by the sum of original edges between the constituent nodes. That is, superedge $\mathbf{A}_s(k, l)$ between supernodes $\mathcal{S}_k$, $\mathcal{S}_l$ is defined as

$$\mathbf{A}_s(k, l) = \sum_{v_i \in \mathcal{S}_k} \sum_{v_j \in \mathcal{S}_l} \mathbf{A}(i, j). \tag{1}$$

Degree of supernodes is defined as the sum of node degrees within supernode $\mathcal{S}_p$, i.e., $d_p^{(s)} = \sum_{i \in \mathcal{S}_p} d_i$. The adjacency matrix of the summary graph $\mathbf{A}_s$ can be formulated using a *membership matrix* $\mathbf{P} \in \mathbb{R}^{n_s \times n}$ as $\mathbf{A}_s = \mathbf{P}\mathbf{A}\mathbf{P}^\top$, where

$$\mathbf{P}(k, i) = \begin{cases} 1 & \text{if } v_i \in \mathcal{S}_k, \\ 0 & \text{otherwise.} \end{cases} \tag{2}$$

One could find a good summary graph by making the summary graph close to the original graph, for example, minimizing $\text{dis}(\mathcal{G}, \mathcal{G}_s)$ for some distance metric dis. However, the summary graph and the original graph have different size, and it is difficult to directly compare two graphs with different sizes.

This issue can be avoided by introducing a reconstructed graph. Given the summary graph $\mathcal{G}_s$, the original graph $\mathcal{G}$ can be approximated with the reconstructed graphs $\mathcal{G}_r$ with adjacency matrix $\mathbf{A}_r$ defined as

$$\mathbf{A}_r = \mathbf{Q}\mathbf{A}_s\mathbf{Q}^\top, \tag{3}$$

where $\mathbf{Q} \in \mathbb{R}^{n \times n_s}$ is the *reconstruction matrix*. The reconstructed graph $\mathcal{G}_r$ has the same size with the original graph $\mathcal{G}$, and is comparable to it. For example, one can minimize the difference of adjacency matrices $\|\mathbf{A} - \mathbf{A}_r\|$ for some matrix norm. Note that $\mathbf{A}_r$ can be seen as a **low-rank approximation** of the original $\mathbf{A}$.

A simple and intuitive reconstruction method is the *uniform* reconstruction scheme, which is widely applied in current works. The corresponding $\mathbf{Q}$ and $\mathbf{A}_r$ are

$$\mathbf{Q}(i, k) = \begin{cases} \frac{1}{|\mathcal{S}_k|} & \text{if } v_i \in \mathcal{S}_k, \\ 0 & \text{otherwise,} \end{cases} \tag{4}$$

$$\mathbf{A}_r(i, j) = \frac{1}{|\mathcal{S}_k|}\mathbf{A}_s(k, l)\frac{1}{|\mathcal{S}_l|}, \qquad v_i \in \mathcal{S}_k, v_j \in \mathcal{S}_l, \tag{5}$$

where $\mathcal{S}_k$ and $\mathcal{S}_l$ are the supernodes to which node i and node j belong, respectively.

It can be seen from Equation (5) that the edges between two supernodes $\mathcal{S}_p$ and $\mathcal{S}_l$, i.e., $\mathbf{A}_s(k, l)$, are equally assigned to each node pair between them, and each node pair has the same connection weight. Thus, this approach assumes the $G(n, p)$ random graph model (or Erdős-Rényi model equivalently) [6] and SBM (Stochastic Block Model) [12]. However, real-world graphs have highly **skewed degree distributions**. Therefore, this uniform reconstruction scheme is not suitable for real-world graphs.

Thus, we introduce the *configuration-based* reconstruction scheme [45]. Different from the uniform reconstruction scheme, it reconstructs $\mathbf{A}_r$ based on node degrees:

Definition 1. (CR Scheme) The **configuration-based reconstruction scheme (CR scheme)** calculates $\mathbf{A}_r(i,j)$ as follows:

$$\mathbf{A}_r(i,j) = \frac{d_i}{D_k}\mathbf{A}_s(k,l)\frac{d_j}{D_l}, \qquad v_i \in \mathcal{S}_k, v_j \in \mathcal{S}_l, \tag{6}$$

where $\mathcal{S}_k$ and $\mathcal{S}_l$ are the supernodes to which node i and node j belong, respectively. We use d_i and d_j to denote the degrees of nodes i and j; and D_k and D_l to denote the degrees of supernodes $\mathcal{S}_k$ and $\mathcal{S}_l$. The corresponding $\mathbf{Q}$ matrix is

$$\mathbf{Q}(i,k) = \begin{cases} \frac{d_i}{D_k} & \text{if } v_i \in \mathcal{S}_k, \\ 0 & \text{otherwise.} \end{cases} \tag{7}$$

In this way, the reconstructed edge weight $\mathbf{A}_r(i,j)$ is proportional to the product of endpoints' degrees. This approach is based on the configuration model [30] and the DC-SBM (degree-corrected stochastic block model) [14], which has proved successful in modularity-based community detection [31].

Note that the proposed CR scheme is able to preserve the degrees of nodes, as show below:

Property 1 (Degree Preservation).

$$\sum_{j=1}^{n} \mathbf{A}_r(i,j) = d_i = \sum_{j=1}^{n} \mathbf{A}(i,j)\,. \tag{8}$$

Proof.

$$\sum_j \mathbf{A}_r(i,j) = \sum_l \sum_{j\in\mathcal{S}_l} \frac{d_i}{D_k}\mathbf{A}_s(k,l)\frac{d_j}{D_l} = \sum_l \frac{d_i}{D_k}\mathbf{A}_s(k,l) = d_i. \qquad \square$$

Thus, we also call the CR scheme *degree-preserving* scheme.

4 CONNECTION WITH NODE EMBEDDING METHODS

In this section, we present the connection of the proposed CR scheme and three matrix-factorization-based node embedding methods: DeepWalk [32], LINE [40], and NetMF [33]. In short, we show that learning node embeddings on a summary graph with restoration is equivalent to learning embeddings on the reconstructed graph under the CR scheme.

4.1 Matrix-factorization-based Node Embedding Methods

DeepWalk. DeepWalk [32] is an unsupervised graph representation learning method inspired by the success of word2vec in text embedding. It generates random walk sequences and treats them as sentences that are later fed into a skip-gram model with negative sampling to learn latent node representations.

LINE. LINE [40] learns embeddings by optimizing a carefully designed objective function that aims to preserve both the first-order and second-order proximity.

NetMF. NetMF aims to unify some node embedding methods into a matrix factorization framework [33]. It shows that DeepWalk is implicitly approximating and factorizing the following proximity matrix:

$$\mathbf{M} := \log\left(\frac{\mathrm{vol}(\mathcal{G})}{b}\left(\frac{1}{T}\sum_{\tau=1}^{T}(\mathbf{D}^{-1}\mathbf{A})^{\tau}\right)\mathbf{D}^{-1}\right), \tag{9}$$

where T and b are the context window size and the number of negative samples in DeepWalk, respectively.

Similarly, LINE is equivalent to factorizing a similar matrix to Equation (9) and is a special case of DeepWalk for $T = 1$:

$$\mathbf{M} \coloneqq \log\left(\frac{\mathrm{vol}(\mathcal{G})}{b}\mathbf{D}^{-1}\mathbf{A}\mathbf{D}^{-1}\right).$$

We throw out the element-wise log function and constant factors away, and extract a form of *kernel matrix* defined as follows.

Definition 2 (Kernel Matrix).

$$\mathcal{K}_\tau(\mathcal{G}) \coloneqq (\mathbf{D}^{-1}\mathbf{A})^\tau\mathbf{D}^{-1}, \tag{10}$$

where τ is a positive integer, and $\mathbf{A}$ and $\mathbf{D}$ are adjacency matrix and degree matrix of $\mathcal{G}$, respectively. We omit the subscript τ if there is no ambiguity.

4.2 Approximating Kernel Matrix

Now, we show that, under **the configuration-based reconstruction scheme** [see Equations (6) and (7)], the kernel matrix on the original graph, $\mathcal{K}(\mathcal{G})$, can be approximated with the same kernel matrix on the summary graph, $\mathcal{K}(\mathcal{G}_s)$, in a closed form.

Theorem 1. *Given $\mathbf{A}_r$ (reconstructed by the configuration-based scheme, see Equation (6)) as a low-rank approximation of the original adjacency matrix $\mathbf{A}$, the kernel matrix of $\mathcal{G}$ can be approximated by the one on $\mathcal{G}_s$ as follows:*

$$\begin{aligned}\mathcal{K}(\mathcal{G}) &\approx \left(\mathbf{D}^{-1}\mathbf{A}_r\right)^\tau\mathbf{D}^{-1}\\ &= \mathbf{R}\left(\mathbf{D}_s^{-1}\mathbf{A}_s\right)^\tau\mathbf{D}_s^{-1}\mathbf{R}^\top\\ &= \mathbf{R}\,\mathcal{K}(\mathcal{G}_s)\,\mathbf{R}^\top,\end{aligned} \tag{11}$$

where $\mathbf{R} \in \mathbb{R}^{n\times n_s}$ is the restoration matrix:

$$\mathbf{R}(i,p) = \begin{cases}1 & \text{if } v_i \in \mathcal{S}_k,\\ 0 & \text{otherwise,}\end{cases} \tag{12}$$

Proof. See Appendix. □

Corollary 1. *Let τ takes values in $\{1, 2, \ldots, T\}$ and sum them together, we have*

$$\sum_{\tau=1}^{T}(\mathbf{D}^{-1}\mathbf{A})^\tau\mathbf{D}^{-1} \approx \sum_{\tau=1}^{T}(\mathbf{D}^{-1}\mathbf{A}_r)^\tau\mathbf{D}^{-1} = \mathbf{R}\left(\sum_{\tau=1}^{T}(\mathbf{D}_s^{-1}\mathbf{A}_s)^\tau\mathbf{D}_s^{-1}\right)\mathbf{R}^\top, \tag{13}$$

where $\mathbf{R}$ is defined in Equation (12).

4.3 Approximating Node Embeddings

Based on Theorem 1, we now discuss how to approximate node embeddings for the original nodes. Since DeepWalk and LINE can be viewed as special cases of NetMF, we focus on NetMF in the following discussion.

Theorem 2. *Embeddings learned by NetMF on the original graph $\mathcal{G}$, $\mathbf{E}$, can be approximated by embeddings learned by NetMF on the summary graph $\mathcal{G}_s$, $\mathbf{E}_s$, using the restoration matrix $\mathbf{R}$ in Equation (12), i.e.,*

$$\mathbf{E} \approx \mathbf{R}\,\mathbf{E}_s. \tag{14}$$

Proof. Consider $\mathbf{A}_r$ as a low-rank approximation of $\mathbf{A}$, and replace $\mathbf{A}$ by $\mathbf{A}_r$ in the NetMF matrix. According to Corollary 1:

$$\begin{aligned}
\mathbf{M} &= \log\left(\frac{\mathrm{vol}(\mathcal{G})}{bT}\sum_{\tau=1}^{T}(\mathbf{D}^{-1}\mathbf{A})^{\tau}\mathbf{D}^{-1}\right)\\
&\approx \log\left(\frac{\mathrm{vol}(\mathcal{G})}{bT}\sum_{\tau=1}^{T}(\mathbf{D}^{-1}\mathbf{A}_r)^{\tau}\mathbf{D}^{-1}\right)\\
&= \log\left(\frac{\mathrm{vol}(\mathcal{G})}{bT}\mathbf{R}\left(\sum_{\tau=1}^{T}(\mathbf{D}_s^{-1}\mathbf{A}_s)^{\tau}\mathbf{D}_s^{-1}\right)\mathbf{R}^{\top}\right)\\
&\overset{1}{=} \mathbf{R}\cdot\log\left(\frac{\mathrm{vol}(\mathcal{G})}{bT}\left(\sum_{\tau=1}^{T}(\mathbf{D}_s^{-1}\mathbf{A}_s)^{\tau}\mathbf{D}_s^{-1}\right)\right)\cdot\mathbf{R}^{\top}\\
&= \mathbf{R}\,\mathbf{M}_s\,\mathbf{R}^{\top}.
\end{aligned}$$

Here $\mathbf{M}_s$ is the corresponding matrix DeepWalk factorizing on summary graph $\mathcal{G}_s$.

Suppose $\mathbf{M}_s$ is factorized into $\mathbf{M}_s = \mathbf{X}_s\mathbf{Y}_s^{\top}$, then $\mathbf{M} \approx (\mathbf{R}\mathbf{X}_s)(\mathbf{R}\mathbf{Y}_s)^{\top}$. That is, embeddings of original graph $\mathcal{G}$ can be approximated by embeddings learned on summary graph $\mathcal{G}_s$ with a restoration matrix $\mathbf{R}$:

$$\mathbf{E} \approx \mathbf{R}\cdot\mathbf{E}_s. \tag{15}$$

□

According to Theorem 2 and the definition of $\mathbf{R}$ matrix ($\mathbf{R}(i,p) = 1$ if $v_i \in \mathcal{S}_p$), we can conclude that nodes in the same supernode get the same embeddings after restoration. This approach, is exactly the way how related works (including HARP, MILE, and GraphZoom) restore the embeddings. Thus, Theorem 2 provides a **theoretical interpretation** for the restoration step of existing methods.

5 PROPOSED METHODS

In this section, we first reveal that the error of kernel matrix is closely related to the error of the normalized adjacency matrix. Then, by showing that the latter error is bounded by a trace maximization objective function, we propose a summarization method HCSumm based on spectral clustering.

5.1 Kernel Matrix Error Analysis

From the previous section, it is known that the kernel matrix is closely related to many graph properties and graph mining tasks. Hence, it is important to preserve the kernel matrix of the original graph. One may ask that how much the error of kernel matrix is introduced by replacing $\mathbf{A}$ by $\mathbf{A}_r$? The following theorem gives a brief analysis.

Theorem 3. *By replacing $\mathbf{A}$ by $\mathbf{A}_r$, the error of kernel matrix is bounded by*

$$\|\mathcal{K}_\tau(\mathcal{G}) - \mathcal{K}_\tau(\mathcal{G}_s)\|_{\mathrm{F}} \le C\cdot\tau\cdot\|\mathbf{D}^{-\frac{1}{2}}\mathbf{A}\mathbf{D}^{-\frac{1}{2}} - \mathbf{D}^{-\frac{1}{2}}\mathbf{A}_r\mathbf{D}^{-\frac{1}{2}}\|_{\mathrm{F}}, \tag{16}$$

where $C = \|\mathbf{D}^{-\frac{1}{2}}\|_2^2$ is a constant only depending on the input graph.

[1]This equation holds, since each row of $\mathbf{R}$ contains exactly one non-zero value "1." Thus, we can take it out of the log function.

Proof. Note that the kernel matrix $\mathcal{K}_\tau(\mathcal{G})$ can be rewritten as

$$\mathcal{K}_\tau(\mathcal{G}) = \mathbf{D}^{-\frac{1}{2}}(\mathbf{D}^{-\frac{1}{2}}\mathbf{A}\mathbf{D}^{-\frac{1}{2}})^\tau\mathbf{D}^{-\frac{1}{2}}.$$

Then,

$$\begin{aligned}
&\|\mathcal{K}_\tau(\mathcal{G}) - \mathcal{K}_\tau(\mathcal{G}_r)\|_{\mathrm{F}} \\
&= \left\|\mathbf{D}^{-\frac{1}{2}}\left((\mathbf{D}^{-\frac{1}{2}}\mathbf{A}\mathbf{D}^{-\frac{1}{2}})^\tau - (\mathbf{D}^{-\frac{1}{2}}\mathbf{A}_r\mathbf{D}^{-\frac{1}{2}})^\tau\right)\mathbf{D}^{-\frac{1}{2}}\right\|_{\mathrm{F}} \\
&= \left\|\mathbf{D}^{-\frac{1}{2}}\right\|_2^2 \cdot \left\|(\mathbf{D}^{-\frac{1}{2}}\mathbf{A}\mathbf{D}^{-\frac{1}{2}})^\tau - (\mathbf{D}^{-\frac{1}{2}}\mathbf{A}_r\mathbf{D}^{-\frac{1}{2}})^\tau\right\|_{\mathrm{F}} \\
&= C \cdot \left\|(\mathbf{D}^{-\frac{1}{2}}\mathbf{A}\mathbf{D}^{-\frac{1}{2}})^\tau - (\mathbf{D}^{-\frac{1}{2}}\mathbf{A}_r\mathbf{D}^{-\frac{1}{2}})^\tau\right\|_{\mathrm{F}},
\end{aligned}$$

where $C = \|\mathbf{D}^{-\frac{1}{2}}\|_2^2 = d_{\min}^{-1}$ is a constant only depending on the input graph.

Denote $\mathcal{A} = \mathbf{D}^{-\frac{1}{2}}\mathbf{A}\mathbf{D}^{-\frac{1}{2}}$ and $\mathcal{A}_r = \mathbf{D}^{-\frac{1}{2}}\mathbf{A}_r\mathbf{D}^{-\frac{1}{2}}$ for notation simplicity, we have

$$\mathcal{A}^\tau - \mathcal{A}_r^\tau = (\mathcal{A}^{\tau-1} - \mathcal{A}_r^{\tau-1})\mathcal{A} + \mathcal{A}_r^{\tau-1}(\mathcal{A} - \mathcal{A}_r).$$

And

$$\begin{aligned}
\|\mathcal{A}^\tau - \mathcal{A}_r^\tau\|_{\mathrm{F}} &\le \|\mathcal{A}(\mathcal{A}^{\tau-1} - \mathcal{A}_r^{\tau-1})\|_{\mathrm{F}} + \|\mathcal{A}_r^{\tau-1}(\mathcal{A} - \mathcal{A}_r)\|_{\mathrm{F}} \\
&\le \|\mathcal{A}\|_2\|\mathcal{A}^{\tau-1} - \mathcal{A}_r^{\tau-1}\|_{\mathrm{F}} + \|\mathcal{A}_r\|_2^{\tau-1}\|\mathcal{A} - \mathcal{A}_r\|_{\mathrm{F}}
\end{aligned}$$

($\|\mathcal{A}\|_2 \le 1$ and $\|\mathcal{A}_r\|_2 \le 1$)

$$\le \|\mathcal{A}^{\tau-1} - \mathcal{A}_r^{\tau-1}\|_{\mathrm{F}} + \|\mathcal{A} - \mathcal{A}_r\|_{\mathrm{F}}.$$

Applying it recursively, we have

$$\|\mathcal{A}^\tau - \mathcal{A}_r^\tau\|_{\mathrm{F}} \le \tau\|\mathcal{A} - \mathcal{A}_r\|_{\mathrm{F}}.$$

Thus,

$$\|\mathcal{K}_\tau(\mathcal{G}) - \mathcal{K}_\tau(\mathcal{G}_r)\|_{\mathrm{F}} \le C \cdot \tau \cdot \left\|\mathbf{D}^{-\frac{1}{2}}\mathbf{A}\mathbf{D}^{-\frac{1}{2}} - \mathbf{D}^{-\frac{1}{2}}\mathbf{A}_r\mathbf{D}^{-\frac{1}{2}}\right\|_{\mathrm{F}}, \tag{17}$$

where $C = \|\mathbf{D}^{\frac{1}{2}-c}\|_2^2$ is a constant only depending on the input graph. □

5.2 HCSumm

Theorem 3 states that the error of τ-order kernel matrix is bounded by τ times the error of $\mathcal{A}$. Hence, we aim to design algorithms minimizing the error of $\mathcal{A} - \mathcal{A}_r$ to preserve the kernel matrix.

Lemma 1. *Let $\mathcal{A}_r$ be the adjacency matrix of $\mathcal{G}_r$. Then, $\mathcal{A}_r$ can be written as*

$$\mathcal{A}_r = \Pi \cdot \mathcal{A} \cdot \Pi, \tag{18}$$

where $\Pi = \mathbf{Y}\mathbf{Y}^\top$ is a projection matrix on the column space of $\mathbf{D}^{\frac{1}{2}}\mathbf{P}^\top$ and $\mathbf{Y} = \mathbf{D}^{\frac{1}{2}}\mathbf{P}^\top(\mathbf{P}\mathbf{D}\mathbf{P})^{-\frac{1}{2}}$:

$$\mathbf{Y}(i,k) = \begin{cases} \frac{\sqrt{d_i}}{\sqrt{D_k}} & \text{if } i \in \mathcal{S}_k, \\ 0 & \text{otherwise,} \end{cases} \qquad \Pi(i,j) = \begin{cases} \frac{\sqrt{d_i d_j}}{D_k} & \text{if } i,j \in \mathcal{S}_k, \\ 0 & \text{otherwise.} \end{cases} \tag{19}$$

Proof. Let $\mathcal{A}_r$ be the normalized adjacency matrix of $\mathcal{G}_r$. Then,

$$\mathcal{A}_r(i,j) = \frac{1}{\sqrt{d_i}}\mathbf{A}_r(i,j)\frac{1}{\sqrt{d_j}} = \frac{1}{\sqrt{d_i}}\frac{d_i}{D_k}\mathbf{A}_s(k,l)\frac{d_j}{D_l}\frac{1}{\sqrt{d_j}} = \frac{\sqrt{d_i}}{D_k}\mathbf{A}_s(k,l)\frac{\sqrt{d_j}}{D_l}.$$

And given the definition of Π in Equation (19), we have

$$\begin{aligned}(\Pi\mathcal{A}\Pi)(i,j) &= \sum_{a\in\mathcal{S}_k, b\in\mathcal{S}_l} \Pi(i,a)\frac{\mathbf{A}(a,b)}{\sqrt{d_i d_j}}\Pi(b,j)\\ &= \sum_{a\in\mathcal{S}_k, b\in\mathcal{S}_l} \frac{\sqrt{d_i d_a}}{D_k}\frac{\mathbf{A}(a,b)}{\sqrt{d_a d_b}}\frac{\sqrt{d_b d_j}}{D_l}\\ &= \sum_{a\in\mathcal{S}_k, b\in\mathcal{S}_l} \frac{\sqrt{d_i}}{D_k}\mathbf{A}(a,b)\frac{\sqrt{d_j}}{D_l}\\ &= \frac{\sqrt{d_i}}{D_k}\mathbf{A}_s(k,l)\frac{\sqrt{d_j}}{D_l} = \mathcal{A}_r(i,j).\end{aligned}$$

□

From the above lemma, the error of the normalized adjacency matrix can be formulated as $\|\mathcal{A} - \Pi\mathcal{A}\Pi\|_{\mathrm{F}}$, which is further bounded by

$$\begin{aligned}\|\mathcal{A} - \Pi\mathcal{A}\Pi\|_{\mathrm{F}} &= \|\mathcal{A} - \Pi\mathcal{A} + \Pi\mathcal{A} - \Pi\mathcal{A}\Pi\|_{\mathrm{F}}\\ &\le \|\mathcal{A} - \Pi\mathcal{A}\|_{\mathrm{F}} + \|\Pi\mathcal{A} - \Pi\mathcal{A}\Pi\|_{\mathrm{F}} \quad (\Pi \text{ is a projection matrix and hence } \|\Pi\|_2 = 1)\\ &\le \|\mathcal{A} - \Pi\mathcal{A}\|_{\mathrm{F}} + \|\mathcal{A} - \mathcal{A}\Pi\|_{\mathrm{F}}\\ &= \|\mathcal{A} - \Pi\mathcal{A}\|_{\mathrm{F}} + \|\Pi\mathcal{A} - \mathcal{A}\|_{\mathrm{F}}\\ &= 2\|\mathcal{A} - \Pi\mathcal{A}\|_{\mathrm{F}}.\end{aligned}$$

Although there is a factor of 2, we find that these two terms are very close in practice. Hence, it is a good choice to use $\|\mathcal{A} - \Pi\mathcal{A}\|$ as an approximation of $\|\mathcal{A} - \Pi\mathcal{A}\Pi\|_{\mathrm{F}}$.

$\|\mathcal{A} - \Pi\mathcal{A}\|$ is easier to analyze and equivalent to a trace optimization problem:

$$\begin{aligned}\|\mathcal{A} - \Pi\mathcal{A}\|_{\mathrm{F}}^2 &= \mathrm{tr}((\mathcal{A} - \Pi\mathcal{A})(\mathcal{A} - \Pi\mathcal{A})^\top)\\ &= \mathrm{tr}(\mathcal{A}\mathcal{A} - \mathcal{A}\Pi\mathcal{A} - \Pi\mathcal{A}\mathcal{A} + \Pi\mathcal{A}\mathcal{A}\Pi)\\ &= \mathrm{tr}(\mathcal{A}^2) - \mathrm{tr}(\Pi\mathcal{A}^2)\\ &= \mathrm{tr}(\mathcal{A}^2) - \mathrm{tr}(\mathbf{Y}^\top\mathcal{A}^2\mathbf{Y}).\end{aligned} \tag{20}$$

Since $\mathrm{tr}(\mathcal{A}^2)$ is a constant, minimizing $\|\mathcal{A} - \Pi\mathcal{A}\|_{\mathrm{F}}^2$ is equivalent to maximizing $\mathrm{tr}(\mathbf{Y}^\top\mathcal{A}^2\mathbf{Y})$ where $\mathbf{Y}^\top\mathbf{Y} = \mathbf{I}$, which is a trace maximization problem. If we relax the constraint that the $\mathbf{Y}$ is a discrete solution obtained from a summary graph, then this trace maximization problem can be easily solved by calculating the first k large eigenvectors of $\mathcal{A}^2$ using the Rayleigh-Ritz theorem. Since $\mathcal{A}^2$ and $\mathcal{A}$ share the eigenvectors, we can use the first k singular vectors of $\mathcal{A}$ instead to avoid calculating $\mathcal{A}^2$.

To obtain the discrete solution from the continuous solution, the typical way is using the k-means algorithm to partition the rows of $\mathbf{Y}$ into k clusters. However, the cluster number in k-means is relatively small compared to the summary graph size in graph summarization problem, which makes k-means insufficient in our scenario. Thus, we use hierarchical clustering with ward linkage (also known as Ward's method) instead. Ward's method is a hierarchical clustering algorithm sharing the same objective function with k-means but working in a bottom-up approach. It starts with each data point as a cluster and iteratively merge the cluster pair raising the minimal cost increment.

Based on the above analysis, we propose a graph summarization algorithm HCSumm using hierarchical clustering, described in Algorithm 1. First, it computes the first d singular vectors of $\mathcal{A}$. To enhance the efficiency, we use randomized SVD [10] instead of eigen-decomposition to

ALGORITHM 1: HCSumm

Input: Graph $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \mathbf{A})$, Summary graph size k, Singular vector number d
Output: Summary graph $\mathcal{G}_s$

$\mathcal{A} \leftarrow$ normalized adjacency matrix
$Z \leftarrow$ randomizedSVD$(\mathcal{A}, d)$
$P \leftarrow$ partition the rows of Z into k clusters using Ward's method
$\mathcal{G}_s \leftarrow$ construct the summary graph using P
return Return $\mathcal{G}_s$

ALGORITHM 2: HCSumm-Large

Input: Graph $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \mathbf{A})$, Summary graph size k, Singular vector number d
Output: Summary graph $\mathcal{G}_s$

$\mathcal{A} \leftarrow$ normalized adjacency matrix
$Z \leftarrow$ randomizedSVD$(\mathcal{A}, d)$
$n \leftarrow |\mathcal{V}|$
while $n > k$ **do**
    $x \leftarrow \arg\min_v \deg(v)$
    $y \leftarrow \arg\min_v \|Z[x] - Z[v]\|_2$
    Merge x and y
    $n \leftarrow n - 1$
end while
return Return $\mathcal{G}_s$

calculate the singular vector of $\mathcal{A}$. Then, it clusters the rows of Z into k clusters using Ward's method. Finally, the summary graph is constructed according to the clustering result partition P and returned.

Algorithm 1 still bears the efficiency problem facing large input graphs, since Ward's method needs to keep track of all the pairwise distances between clusters. Thus, we propose HCSumm-Large (Algorithm2) for large-scale graphs using a degree heuristic. In each step, it chooses a node x with the minimum degree and find another node y nearest to x. To find the closest node to x, we use faiss [13] library and build a simple IVF index on Z. Then, it merges the two nodes together and repeats the process until all the nodes are merged into k supernodes.

6 EXPERIMENTS

In this section, we design experiments to answer the following research questions:

- **Summary Quality:** How well does HCSumm preserve the normalized adjacency matrix of input graphs?
- **Node Embedding Preservation:** How well does HCSumm preserve the node embeddings of input graphs?
- **Scalability:** How does HCSumm scale with the input graph size?

6.1 Experimental Setup

Datasets. We evaluate HCSumm on four real-world social network datasets frequently used in node embedding learning. The statistics of these datasets are shown in Table 2. Cora dataset is a citation network of machine learning papers and labels are the research areas of papers. BlogCatalog dataset is a social network of bloggers at BlogCatalog website and labels are the interests of

Table 2. Dataset Statistics

Dataset	#Nodes	#Edges	#Labels
Cora	2,307	5,278	7
BlogCatalog	10,312	667,966	39
Flickr	89,250	5,899,882	195
YouTube	1,138,499	2,990,443	47

bloggers. Flickr dataset is a user social network on Flickr website and labels are the user interest groups. YouTube dataset is a network of users on YouTube website and labels are the user interest tags.

Baselines. We compare our HCSumm with two baselines, GraphZoom and SpecSumm. GraphZoom[2] is the state-of-the-art graph summarization method for learning node embeddings and show significant better performance than earlier methods such as HARP and MILE. SpecSumm[3] shares the similar approach with HCSumm, but aims at minimize the reconstruction error of the adjacency matrix. It computes the first d eigenvectors of the adjacency matrix and uses mini-batch k-means to obtain the summary graph.

Summary sizes. We summarize input graphs with different summary sizes and evaluate the quality of them. To make a fair comparison, we should evaluate the summary quality of different methods with the same summary sizes. For our method HCSumm and SpecSumm, the summary size is a parameter that can be set by users. For GraphZoom, it is a multi-level summarization method and produces summary graphs with different sizes in each level. Summary size of each level is fixed and users can only set the number of level but not the summary size. For more details, please refer to the original paper [5]. Thus, to make a fair comparison, we set the summary sizes in HCSumm and SpecSumm to the same values as GraphZoom's summary sizes in different levels.

Implementation details. We implement HCSumm in Python3. For SpecSumm and GraphZoom, we use the source code released by the authors. All experiments are performed on a machine with a 2.4 GHz Intel Xeon E5-2640 CPU and 128 GB memory. GraphZoom has a variant version utilizing node features. We use the feature-fusion version of GraphZoom on Flickr, since it has node features and use the vanilla version on other datasets without node features. For our method, we run the vanilla HCSumm (Algorithm 1) on the BlogCatalog dataset and HCSumm-Large (Algorithm 2) on the other two datasets.

6.2 Summary Quality

We first evaluate the summary graph quality of different methods. Two metrics, $\|\mathcal{A} - \Pi\mathcal{A}\Pi\|_F$ and $\|\mathcal{A} - \Pi\mathcal{A}\|_F$ are applied to measure the quality. The former is the Frobenius norm of the difference between the original normalized adjacency matrix and the reconstructed one, and the latter is the objective function in the trace optimization problem (see Equation (20)). Due to the memory limit, we only evaluate on BlogCatalog and Flickr datasets.

Experimental Results. The results are listed in Table 4. From the table, we notice that the error_1 and error_2 terms, i.e., $\|\mathcal{A} - \Pi\mathcal{A}\Pi\|_F$ and $\|\mathcal{A} - \Pi\mathcal{A}\|_F$, are very close and the ratio of them is far from the theoretical bound of 2. Thus, it is reasonable to use $\|\mathcal{A} - \Pi\mathcal{A}\|_F$ as a surrogate of $\|\mathcal{A} - \Pi\mathcal{A}\Pi\|_F$. For BlogCatalog and Cora dataset, our method achieves the smallest error measures. For Flickr dataset, HCSumm always outperforms SpecSumm. Compared to GraphZoom, HCSumm

[2] https://github.com/cornell-zhang/GraphZoom
[3] https://version.helsinki.fi/ads/specsumm

Table 4. Error Measures of Summary Graphs

Size	Method	$error_1$	$error_2$
1170 (r = 43.21%)	HCSumm	**15.93**	**17.37**
	GraphZoom	18.35	20.56
	SpecSumm	20.70	23.39
523 (r = 15.62%)	HCSumm	**20.56**	**21.38**
	GraphZoom	22.78	23.88
	SpecSumm	24.17	25.81
230 (r = 9.97%)	HCSumm	**24.29**	**25.10**
	GraphZoom	24.31	25.17
	SpecSumm	25.91	26.62

(a) Cora

Size	Method	$error_1$	$error_2$
6997 (r = 67.85%)	HCSumm	**4.60**	**5.96**
	GraphZoom	5.30	6.81
	SpecSumm	5.39	6.95
4539 (r = 44.01%)	HCSumm	**6.30**	**7.72**
	GraphZoom	7.01	8.29
	SpecSumm	6.78	8.23
2891 (r = 28.03%)	HCSumm	**7.33**	**8.54**
	GraphZoom	8.04	8.93
	SpecSumm	7.74	8.88

(b) BlogCatalog

Size	Method	$error_1$	$error_2$
22525 (r = 25.24%)	HCSumm	26.86	29.14
	GraphZoom	**24.37**	**27.83**
	SpecSumm	26.48	29.19
7482 (r = 8.38%)	HCSumm	28.94	29.84
	GraphZoom	**28.24**	**29.62**
	SpecSumm	28.96	30.01
2954 (r = 3.31%)	HCSumm	**29.42**	**29.91**
	GraphZoom	29.43	30.01
	SpecSumm	29.68	30.13

(c) Flickr

r stands for the summary ratio. $error_1$ and $error_2$ refer to $\|\mathcal{A} - \Pi\mathcal{A}\|_F$ and $\|\mathcal{A} - \Pi\mathcal{A}\Pi\|_F$, respectively.

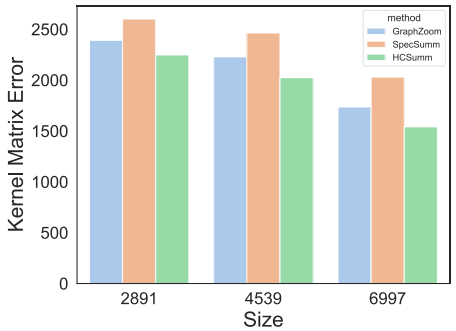


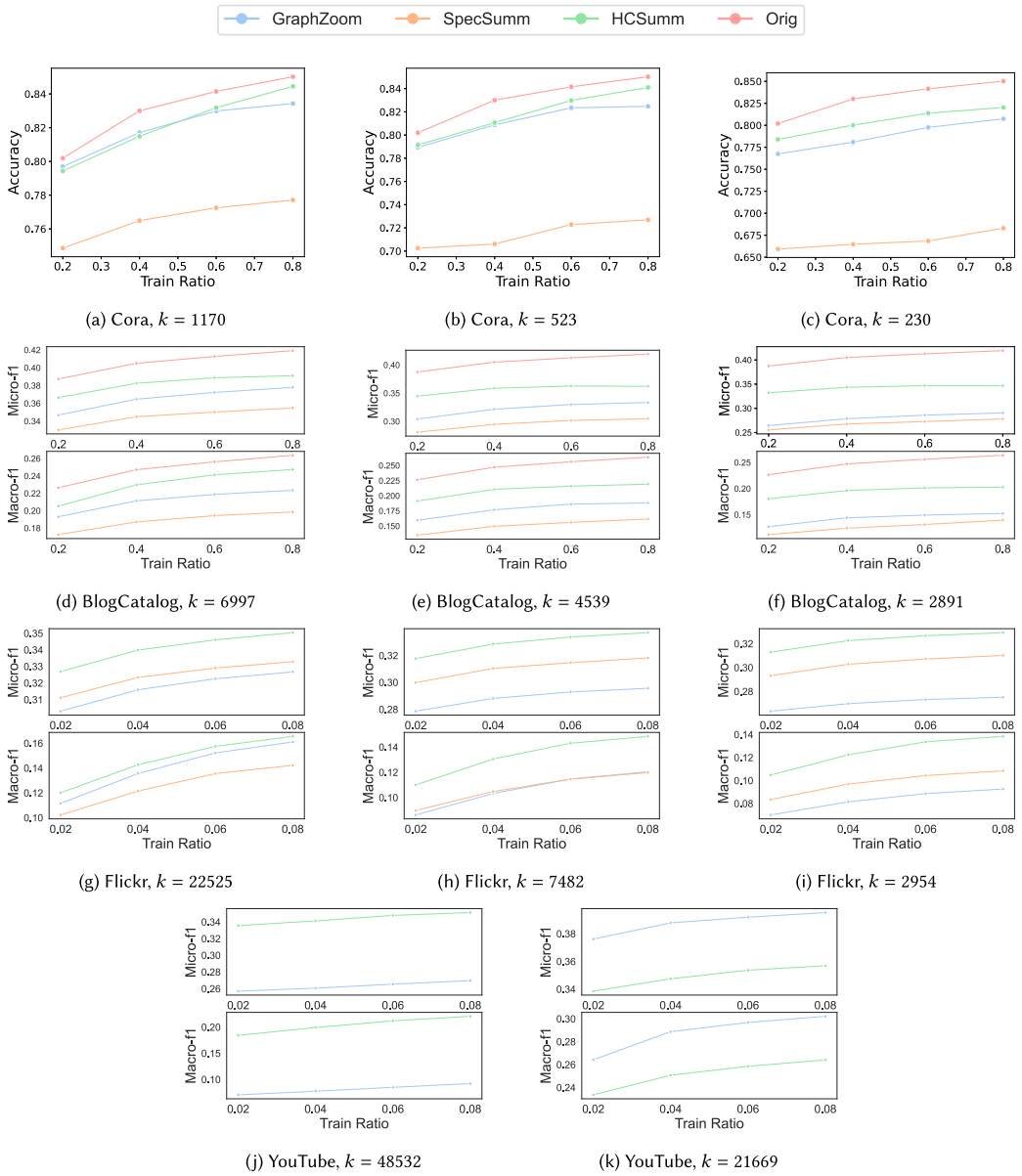
Fig. 2. Kernel matrix error of BlogCatalog datasets. The x-axis is the summary graph size. Our method achieves the smallest kernel matrix error.

does not achieve the smallest error when the summary size is 22,525. As the summary size goes smaller, the errors of HCSumm gradually approaches that of GraphZoom and outperforms it when the summary size is 2,954.

Kernel Matrix Error. We also calculate the kernel matrix (see Equation (9)) error of different summarization ratios. The kernel matrix error is defined as the Frobenius norm of the difference between the original kernel matrix and the kernel matrix of the summary graph. Since the node embeddings are directly from the kernel matrix, the kernel matrix error can reflect how well the node embeddings are preserved by different methods. Due to the memory limit (the kernel matrix is dense), we only calculate the kernel matrix error on BlogCatalog dataset. The results are shown in Figure 2. From the results, we can see that HCSumm achieves the smallest kernel matrix error and thus preserves the node embeddings best. This result is consistent with the node classification performance in the next section.

6.3 Node Embedding Preservation

In this experiment, we aim to evaluate how well HCSumm preserves the node embeddings. We evaluate it by downstream node classification tasks. We run NetMF and DeepWalk on the summary graphs and restore the embeddings of the original nodes (Equation (15)). Then, we use the restored embeddings to train a Logistic Regression classifier and evaluate the performance. We set the training ratios to $\{0.20, 0.40, 0.60, 0.80\}$ on BlogCatalog and Cora dataset and $\{0.02, 0.04, 0.06, 0.08\}$ on Flickr and YouTube dataset and report the mean accuracy (for Cora dataset) and the micro f1 scores and macro f1 scores (for other three datasets) of five average runs. The dimension of the embedding is set to 128 in all experiments. We do not run SpecSumm on YouTube dataset due to its long run time on such a large dataset.



(a) Cora, $k = 1170$ (b) Cora, $k = 523$ (c) Cora, $k = 230$

(d) BlogCatalog, $k = 6997$ (e) BlogCatalog, $k = 4539$ (f) BlogCatalog, $k = 2891$

(g) Flickr, $k = 22525$ (h) Flickr, $k = 7482$ (i) Flickr, $k = 2954$

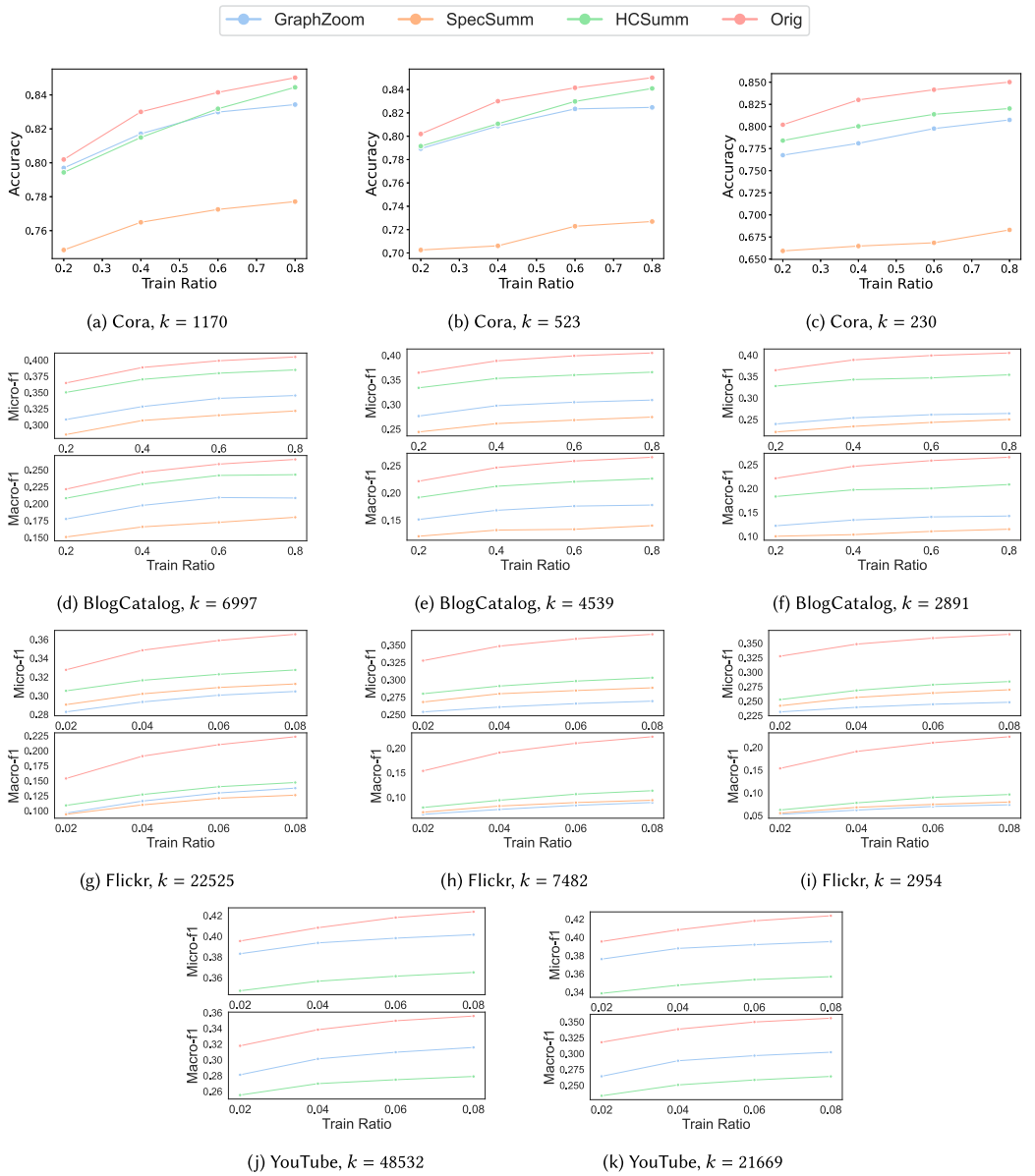
(j) YouTube, $k = 48532$ (k) YouTube, $k = 21669$

Fig. 3. NetMF performance on the node classification task.

6.3.1 NetMF. **Parameter Settings.** We set $T = 10$ in NetMF. For graphs larger than 20,000 nodes, we use the variant NetMF-large [33] instead of the original NetMF.

Experimental Results. Mean micro f1 scores and macro f1 scores of five average runs are shown in Figure 3.[4] It can be seen that both the micro f1 scores and macro f1 scores drop after summarization. Compared to baselines, our HCSumm method achieves the slightest drop on Cora, BlogCatalog and Flickr dataset. On YouTube dataset, despite that GraphZoom outperforms

[4]NetMF cannot run on the YouTube dataset (exceeds memory limit), hence some results are missing in the figure.



(a) Cora, $k = 1170$ (b) Cora, $k = 523$ (c) Cora, $k = 230$

(d) BlogCatalog, $k = 6997$ (e) BlogCatalog, $k = 4539$ (f) BlogCatalog, $k = 2891$

(g) Flickr, $k = 22525$ (h) Flickr, $k = 7482$ (i) Flickr, $k = 2954$

(j) YouTube, $k = 48532$ (k) YouTube, $k = 21669$

Fig. 4. DeepWalk performance on the node classification task.

HCSumm when the summary size is GraphZoom shows an unstable performance on different summary sizes. For example, the micro f1 score drops under 0.28 when the summary size is 48,532 and goes up to 0.34 when the summary size is 21,669. On the contrary, HCSumm method achieves a stable and relative good performance on all summary sizes. Overall, our HCSumm method can preserve the node embedding information better than baselines and are consistent with the results in the previous section.

6.3.2 DeepWalk. **Parameter Settings.** The window size and the negative samples b are set to 10 and 1, respectively. The number of walks per node is set to 10 and the length of each walk is 80.

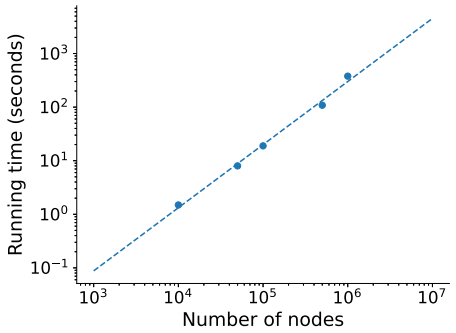


Fig. 5. Scalability of HCSumm method.

Experimental Results. Similar to NetMF, we report the mean micro f1 scores and macro f1 scores of five average runs in Figure 4. HCSumm outperforms other baselines on Cora, BlogCatalog, and Flickr dataset and is slightly worse than GraphZoom only on YouTube dataset. In general, HCSumm preserves the node embeddings better than baselines and is consistent with the results in the previous section.

6.4 Scalability

In this experiment, we evaluate the efficiency and scalability of our HCSumm method. We sample graphs with different sizes ranging from 1,000 to 1 million the largest YouTube dataset and record the running time of HCSumm method on these graphs. We represent the average running time of 5 runs in Figure 5. It can be seen that the running time of HCSumm method is linear with the graph size.

7 CONCLUSION

In this work, we study the connection between graph summarization and node embedding learning. We reveal that three matrix-factorization-based node embeddings (DeepWalk, LINE, and NetMF) of the original graph and the summary graph are closely related via a configuration-based reconstructed graph. We analyze the upper bound of node embedding error and propose HCSumm to summarize input graphs while preserving node embeddings. Extensive experiments on real-world datasets show that HCSumm preserves the node embedding better than baselines. Overall, our study helps understand the existing works of learning node embeddings via graph summarization and provides theoretical insights for future works on this problem.

APPENDIX

A PROOFS

A.1 Proof of Theorem 1

Before we prove Theorem 1, we first introduce Lemmas 2 and 3.

Lemma 2.

$$\mathbf{Q}^\top \mathbf{D}^{-1} \mathbf{Q} = \mathbf{D}_s^{-1},$$

where $\mathbf{Q}$ *is the reconstruction matrix (Equation* (7)*),* $\mathbf{D}$ *and* $\mathbf{D}_s$ *are degree matrix of the original graph and the summary graph.*

Proof. The (p, q)th entry in $\mathbf{Q}^\top \mathbf{D}^{-1} \mathbf{Q}$ is

$$\mathbf{Q}^\top \mathbf{D}^{-1} \mathbf{Q}(p, q) = \sum_i \mathbf{Q}(i, p) \frac{1}{d_i} \mathbf{Q}(i, q).$$

It is easy to see that the result is not zero only when $p = q$ (since a node v_i cannot belong to two supernodes $\mathcal{S}_p$ and $\mathcal{S}_q$ simultaneously). And diagonal items are (note that $d_p^{(s)} = \sum_{v_i \in \mathcal{S}_p} d_i$):

$$\begin{aligned}\mathbf{Q}^\top \mathbf{D}^{-1} \mathbf{Q}(p,p) &= \sum_{v_i \in \mathcal{S}_p} \mathbf{Q}(i,p) \frac{1}{d_i} \mathbf{Q}(i,p) = \sum_{v_i \in \mathcal{S}_p} \frac{d_i}{D_p} \frac{1}{d_i} \frac{d_i}{D_p} \\ &= \sum_{v_i \in \mathcal{S}_p} \frac{d_i}{D_p} \frac{1}{D_p} = \frac{1}{D_p} = \mathbf{D}_s^{-1}(p,p).\end{aligned}$$

□

Lemma 3.

$$\mathbf{R}\mathbf{D}_s^{-1} = \mathbf{D}^{-1}\mathbf{Q}. \tag{21}$$

Proof. Suppose $v_i \in \mathcal{S}_k$, then the (i,k)-th entry of $\mathbf{R}\mathbf{D}_s^{-1}$ is

$$\mathbf{R}\mathbf{D}_s^{-1}(i,k) = 1 \cdot (D_k)^{-1} = \frac{1}{D_k}.$$

And the (i,k)th entry of $\mathbf{D}^{-c}\mathbf{Q}$ is

$$\mathbf{D}^{-1}\mathbf{Q}(i,k) = \frac{1}{d_i} \frac{d_i}{D_k} = \frac{1}{D_k}.$$

Thus, $\mathbf{R}\mathbf{D}_s^{-1} = \mathbf{D}^{-1}\mathbf{Q}$. □

Now, we prove Theorem 1. Denote $\mathcal{K}_\tau(\mathcal{G}) = \left(\mathbf{D}^{-1}\mathbf{A}_r\right)^\tau \mathbf{D}^{-1}$ for convenience.
Prove by induction. When $\tau = 1$,

$$\begin{aligned}\mathcal{K}_1(\mathcal{G}_r) &= \mathbf{D}^{-1}\mathbf{A}_r\mathbf{D}^{-1} = \mathbf{D}^{-1}\mathbf{Q}\mathbf{A}_s\mathbf{Q}^\top\mathbf{D}^{-1} \\ &= \mathbf{R}\mathbf{D}_s^{-1}\mathbf{A}_s\mathbf{D}_s^{-1}\mathbf{R}^\top \qquad \text{(Lemma 3)} \\ &= \mathbf{R}\,\mathcal{K}_1(\mathcal{G}_s)\,\mathbf{R}^\top.\end{aligned}$$

Suppose the lemma holds for $\tau = i$, i.e., $\mathcal{K}_i(\mathcal{G}_r) = \mathbf{R}\,\mathcal{K}_i(\mathcal{G}_s)\,\mathbf{R}^\top$. For the case $\tau = i+1$,

$$\begin{aligned}\mathcal{K}_{i+1}(\mathcal{G}_r) &= \mathbf{D}^{-1}\mathbf{A}_r\mathcal{K}_i(\mathcal{G}_r) = \mathbf{D}^{-1}\mathbf{A}_r\mathbf{R}\,\mathcal{K}_i(\mathcal{G}_s)\,\mathbf{R}^\top \\ &= \mathbf{D}^{-1}\mathbf{Q}\mathbf{A}_s\mathbf{Q}^\top\mathbf{R}\,\mathcal{K}_i(\mathcal{G}_s)\,\mathbf{R}^\top \\ &\overset{5}{=} \mathbf{D}^{-1}\mathbf{Q}\mathbf{A}_s(\mathbf{Q}^\top\mathbf{D}^{-1}\mathbf{Q})\mathbf{D}_s\,\mathcal{K}_i(\mathcal{G}_s)\,\mathbf{R}^\top \\ &\text{(Lemmas 2 and 3)} \\ &= \mathbf{R}\mathbf{D}_s^{-1}\mathbf{A}_s\,\mathcal{K}_i(\mathcal{G}_s)\,\mathbf{R}^\top \\ &= \mathbf{R}\,\mathcal{K}_{i+1}(\mathcal{G}_s)\,\mathbf{R}^\top.\end{aligned}$$

Applying principal of induction finishes the proof.

REFERENCES

[1] Bijaya Adhikari, Yao Zhang, Aditya Bharadwaj, and B. Aditya Prakash. 2017. Condensing temporal networks using propagation. In *Proceedings of the ICDM*.

[2] Maham Anwar Beg, Muhammad Ahmad, Arif Zaman, and Imdadullah Khan. 2018. Scalable approximation algorithm for graph summarization. In *Proceedings of the PAKDD*.

[3] Haochen Chen, Bryan Perozzi, Yifan Hu, and Steven Skiena. 2018. HARP: Hierarchical representation learning for networks. In *Proceedings of the AAAI*.

[4] Graham Cormode and Shan Muthukrishnan. 2005. An improved data stream summary: The count-min sketch and its applications. *J. Algor.* 55, 1 (2005), 58–75.

[5] $\mathbf{D}^{-c}\mathbf{Q} = \mathbf{R}\mathbf{D}_s^{-c} \Rightarrow \mathbf{Q} = \mathbf{D}^c\mathbf{R}\mathbf{D}_s^{-c} \Rightarrow \mathbf{Q}\mathbf{D}_s^c = \mathbf{D}^c\mathbf{R}$.

[5] Chenhui Deng, Zhiqiang Zhao, Yongyu Wang, Zhiru Zhang, and Zhuo Feng. 2020. GraphZoom: A multi-level spectral approach for accurate and scalable graph embedding. In *Proceedings of the ICLR*.
[6] P. Erdös and A. Rényi. 1959. On random graphs I. *Publicationes Mathematicae Debrecen* 6 (1959), 290.
[7] Matthew Fahrbach, Gramoz Goranci, Richard Peng, Sushant Sachdeva, and Chi Wang. 2020. Faster graph embeddings via coarsening. In *Proceedings of the ICML*.
[8] Wenfei Fan, Jianzhong Li, Xin Wang, and Yinghui Wu. 2012. Query preserving graph compression. In *Proceedings of the ACM SIGMOD International Conference on Management of Data*. 157–168.
[9] G. Fu, C. Hou, and X. Yao. 2019. Learning topological representation for networks via hierarchical sampling. In *Proceedings of the IJCNN*.
[10] Nathan Halko, Per-Gunnar Martinsson, and Joel A. Tropp. 2011. Finding structure with randomness: Probabilistic algorithms for constructing approximate matrix decompositions. *SIAM Rev.* 53, 2 (2011), 217–288.
[11] Nasrin Hassanlou, Maryam Shoaran, and Alex Thomo. 2013. Probabilistic graph summarization. In *Proceedings of the WAIM*.
[12] Paul W. Holland, Kathryn Blackmond Laskey, and Samuel Leinhardt. 1983. Stochastic blockmodels: First steps. *Soc. Netw.* 5, 2 (1983), 109–137.
[13] Jeff Johnson, Matthijs Douze, and Hervé Jégou. 2019. Billion-scale similarity search with GPUs. *IEEE Trans. Big Data* 7, 3 (2019), 535–547.
[14] Brian Karrer and Mark E. J. Newman. 2011. Stochastic blockmodels and community structure in networks. *Phys. Rev. E* 83, 1 (2011), 016107.
[15] George Karypis and Vipin Kumar. 1998. A fast and high quality multilevel scheme for partitioning irregular graphs. *SIAM J. Sci. Comput.* (1998), 359392.
[16] Kifayat Ullah Khan, Waqas Nawaz, and Young-Koo Lee. 2017. Set-based unified approach for summarization of a multi-attributed graph. In *Proceedings of the WWW*. 543–570.
[17] Jihoon Ko, Yunbum Kook, and Kijung Shin. 2020. Incremental lossless graph summarization. In *Proceedings of the KDD*.
[18] Danai Koutra, U. Kang, Jilles Vreeken, and Christos Faloutsos. 2014. Vog: Summarizing and understanding large graphs. In *Proceedings of the SDM*.
[19] Kyuhan Lee, Hyeonsoo Jo, Jihoon Ko, Sungsu Lim, and Kijung Shin. 2020. SSumM: Sparse summarization of massive graphs. In *Proceedings of the KDD*.
[20] Kristen LeFevre and Evimaria Terzi. 2010. GraSS: Graph structure summarization. In *Proceedings of the ICDM*.
[21] Xiangfeng Li, Shenghua Liu, Zifeng Li, Xiaotian Han, Chuan Shi, Bryan Hooi, He Huang, and Xueqi Cheng. 2020. Flowscope: Spotting money laundering based on graphs. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 34. 4731–4738.
[22] Jiongqian Liang, Saket Gurukar, and Srinivasan Parthasarathy. 2020. MILE: A multi-level framework for scalable graph embedding. Retrieved from https://arXiv:1802.09612
[23] Yuzhi Liang, Yukun Wang, Kai Lei, Min Yang, Ziyu Lyu et al. 2020. Reachability preserving compression for dynamic graph. *Info. Sci.* 520 (2020), 232–249.
[24] Wenqing Lin, Feng He, Faqiang Zhang, Xu Cheng, and Hongyun Cai. 2020. Initialization for network embedding: A graph partition approach. In *Proceedings of the WSDM*.
[25] Yike Liu, Tara Safavi, Abhilash Dighe, and Danai Koutra. 2018. Graph summarization methods and applications: A survey. *Comput. Surveys* 51, 3 (2018), 1–34.
[26] Hossein Maserrat and Jian Pei. 2010. Neighbor query friendly compression of social networks. In *Proceedings of the 16th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. 533–542.
[27] Arpit Merchant, Michael Mathioudakis, and Yanhao Wang. 2022. Graph summarization via node grouping: A spectral algorithm. In *Proceedings of the 16th ACM International Conference On Web Search And Data Mining*.
[28] Saket Navlakha, Rajeev Rastogi, and Nisheeth Shrivastava. 2008. Graph summarization with bounded error. In *Proceedings of the SIGMOD*.
[29] Amin Emamzadeh Esmaeili Nejad, Mansoor Zolghadri Jahromi, and Mohammad Taheri. 2021. Graph compression based on transitivity for neighborhood query. *Info. Sci.* 576 (2021), 312–328.
[30] Mark Newman. 2018. *Networks*. Oxford University Press.
[31] M. E. J. Newman. 2006. Modularity and community structure in networks. *Proc. Natl. Acad. Sci. U.S.A.* 103, 23 (2006), 8577–8582. https://doi.org/10.1073/pnas.0601602103
[32] Bryan Perozzi, Rami Al-Rfou, and Steven Skiena. 2014. DeepWalk: Online learning of social representations. In *Proceedings of the KDD*.
[33] Jiezhong Qiu, Yuxiao Dong, Hao Ma, Jian Li, Kuansan Wang, and Jie Tang. 2018. Network embedding as matrix factorization: Unifying deepwalk, LINE, PTE, and node2vec. In *Proceedings of the WSDM*.

[34] Qiang Qu, Siyuan Liu, Christian S. Jensen, Feida Zhu, and Christos Faloutsos. 2014. Interestingness-driven diffusion process summarization in dynamic networks. In *Proceedings of the ECML-PKDD*.
[35] Matteo Riondato, David García-Soriano, and Francesco Bonchi. 2017. Graph summarization with quality guarantees. *Data Min. Knowl. Discov.* 31, 2 (2017), 314–349.
[36] Amin Sadri, Flora D Salim, Yongli Ren, Masoomeh Zameni, Jeffrey Chan, and Timos Sellis. 2017. Shrink: Distance preserving graph compression. *Info. Syst.* 69 (2017), 180–193.
[37] Neil Shah, Danai Koutra, Tianmin Zou, Brian Gallagher, and Christos Faloutsos. 2015. Timecrunch: Interpretable dynamic graph summarization. In *Proceedings of the KDD*.
[38] Hua-Wei Shen, Xue-Qi Cheng, and Jia-Feng Guo. 2011. Exploring the structural regularities in networks. *Phys. Rev. E* 84, 5 (2011), 056111.
[39] Kijung Shin, Amol Ghoting, Myunghwan Kim, and Hema Raghavan. 2019. Sweg: Lossless and lossy summarization of web-scale graphs. In *Proceedings of the WWW*.
[40] Jian Tang, Meng Qu, Mingzhe Wang, Ming Zhang, Jun Yan, and Qiaozhu Mei. 2015. LINE: Large-scale information network embedding. In *Proceedings of the WWW*.
[41] Nan Tang, Qing Chen, and Prasenjit Mitra. 2016. Graph stream summarization: From big bang to big crunch. In *Proceedings of the SIGMOD*.
[42] Ye Wu, Zhinong Zhong, Wei Xiong, and Ning Jing. 2014. Graph summarization for attributed graphs. In *Proceedings of the ICIEE*.
[43] Yujun Yan, Jiong Zhu, Marlena Duda, Eric Solarz, Chandra Sripada, and Danai Koutra. 2019. GroupINN: Grouping-based interpretable neural network-based classification of limited, noisy brain data. In *Proceedings of the KDD*.
[44] Quinton Yong, Mahdi Hajiabadi, Venkatesh Srinivasan, and Alex Thomo. 2021. Efficient graph summarization using weighted lsh at billion-scale. In *Proceedings of the International Conference on Management of Data*. 2357–2365.
[45] Houquan Zhou, Shenghua Liu, Kyuhan Lee, Kijung Shin, Huawei Shen, and Xueqi Cheng. 2021. DPGS: Degree-preserving graph summarization. In *Proceedings of the SDM* (2021).

Received 16 April 2023; revised 22 September 2023; accepted 29 January 2024