Graph Summarization via Node Grouping: A Spectral Algorithm

Arpit Merchant
arpit.merchant@helsinki.fi
University of Helsinki
Helsinki, Finland

Michael Mathioudakis
michael.mathioudakis@helsinki.fi
University of Helsinki
Helsinki, Finland

Yanhao Wang
yhwang@dase.ecnu.edu.cn
East China Normal University
Shanghai, China

ABSTRACT

Graph summarization via node grouping is a popular method to build concise graph representations by grouping nodes from the original graph into supernodes and encoding edges into superedges such that the loss of adjacency information is minimized. Such summaries have immense applications in large-scale graph analytics due to their small size and high query processing efficiency. In this paper, we reformulate the loss minimization problem for summarization into an equivalent integer maximization problem. By initially allowing relaxed (fractional) solutions for integer maximization, we analytically expose the underlying connections to the spectral properties of the adjacency matrix. Consequently, we design an algorithm called SpecSumm that consists of two phases. In the first phase, motivated by spectral graph theory, we apply k-means clustering on the k largest (in magnitude) eigenvectors of the adjacency matrix to assign nodes to supernodes. In the second phase, we propose a greedy heuristic that updates the initial assignment to further improve summary quality. Finally, via extensive experiments on 11 datasets, we show that SpecSumm efficiently produces high-quality summaries compared to state-of-the-art summarization algorithms and scales to graphs with millions of nodes.

CCS CONCEPTS

• **Theory of computation → Mixed discrete-continuous optimization**; **Integer programming**; • **Mathematics of computing → Graph algorithms**.

KEYWORDS

Graph Summarization, Spectral Algorithms, Clustering

ACM Reference Format:
Arpit Merchant, Michael Mathioudakis, and Yanhao Wang. 2023. Graph Summarization via Node Grouping: A Spectral Algorithm. In *Proceedings of the Sixteenth ACM International Conference on Web Search and Data Mining (WSDM '23), February 27-March 3, 2023, Singapore, Singapore.* ACM, New York, NY, USA, 9 pages. https://doi.org/10.1145/3539597.3570441

1 INTRODUCTION

Graphs have become ubiquitous in diverse fields such as sociology, bioinformatics, and computer science to model different types of relations among objects [24, 38]. Understanding their structure, querying their properties, and designing meaningful visualizations of such graphs can lead to deeper insights about various phenomena. With increasing graph sizes, a necessary first step for graph analytics is to build an accurate yet small representation of the original graph that is more efficient to process [40]. To this end, we study the graph summarization problem wherein the goal is to concisely preserve overall graph structure while reducing its size.

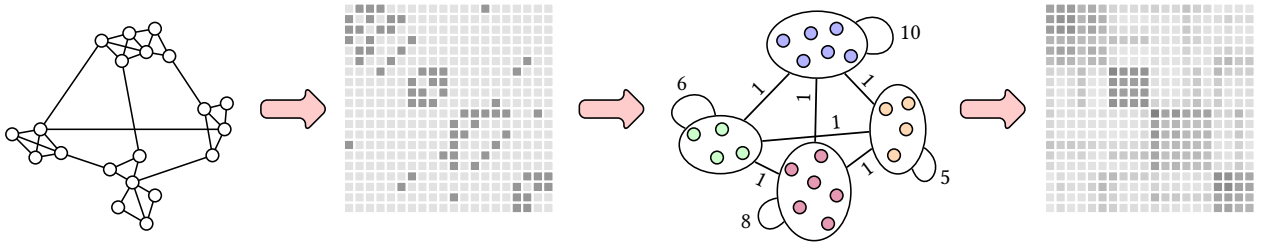
Graph summarization has been extensively studied in literature (see [26] for a comprehensive survey). In general, algorithms for this task can be broadly classified into three categories based on different objectives, namely, *(a) query efficiency*, *(b) space reduction*, and *(c) reconstruction error*. Respectively, these categories include *(i) application-based* methods tailored for efficiently processing specific types of queries such as reachability [12], distances [42], neighborhoods [27], etc., *(ii) compression-based* methods that encode graph structure using fewer bits [5, 8, 28], and *(iii) aggregation-based* methods that combine adjacent nodes and edges into supernodes and superedges to best preserve topology information [20, 21, 33].

One popular approach among the above, as well as the focus of this work, is to create aggregation-based supergraph summaries [3, 20, 21, 33] or k-summaries, for short. Informally, a k-summary is constructed as follows: given size k as input, each node in the original graph is assigned to one of k supernodes. Then, a superedge is added between each pair of supernodes. Each superedge is assigned a weight equal to the number of edges in the original graph between the nodes within the corresponding supernode pair. The quality of a k-summary is measured by the reconstruction error, typically l_2-error, defined as the entry-wise difference between the original and recovered adjacency matrices. Thus, the summarization objective is to minimize the l_2-error. Aggregation-based algorithms for this task in literature exhibit two primary limitations. First, most algorithms including GraSS [21] and S2L [33] cannot scale to large graphs because of high time complexity, dimensionality, or memory footprint. Second, algorithms that can scale such as SSumM [20] produce summaries with higher reconstruction errors and poorly preserved graph topologies (eg. number of triangles).

Our Contributions. To address these limitations, we design a scalable algorithm to build a k-summary that best preserves adjacency information. We reformulate the l_2-error minimization problem into an equivalent integer trace maximization problem. An integral solution indicates the supernode that each node of the original graph belongs to. We start by relaxing the integer problem to allow fractional memberships. We theoretically prove that the k largest *in magnitude* eigenvectors of the adjacency matrix provide a non-trivial lower bound for the relaxed problem. We also propose an orthonormality-constrained steepest ascent algorithm (called Ocsa) adapted from [43] to show that the eigenvectors represent at least a locally optimal solution. Our approach to building the summary, which we call SpecSumm, comprises of two phases. In the first phase, motivated by spectral graph theory, we apply k-means clustering

WSDM '23, February 27-March 3, 2023, Singapore, Singapore

ACM ISBN 978-1-4503-9407-9/23/02.
https://doi.org/10.1145/3539597.3570441



(a) Original Graph (b) Original Adjacency Matrix (c) 4-Summary Supergraph (d) Recovered Adjacency Matrix

Figure 1: Illustration of a 4-summary created by SpecSumm and adjacency matrix recovered from the summary on a toy graph.

on the eigenvector solution to obtain an initial membership matrix. The second phase comprises of a heuristic that samples nodes uniformly at random and greedily updates their membership to a different supernode if the reassignment improves the objective. The k-summary is constructed from the final membership matrix after reassignments. Figure 1 illustrates a 4-summary of a toy graph obtained via SpecSumm and the recovered adjacency matrix.

In addition, we provide extensive empirical evidence for the efficacy of our approach. We implement three variants of Ocsa using the eigenvectors, a QR-decomposition of a random matrix, and a DeepWalk embedding [31] as initial feasible solutions for the relaxed problem. We show that Ocsa converges to the eigenvector solution after sufficiently many iterations. We compare SpecSumm, with and without the reassignment heuristic, with Ocsa and two state-of-the-art baselines over 11 real graphs ranging from 1,000 to 2.3 million nodes. Across different datasets and summary sizes, results show that SpecSumm consistently and efficiently builds summaries with low reconstruction errors. Lastly, we analyze the scalability of SpecSumm on three large graphs via an ablation study for construction time and summary quality as a function of the number of eigenvectors and summary size. We observe that smaller summaries based on more eigenvectors can be built up to 17× faster than larger summaries based on fewer eigenvectors while maintaining comparable quality, thereby offering useful trade-offs for real-world applications. Our main contributions include:

- We introduce a novel reformulation for the k-summary problem and analytically motivate the design of our algorithms.
- We propose SpecSumm that clusters the eigenvectors of the original adjacency matrix to create an initial high-quality k-summary and further refines the summary using a greedy heuristic.
- We show via extensive experiments that SpecSumm constructs summaries of upto 22.5% and 76.1% higher quality on small to medium sized graphs compared to state-of-the-art baselines S2L and SSumM while running upto 200× faster than S2L. Further, SpecSumm scales well to massive graphs with millions of nodes and produces concise, meaningful summaries within 3 hours.

2 RELATED WORK

We categorize previous studies into three broad classes based on their summarization objectives. We refer interested readers to Liu et al. [26] for a more extensive survey.

Query Efficiency. Methods in this class construct summaries tailored for processing specific types of graph queries. Maserrat and Pei [27] and Nejad et al. [29] designed summaries that efficiently search for neighbors of a query vertex. Toivonen et al. [42] and Sadri et al. [36] summarize weighted graphs to preserve the distances between vertices. Fan et al. [12] and Liang et al. [25] devised graph summaries for efficient reachability queries. A separate but related set of methods in this class construct summaries for user-specified utilities [15, 19], modularity [14], and motifs [11]. However, these summaries do not include adjacency recovery procedures and further, our goal is to build a general-purpose summary for different types of queries. This makes a direct comparison infeasible.

Space Reduction. Methods in this class store a (lossless or lossy) representation of a graph using minimum possible space. For instance, VoG [18] uses Minimum Description Length for compression to encode a vocabulary of subgraphs such as stars and cliques. Subsequent studies proposed different graph reordering and encoding schemes to improve compression ratios [4–8, 10]. Aggregation-based schemes for compression proposed by Navlakha et al. [28] among others [13, 16, 17, 39, 41, 46] maintain extra edge corrections to recover the missing information due to node/edge grouping. However, unlike our paradigm, these methods either do not create hypergraphs or they do not minimize reconstruction loss or both. Within this class, SSumM [20] presents the closest summary specification to ours and thus we include it as a baseline for comparison. Note, SSumM has a different objective: it minimizes the number of bits required for storage jointly with the reconstruction error, which is achieved by coarsening supernodes and pruning superedges. As a result, SSumM cannot guarantee that the summary size is exactly equal to the user-specified input k and, as shown in the experiments, it exhibits higher reconstruction errors than our algorithms while having higher or comparable efficiencies.

Reconstruction Error. Methods in this class build supergraph summaries such that the error in reconstructing adjacency matrices is minimized and are thus closely related to our work. GraSS [21] constructs a k-summary by repeatedly merging a pair of supernodes that maximally decreases the reconstruction error until only k supernodes remain. ScalableSumm [3] adopts a similar merging-based scheme as GraSS. Additionally, it utilizes a sampling method for candidate pair selection and a count-min sketch [9] for reconstruction error estimation. However, merging-based schemes suffer from low summary quality when the summary size k is small. Riondato et al. [33] proposed S2L which employs k-means clustering on the rows of the adjacency matrix to create supernodes.

S2L provides a theoretical guarantee on the l_p-reconstruction error of the output summary. Nevertheless, S2L incurs costly distance computations given the high dimensionality of the adjacency matrix and thus is not scalable to massive graphs. We compare with S2L in the experiments and the results confirm that SpecSumm outperforms S2L in terms of both summary quality and efficiency.

3 PRELIMINARIES

Consider an unweighted, undirected graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ where $\mathcal{V}$ is a set of n nodes and $\mathcal{E}$ is a set of m edges. We denote its adjacency matrix by $A \in \{0,1\}^{n\times n}$. A k-partition of $\mathcal{V}$ is defined as $V = \{V_1, \ldots, V_k\}$ such that $\forall i \neq j \in [k], V_i \cap V_j = \emptyset$ and $\bigcup_{i=1}^{k} V_i = \mathcal{V}$. Let $X_V \in \{0,1\}^{n\times k}$ represent a *membership matrix* corresponding to partition V, where the (i,j)-th entry is 1 if node i belongs to set V_j and 0 otherwise. Each node is assigned to exactly one partition and thus X_V is orthogonal. Let $Z_V = X_V(X_V^\top X_V)^{-1/2}$ be the associated normalized membership matrix where $Z_V^\top Z_V = \mathbb{I}$. We denote $P_V = Z_V Z_V^\top$ as a smoothing operator, i.e., the orthogonal projection onto the subspace spanned by the columns of Z_V.

Given a k-partition V of $\mathcal{V}$, let $V \times V$ denote the set of all superedges between every pair of subsets in V. Then, a k-summary of $\mathcal{G}$ is defined as a weighted, supergraph $S_{\mathcal{G},V} = \{V, V \times V\}$ of $|V| = k$ supernodes and $k(k-1)/2$ superedges. For $i, j \in [k]$, the weight of a superedge between supernodes V_i and V_j is given by:

$$A_S(V_i, V_j) := \frac{\sum_{u\in V_i, v\in V_j} A(u,v)}{|V_i| \cdot |V_j|}, \tag{1}$$

where A_S is called the density matrix of S.[1] This weight denotes the fraction of actual edges in $\mathcal{G}$ between the nodes in V_i and V_j divided by the maximum possible number of edges. We use A_S to approximate the original adjacency matrix. This approximation recovered from a summary is referred to as a *lifted adjacency matrix* [33]. Its (u,v)-th entry captures the probability of the existence of an edge between u and v in $\mathcal{G}$. Specifically, $A_S^{\uparrow}(u,v) = A_S(S(u), S(v))$, where $S(u)$ represents the supernode that u belongs to. In matrix notation, the lifted adjacency matrix is written as $A_S^{\uparrow} = P_V A P_V$ [33]. The quality of a k-summary S is measured by the l_2-norm of the entry-wise difference between A and $A_S^{\uparrow}$ [20, 33]. Formally:

$$L\left(A, A_S^{\uparrow}\right) = \|A - A_S^{\uparrow}\|_2^2 = \sum_{u\in\mathcal{V}} \sum_{v\in\mathcal{V}} \left(A(u,v) - A_S^{\uparrow}(u,v)\right)^2 \tag{2}$$

This l_2-norm error is exactly twice that of the l_1-norm error, thereby making these errors equivalent [33]. Thus, we focus on finding a summary S that minimizes $L(A, A_S^{\uparrow})$. We rewrite l_2-error as follows:

Lemma 3.1. $L\left(A, A_S^{\uparrow}\right) = \boldsymbol{tr}[A^2] - \underbrace{\boldsymbol{tr}[\left(Z_S^\top A Z_S\right)^2]}_{\mathcal{F}_{Z_S}}$

We defer all proofs to Appendix A (see here for the full version). Since the first term, $\boldsymbol{tr}[A^2] = 2\cdot|\mathcal{E}|$ is a constant, the matrix Z_S that maximizes the second term, $\mathcal{F}_{Z_S}$, also minimizes $L(A,\cdot)$. Formally, we recast the graph summarization problem given graph $\mathcal{G}$ and size k as the following integer trace maximization problem:

[1] We omit $\mathcal{G}$ and V from the subscript for notational convenience.

Problem 1. *[Graph k-Summarization]*

$$\begin{aligned} \arg\max_{Z} \quad & \boldsymbol{tr}[\left(Z^\top A Z\right)^2] \\ \text{s.t.} \quad & Z^\top Z = \mathbb{I} \text{ where } Z = X\left(X^\top X\right)^{-1/2} \\ & X \in \{0,1\}^{n\times k} \end{aligned}$$

4 ALGORITHMS

In this section, we present our approach for graph k-summarization (i.e., Problem 1) along with the underlying analytical motivations. Our approach consists of three steps. First, we relax the membership matrix X to accept real entries with all other conditions remaining intact. Formally, this gives us the following relaxed problem:

Problem 2. *[Relaxed Graph k-Summarization]*

$$\begin{aligned} \arg\max_{Z} \quad & \boldsymbol{tr}[\left(Z^\top A Z\right)^2] \\ \text{s.t.} \quad & Z^\top Z = \mathbb{I} \text{ where } Z = X\left(X^\top X\right)^{-1/2} \\ & X \in \mathbb{R}^{n\times k} \end{aligned}$$

Second, in Section 4.1, we design two solutions for Problem 2. And third, in Section 4.2, we define a heuristic rounding algorithm to convert the relaxed solution to an integral solution for Problem 1.

4.1 Relaxed Graph k-Summarization

Consider the trivial solution when $k = n$. The following result is obtained immediately via substitution:

Lemma 4.1. *Given an adjacency matrix A and $k = n$, $Z = [\boldsymbol{e}_1, \ldots, \boldsymbol{e}_k]$ optimally solves Problem 2 where $\boldsymbol{e}_i$ are the eigenvectors of A.*

For general values of k, we write the objective in vector form. Let $Z = [z_1, \ldots, z_k]$ where z_i represents the i-th column of Z. Then:

$$\begin{aligned} \boldsymbol{tr}[\left(Z^\top A Z\right)^2] &= \boldsymbol{tr}[\left([z_1, \ldots, z_k]^\top A [z_1, \ldots, z_k]\right)^2] \\ &= \underbrace{\sum_{j=1}^{k} \left(z_j^\top A z_j\right)^2}_{T_1} + \underbrace{\sum_{j=1}^{k} \sum_{i\in[k]\setminus\{j\}} (z_j^\top A z_i)^2}_{T_2} \end{aligned} \tag{3}$$

A trivial lower bound for $\mathcal{F}_Z$ is 0 since the individual terms in Equation 3 are squares of scalar numbers. Below, for $k = \{1, \ldots, n\}$, we analyze the two terms, T_1 and T_2, to obtain non-trivial solutions.

Largest-Magnitude Eigenvectors. Our main result proves that the k largest (*in magnitude*) eigenvectors of A represent a non-trivial lower bound on the value of the relaxed objective function and thus a non-trivial feasible solution to Problem 2.

Theorem 4.2. *A constructive lower bound for the maximization objective (Problem 2) is given as follows:*

$$\boldsymbol{tr}[\left(Z^\top A Z\right)^2] \geq \sum_{j=1}^{k} \lambda_j^2, \tag{4}$$

where, for $j \in [k]$, λ_j is the j-th largest (in magnitude) eigenvalue of A. Further, this lower bound is achieved when $Z = [\boldsymbol{e}_1, \ldots, \boldsymbol{e}_k]$ where each $\boldsymbol{e}_j$ is the eigenvector corresponding to λ_j.

Algorithm 1: LM-EigVecs (Relaxed Problem)

Input: Adjacency matrix A of $\mathcal{G}$; summary size k.
Output: Feasible solution Z for Problem 2.
// Compute the k largest (in magnitude) eigenvectors
$Z \leftarrow$ ComputeEigVecs(A, k)
return Z

Proof Sketch. We prove the above result by induction over k. In the base case when $k = 1$, there are no cross-terms (i.e., T_2) and T_1 consists of just one term. This yields the following result:

Lemma 4.3. *Given an adjacency matrix A and $k = 1$, the maximum value of the relaxed objective in Problem 2 is achieved by the largest-magnitude eigenvector $\boldsymbol{e}_1$ of A, i.e.,*

$$\arg\max_{Z} T_1 = \arg\max_{z} \boldsymbol{tr}[\left(z^\top A z\right)^2] = \boldsymbol{e}_1 \tag{5}$$

In the induction step, we show that for higher values of k, T_1 is maximized by the k largest (in magnitude) eigenvectors of A.

Lemma 4.4. *Given an adjacency matrix A and $k \in \{2, \ldots, n\}$, the set of self-terms, T_1, in the relaxed objective (Equation 3) is maximized by the k largest (magnitude) eigenvectors $\boldsymbol{e}_1, \ldots, \boldsymbol{e}_k$ of A.*

$$\arg\max_{Z} T_1 = \arg\max_{Z} \sum_{j=1}^{k} \left(z_j^\top A z_j\right)^2 = [\boldsymbol{e}_1, \ldots, \boldsymbol{e}_k] \tag{6}$$

Here, the cross-terms (T_2) always reduce to 0. It follows from the definition of eigenvectors and their mutual orthogonality whereby, for any $i, j \in [k], \boldsymbol{e}_i \neq \boldsymbol{e}_j, \left(\boldsymbol{e}_i^\top A \boldsymbol{e}_j\right)^2 = \left(\boldsymbol{e}_i^\top \lambda_j \boldsymbol{e}_j\right)^2 = 0$. Putting Lemmas 4.3 and 4.4 together proves Theorem 4.2. Algorithm 1 codifies it into a subroutine we refer to as LM-EigVecs.

We conjecture that these eigenvectors represent an optimal solution for the entire relaxed objective. That is, $\boldsymbol{tr}[\left(Z^\top A Z\right)^2] \leq \sum_{j=1}^{k} \lambda_j^2$ (*Conjecture 1*). However, this upper bound is not straightforward to determine for general k values and arbitrary graphs. Lemma 4.5 proves a non-constructive result identifying some cases when Conjecture 1 holds true.

Lemma 4.5. *Given $k \geq 2$ and a fixed adjacency matrix A such that the largest magnitude eigenvalue has multiplicity $m < k$, there exist feasible non-eigenvector solutions $Z = [z_1, \ldots, z_k]$ such that T_2 in the relaxed objective (Equation 3) is non-zero. Otherwise, if $m \geq k$, then $T_2 = 0$ and the eigenvector solution is optimal for Problem 2.*

$$\exists\, Z,\ s.t.\ Z^\top Z = \mathbb{I},\ and\ T_2 = \sum_{j=1}^{k} \sum_{i \in [k] \setminus \{j\}} (z_j^\top A z_i)^2 > 0 \tag{7}$$

In other words, Lemma 4.5 implies that there exist feasible orthonormal solutions Z for Problem 2 that are different from the eigenvector solution and the value of T_2 for these solutions is larger than the corresponding value of T_2 for the eigenvector solution (which is 0). Due to the non-constructive nature of the result, it is an open problem to obtain exact upper bounds for T_1 and T_2 in arbitrary graphs. So we design a heuristic algorithm called Ocsa to construct alternative candidates for such Z. In Section 5, we provide empirical evidence supporting our conjecture. We show that the solution returned by Ocsa converges to the eigenvector solution.

Algorithm 2: Ocsa (Relaxed Problem)

Input: Adjacency matrix A of graph $\mathcal{G}$; summary size k; error tolerance ϵ; number of iterations T.
Output: Feasible solution Z for Problem 2.
// Initial feasible solution
Draw a random matrix from $\mathbb{R}^{n \times k}$ as R
$Z^{(0)} \leftarrow$ QR-Decomposition(R)
for $t \leftarrow 1$ ***to*** T **do**
  // Preparation for gradient ascent
  Compute gradients $G^{(t)}$ (cf. Equation 8)
  Compute $\tau \leftarrow$ Newton-Line-Search [30]
  Compute skew-symmetric matrix $P^{(t)}$ (cf. Equation 9)
  Compute new iterate $Y^{(t)}(\tau)$ (cf. Equation 11)
  // Update the current solution
  $Z^{(t+1)} \leftarrow Z^{(t)} + \frac{\tau}{2} P^{(t)} \left(Z^{(t)} + Y^{(t)}(\tau)\right)$ (cf. Equation 10)
  if $\frac{\mathcal{F}_{Z^{(t+1)}} - \mathcal{F}_{Z^{(t)}}}{\mathcal{F}_{Z^{(t)}}} \leq \epsilon$, **then break**.
return $Z^{(T)}$

Orthogonality-Constrained Optimization Heuristic. Our algorithm, Ocsa, is directly adapted from Wen and Yin [43].

The set of feasible solutions $\mathcal{M} = \{Z \in \mathbb{R}^{n \times k} : Z^\top Z = \mathbb{I}\}$ is called a Stiefel Manifold. In problems involving such manifolds, there are usually no guarantees for obtaining the global maximizer [43]. Our iterative heuristic solution then relies on constraint-preserving steepest ascent. It proceeds as follows: As a first step, we construct a feasible initial solution denoted as $Z^{(0)}$. For instance, $Z^{(0)}$ may be the eigenvector solution obtained previously, or the Q matrix from the QR decomposition of a random $n \times k$ matrix. The second step comprises of T iterations. At each iteration $t \in [T]$, we first compute the gradient of the objective function with respect to the current solution $Z^{(t)}$ and then update $Z^{(t)}$.

Lemma 4.6. *Given an adjacency matrix A and a solution Z, denote G as the $(n \times k)$-dimensional gradient matrix of the trace objective with respect to Z. Then, the (i, j)-th entry of the gradient is:*

$$G_{ij} = \frac{\partial\, \boldsymbol{tr}[\left(Z^\top A Z\right)^2]}{\partial\, Z_{ij}} = \boldsymbol{tr}[2(Z^\top A Z) \times (Z^\top A \boldsymbol{J}^{ij} + \boldsymbol{J}^{ji} A Z)] \tag{8}$$

where $\boldsymbol{J}^{ij}$ is the single-entry matrix of appropriate dimensions whose (i, j)-th entry is 1 and all other entries are 0.

Given Z and the gradient matrix G, we define P as:

$$P = G Z^\top + Z G^\top \tag{9}$$

Using steepest ascent, we find the best gradient direction and set the new solution as $Z^{(t+1)} = Z^{(t)} + \tau P^{(t)} Z^{(t)}$ where τ is the best step size computed using Newton's Line Search method (cf. Algorithm 3.2 [30]). However, $Z^{(t+1)}$ may not necessarily be orthonormal. Thus, we use the Cayley transformation as defined in OptStiefelGBB [43] to create the next constraint-preserving iterate, i.e.,

$$Z^{(t+1)} = Z^{(t)} + \frac{\tau}{2} P^{(t)} \left(Z^{(t)} + Y^{(t)}(\tau)\right) \tag{10}$$

where $Y^{(t)}(\tau)$ is given by:

$$Y^{(t)}(\tau) = Z^{(t)} Q^{(t)} \text{ and } Q^{(t)} = \left(\mathbb{I} + \frac{\tau}{2} P^{(t)}\right)^{-1} \left(\mathbb{I} - \frac{\tau}{2} P^{(t)}\right) \tag{11}$$

Algorithm 3: SpecSumm

Input: Adjacency matrix A of graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$; summary size k; number of samples per round D.
Output: Membership and density matrices X_S, A_S of summary S.
// Phase 1: Create initial node membership assignment
$Z \leftarrow$ LM-EigVecs(A, k) or Ocsa(A, k)
$X^{(0)} \leftarrow k$-Means(Z, k)
Compute the current best cost $C_{\text{best}} \leftarrow \mathcal{F}_{X^{(0)}}$
// Phase 2: Update node memberships (optional)
for $r \leftarrow 1$ ***to*** T **do**
  Sample D nodes from $\mathcal{V}$ without replacement
  for $v \in \{v_1, \ldots, v_D\}$ **do**
    Get the current supernode of v as $S(v)$
    for $j \in [k] \setminus \{S(v)\}$ **do**
      Reassign node v to supernode j
      Build a temporary membership matrix $\tilde{X_v}$
      Compute the new cost $C_{\text{new}} \leftarrow \mathcal{F}_{\tilde{X_v}}$
      if $C_{new} > C_{best}$ **then**
        $C_{\text{best}} \leftarrow C_{\text{new}}$
        Update the membership $X^{(r)} \leftarrow \tilde{X_v}$
$X_{\text{final}} \leftarrow X^{(T)}$
Compute densities A_S (cf. Equation 1)
return X_{final}, A_S

Wen and Yin [43] show that the update scheme in Equation 10 preserves orthonormality, maintains a smooth curve for $Y^{(t)}(\tau)$ over τ, and converges to a stationary point given sufficient iterations (cf. Lemma 3 [43]). Algorithm 2 presents the pseudocode for Ocsa.

4.2 The SpecSumm Algorithm

We now propose our algorithm called SpecSumm which consists of two phases, namely k-Means and Reassignment. In the first phase, SpecSumm converts the relaxed solution (obtained previously) into an integral solution using k-means clustering. In the second (optional) phase, SpecSumm improves the k-means solution using a greedy heuristic. We discuss each of these in further detail below.

k-Means Clustering. A good-quality summary S, as per Problem 1, implies placing nearby nodes in the same supernode and distant nodes in different supernodes. The final relaxed solution $Z^{(T)}$ represents an embedding of nodes in k-dimensional Euclidean space such that the summarization objective is optimized. Let $a_1, \ldots, a_n \in \mathbb{R}^k$ denote this embedding of n points where a_i is the i-th row of $Z^{(T)}$. To create supernodes, we use the continuous k-Means algorithm which constructs a set of k centroids $c_1, \ldots, c_k \in \mathbb{R}^k$ such that the following cost function is minimized:

$$\min_{c_1,\ldots,c_k} \quad \sum_{i=1}^{n} \|a_i - c_{l(i)}\|_2^2 \tag{12}$$

where $l(i)$ is the centroid closest to a_i. Then, the (i, j)-th entry of the membership matrix X is 1 if node $l(i) = j$ and 0 otherwise. Thus, each node is assigned to exactly one supernode.

Reassignment. One limitation of using k-Means alone is that it does not directly optimize the objective, $\mathcal{F}_Z$, in Problem 1. To improve the quality of the summary returned by k-Means, we propose Reassignment as a secondary heuristic. Let T denote the number of rounds. In each round $r \in [T]$, we proceed as follows: Let $X^{(r)}$ denote the current membership matrix. We randomly sample D nodes from $\mathcal{V}$ without replacement. For each sampled node v, we check if moving v from its current supernode, say $S(v)$, to another supernode improves the objective value ($\mathcal{F}_{X^{(r)}}$). If yes, then we reassign v to that supernode. If there are more than one such candidate supernodes, we reassign v to that supernode which results in the maximum increase in the current $\mathcal{F}_{X^{(r)}}$. Otherwise, we do not reassign v. At each step, and thus after T rounds, this ensures that Reassignment returns a feasible solution that is at least as good as the solution obtained from k-Means in the context of Problem 1. Finally, we use the final membership matrix $X^{(T)}$ to create the k-summary by computing edge densities according to Equation 1. Algorithm 3 presents the pseudocode of SpecSumm.

Time Complexity. The complexity of computing the top-k eigenvectors of a sparse symmetric matrix is $O(mkt_1)$ [2] where t_1 is the number of Arnoldi iterations. The complexity of computing a clustering using mini-batch k-Means is $O(nkt_2)$ where t_2 is the number of clustering iterations [37, 44]. Finally, computing the densities requires $O(m)$ time [33]. Thus, the total computation complexity of our algorithm is $O(mkt_1 + nkt_2)$. However, the widespread use and study of each of the components involved in SpecSumm indicates that scaling summarization to massive graphs is feasible.

Table 1: Dataset Statistics: number of nodes ($|\mathcal{V}|$), number of edges ($|\mathcal{E}|$), average degree (d_{avg}), density (ρ), diameter (D), clustering coefficient (C). † denotes originally disconnected graphs for which we use their largest connected component.

Dataset	Size		Graph Properties			
	$\lvert\mathcal{V}\rvert$	$\lvert\mathcal{E}\rvert$	d_{avg}	ρ	D	C
SBM [1]	1,000	29,872	59.74	5.98×10^{-2}	3	0.06
Cora† [38]	2,485	5,069	4.08	1.64×10^{-3}	19	0.24
PPI †[32]	3,852	37,841	19.65	5.10×10^{-3}	8	0.15
ca-GrQc† [23]	4,158	13,428	6.46	1.55×10^{-3}	17	0.56
LastFM-Asia [35]	7,624	27,806	7.29	9.57×10^{-4}	15	0.22
BlogCatalog† [32]	10,312	333,983	64.78	6.28×10^{-3}	5	0.46
Facebook [34]	22,470	171,002	15.22	6.77×10^{-4}	15	0.36
email-Enron† [24]	33,696	180,811	10.73	3.19×10^{-4}	13	0.51
Amazon [45]	334,863	925,872	5.52	1.65×10^{-5}	44	0.40
Youtube [45]	1,134,890	2,987,624	5.26	4.63×10^{-6}	20	0.08
Wikitalk [22]	2,394,385	5,021,410	4.19	1.75×10^{-6}	9	0.05

5 EXPERIMENTS

We perform extensive experiments to evaluate the efficacy of our algorithms. Section 5.1 describes our setup. Section 5.2 presents our main results. Extended results are deferred to Appendix B.

5.1 Setup

Datasets. We evaluate our algorithms on 11 publicly available datasets spanning various domains and with sizes ranging from 1K to 2.39M nodes. SBM [1] is a stochastic block model graph comprising of 20 clusters of 50 nodes each, with intra-cluster and inter-cluster probabilities set to 0.25 and 0.05, respectively. Cora [38] and ca-GrQc [23] are academic citation and collaboration networks. PPI [32] is a protein-protein interaction network. LastFM-Asia [35],

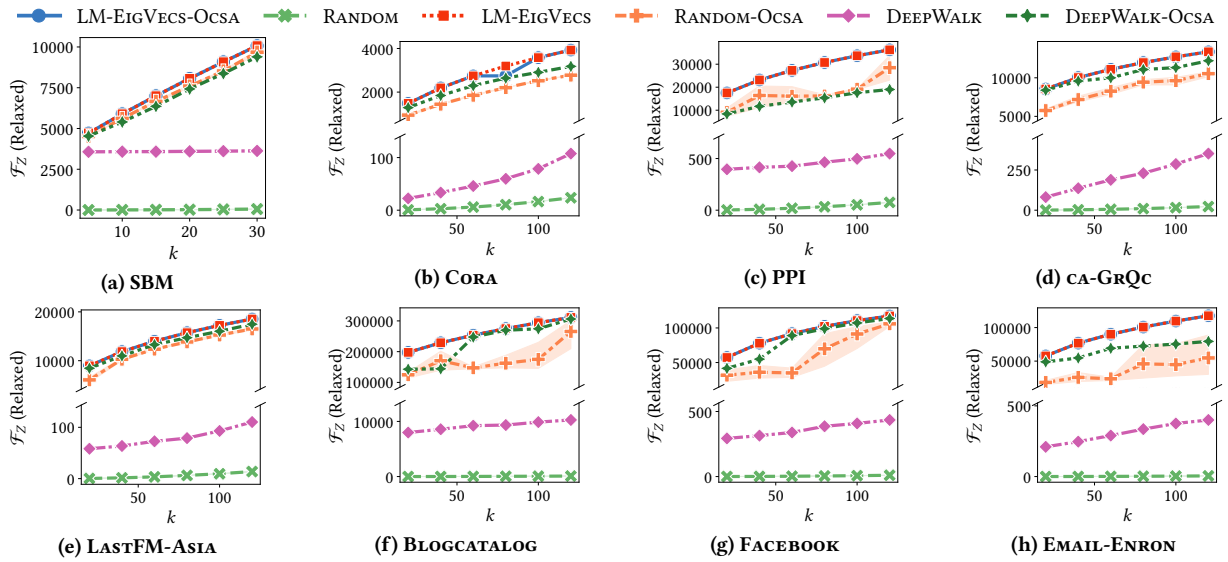


Figure 2: Objective value ($\mathcal{F}_Z$) of Problem 2 with respect to summary size k for different variants of LM-EigVecs and Ocsa.

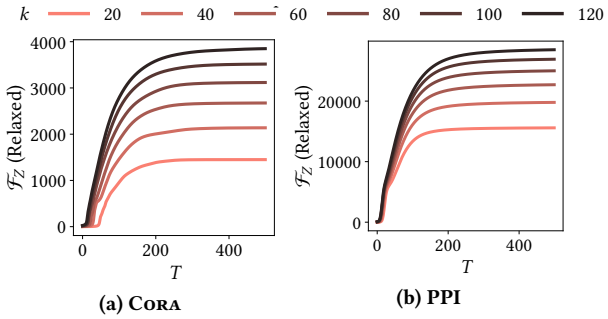


Figure 3: Objective value ($\mathcal{F}_Z$) of Problem 2 as a function of the number of iterations (T) for Random-Ocsa.

Blogcatalog [32], and Youtube [45] are social networks. Amazon [45] is a product co-purchasing network. Facebook [34] is a web-graph of Facebook sites. Email-Enron [24] and Wikitalk [22] are communication networks. If a graph is disconnected, we extract its largest connected component for our experiments. Table 1 summarizes the statistics of the processed datasets.

Algorithms. We evaluate the following algorithms for the relaxed problem: (i) LM-EigVecs (Algorithm 1), and three variants of Ocsa (Algorithm 2) depending on the choice of the initial feasible solution, namely (ii) LM-EigVecs-Ocsa (largest-magnitude eigenvectors) (iii) Random-Ocsa (random QR matrix), and (iv) DeepWalk-Ocsa (QR decomposition of a DeepWalk [31] node embedding).

For the integer problem, we consider two variants of our algorithm: (i) SpecSumm-R and (ii) SpecSumm that apply k-Means on the eigenvectors with and without the Reassignment heuristic, respectively. We compare against (iii) DeepWalk-Ocsa-KM (k-Means on the relaxed solution returned by DeepWalk-Ocsa) and two state-of-the-art competitors (iv) S2L [33] and (v) SSumM [20].

Parameter Setting. We construct graph summaries of size $k \in \{5, 10, \ldots, 30\}$ for SBM, $k \in \{20, 40, \ldots, 120\}$ for small graphs, and $k \in \{100, 250, 500, 1000, 2000, 5000\}$ for large graphs. Unless otherwise specified, the number of eigenvectors is set to k. Ocsa is executed for $T = 100$ iterations with initial step size $\tau = 0.001$ and tolerance $\epsilon = 0.001$. For fair comparison, all algorithms use the same Mini-Batch k-Means algorithm by Sculley [37] with kmeans++ initialization. For Reassignment, we set $T = 4$ (number of rounds) and $D = 500$ (number of samples per round) for each dataset and k.

Implementation. We implement our algorithms in Python 3. For SSumM, we use the Java version by Lee et al. [20]. All experiments were conducted on a Linux machine with 32 cores and 50GB RAM. Our code is available at https://version.helsinki.fi/ads/specsumm.

5.2 Experimental Results

Results for the Relaxed Problem. Figure 2 presents the trace objective value ($\mathcal{F}_Z$) achieved by LM-EigVecs and Ocsa as a function of k. As expected, $\mathcal{F}_Z$ always increases with k. LM-EigVecs attains the highest objective value across k in each dataset, with a maximum relative improvement of up to 52.12% over the nearest competitor, DeepWalk-Ocsa ($k = 20$ on PPI). Also, LM-EigVecs-Ocsa achieves exactly the same value of $\mathcal{F}_Z$ as LM-EigVecs because Ocsa always exits immediately after the first iteration (Line 10, Algorithm 2) thereby implying that it cannot find an ascent step that improves the initial solution. Lastly, we analyze the convergence of Ocsa on Cora and PPI by allowing it to run for up to 500 iterations. While Ocsa significantly improves upon the naive variants, i.e., Random and DeepWalk, given sufficiently many iterations, it converges to the $\mathcal{F}_Z$ value achieved by LM-EigVecs (cf. Figure 3). This provides empirical support for our conjecture that eigenvectors are a stationary point representing at least a local maxima.

Summary Quality. Table 2 reports $\mathcal{F}_Z$ values (averaged over 5 random seeds) of Problem 1 attained by each algorithm for varying summary sizes k on different datasets. SpecSumm-R outperforms other algorithms across datasets while SpecSumm mostly achieves the second highest values. SpecSumm-R is particularly effective

Table 2: Objective value, $\mathcal{F}_Z$ ($\times 10^3$), of Problem 1 for the summaries computed by each algorithm across different datasets. The values highlighted in blue denote the best quality and the underlined values denote the second-best quality.

Algorithm	k = 5	10	15	20	25	30
SSumM	3.32	3.21	3.39	3.26	3.55	3.56
S2L	3.57	3.59	3.61	3.62	3.63	3.64
DeepWalk-Ocsa-KM	3.74	3.79	3.86	3.9	3.93	4.12
SpecSumm	3.88	4.23	4.56	4.89	5.08	5.11
SpecSumm-R	3.97	4.39	4.86	5.22	5.42	5.48

(a) SBM

Algorithm	k = 20	40	60	80	100	120
SSumM	0.33	0.51	0.74	0.88	0.86	0.96
S2L	0.22	0.42	0.7	0.89	0.94	1.05
DeepWalk-Ocsa-KM	0.32	0.85	1.09	1.3	1.37	1.6
SpecSumm	0.49	0.86	1.25	1.48	1.43	1.59
SpecSumm-R	0.58	1.03	1.4	1.7	1.72	1.9

(b) Cora

Algorithm	k = 20	40	60	80	100	120
SSumM	2.27	2.21	1.98	2.58	2.51	3.12
S2L	5.18	5.49	6.58	7.09	6.99	7.26
DeepWalk-Ocsa-KM	3.78	5.09	5.11	5.22	5.64	5.43
SpecSumm	6.23	8.0	9.65	9.94	10.01	10.49
SpecSumm-R	7.38	9.6	11.23	11.92	12.22	12.96

(c) PPI

Algorithm	k = 20	40	60	80	100	120
SSumM	6.03	6.52	7.01	7.42	7.79	7.86
S2L	5.56	7.03	7.48	7.17	8.14	8.15
DeepWalk-Ocsa-KM	5.88	6.87	7.83	7.96	8.88	8.81
SpecSumm	6.58	7.33	7.7	7.65	7.85	8.41
SpecSumm-R	6.67	7.5	7.98	8.01	8.32	8.99

(d) ca-GrQc

Algorithm	k = 20	40	60	80	100	120
SSumM	2.33	3.05	3.08	3.81	3.83	3.9
S2L	3.34	4.22	5.01	5.34	6.02	6.22
DeepWalk-Ocsa-KM	3.62	4.89	5.87	6.87	7.69	8.31
SpecSumm	3.91	5.28	6.25	6.76	7.54	7.98
SpecSumm-R	3.99	5.43	6.48	7.03	7.94	8.41

(e) LastFM-Asia

Algorithm	k = 20	40	60	80	100	120
SSumM	70.78	70.87	65.88	64.42	67.62	67.44
S2L	96.52	105.66	108.9	112.03	113.62	112.31
DeepWalk-Ocsa-KM	58.14	59.56	106.68	114.8	109.96	121.01
SpecSumm	86.07	99.65	100.81	112.9	116.07	116.83
SpecSumm-R	91.17	107.19	109.75	121.27	123.94	124.88

(f) Blogcatalog

Algorithm	k = 20	40	60	80	100	120
SSumM	13.15	13.23	13.63	13.1	60.68	61.05
S2L	17.62	31.64	39.35	45.66	51.56	57.2
DeepWalk-Ocsa-KM	20.05	34.25	50.16	54.24	59.77	64.32
SpecSumm	26.96	40.7	49.04	54.26	59.65	63.48
SpecSumm-R	27.16	40.95	49.33	54.64	60.1	64.0

(g) Facebook

Algorithm	k = 20	40	60	80	100	120
SSumM	3.11	12.23	12.11	12.24	19.44	19.61
S2L	16.92	23.36	25.99	27.76	29.23	31.57
DeepWalk-Ocsa-KM	14.2	15.77	19.4	23.71	24.16	26.3
SpecSumm	17.25	21.0	23.81	27.09	29.02	32.44
SpecSumm-R	17.4	21.31	24.09	27.41	29.43	32.98

(h) Email-Enron

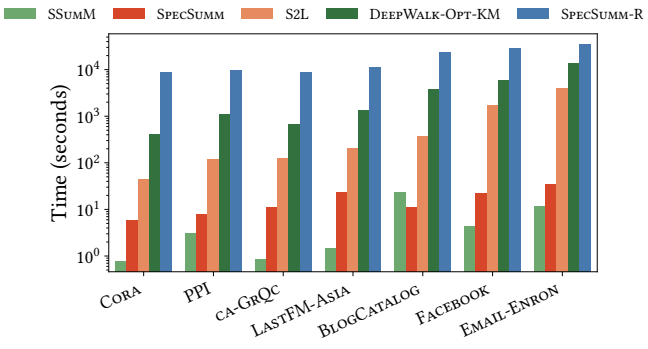
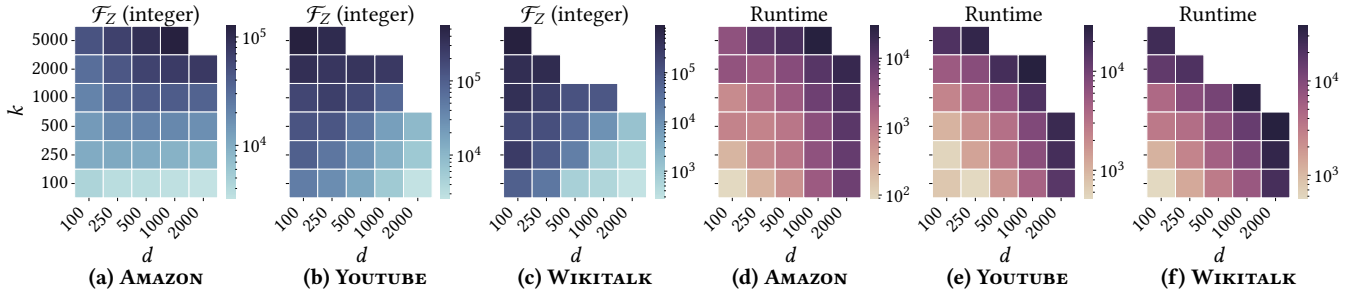


Figure 4: Running time (in log-scale) of different algorithms when the summary size $k = 120$.

on PPI where an improvement of upto 23.5% over SpecSumm is achieved. SpecSumm itself consistently produces higher-quality summaries than S2L– up to 51.1% on smaller graphs like Cora for $k = 120$. Moreover, the summary quality of SSumM is (upto 76.1%) inferior to that of SpecSumm as SSumM over-sparsifies the original graph by minimizing aggregate (over entire A) error and destroying topological structure. The results for l_2-reconstruction errors are included in Appendix B. As shown in Section 3, the problems of trace maximization and l_2-loss minimization are theoretically equivalent, and thus the results in terms of both objective values are consistent, i.e., any summary attaining a higher $\mathcal{F}_Z$ value than another summary must have a smaller l_2-loss as well.

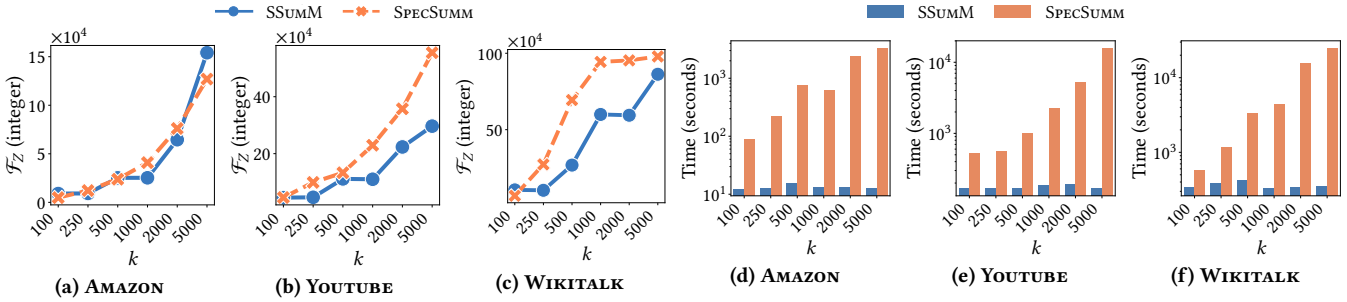
Beyond aggregate measures such as $\mathcal{F}_Z$, we also evaluate quality based on estimates for typical graph queries such as the number of triangles (cf. Appendix B) recovered from the summary. SpecSumm provides consistently more accurate estimates than S2L and SSumM. This further confirms the practical applicability of our approach.

Runtime. Figure 4 presents the average runtime (in seconds) of different algorithms. Due to space constraints, we only provide the results for $k = 120$. Generally, larger graph and summary sizes indicate longer running times as well. SpecSumm is up to 200× faster than S2L on Email-Enron while still providing a summary of better quality. On the other hand, DeepWalk-Ocsa-KM and SpecSumm-R are over two orders of magnitude slower than SpecSumm, requiring approximately 4 and 10 hours on Email-Enron, respectively. Such high overhead for DeepWalk-Ocsa-KM comes from the expensive gradient computation of Ocsa. And SpecSumm-R is slow since it cannot be parallelized and requires recomputing $\mathcal{F}_Z$ during each iteration (Line 10, Algorithm 3). The low efficiencies make both



(a) Amazon (b) Youtube (c) Wikitalk (d) Amazon (e) Youtube (f) Wikitalk

Figure 5: Trade-off between number of eigenvectors (d) and summary size (k) for the trace objective value ($\mathcal{F}_Z$) for Amazon, Youtube, and Wikitalk. Darker shades of blue and red represent higher quality and longer running times, respectively.



(a) Amazon (b) Youtube (c) Wikitalk (d) Amazon (e) Youtube (f) Wikitalk

Figure 6: Comparison between SpecSumm and SSumM in terms of $\mathcal{F}_Z$ and construction time as a function of k.

algorithms impractical when the graph sizes are large. Finally, although SSumM runs much faster than SpecSumm on small graphs (e.g., Cora and ca-GrQc), the gaps in time efficiency reduce when the graph size is larger (e.g., Blogcatalog and Email-Enron).

Scalability. We evaluate the scalability of SpecSumm by creating summaries for the three largest graphs, namely Amazon, Youtube, and Wikitalk. For these experiments, we set 12 hours as the time limit in each setting and parameter configuration.

Previously, given summary size k as the only input parameter, we computed k eigenvectors. However, it is possible to decouple the number of eigenvectors (say, d) from k. We trade off summary quality for efficiency by using fewer than k eigenvectors. Figure 5 depicts the $\mathcal{F}_Z$ and the total running time of SpecSumm as a function of d and k, respectively. Darker colors in blue and red indicate higher quality and longer times, accordingly. The missing regions indicate parameter settings for which SpecSumm did not complete within 12 hours. Results for S2L, DeepWalk-Ocsa-KM, and SpecSumm-R are omitted since they did not finish within 12 hours.

SpecSumm builds small summaries of large graphs very quickly. For $k = 100$ and $d = 100$, it only takes 89 and 569 seconds on Amazon and Wikitalk, respectively. As graph size increases, LM-EigVecs scales reasonably while k-Means is comparatively slower. For Wikitalk, LM-EigVecs requires up to 5.7 hours to compute 2000 eigenvectors whereas k-Means taking up to 6.9 hours to create $k = 5000$ clusters when $d = 100$. However, choosing appropriate values for d and k hugely affects quality. For a fixed k, increasing d up to k improves $\mathcal{F}_Z$ values. Conversely, constructing smaller summaries from larger number of eigenvectors results in even lower values of $\mathcal{F}_Z$. However, there exist intermediary settings for d and k that offer the best trade-off between summary quality and efficiency. That is, smaller summaries based on higher number of eigenvectors can be constructed up to 17× faster than larger summaries based on fewer eigenvectors while having comparable quality.

Finally, we compare SpecSumm with SSumM on the three largest graphs. Figure 6 reports $\mathcal{F}_Z$ and runtimes, respectively. While SSumM runs upto 3 orders of magnitude faster, its summary quality is (upto 2.3×) worse than that of SpecSumm. Note that we choose the compression ratios such that the size of the summary created by SSumM is slightly higher than k (e.g., 5,129 for k = 5,000 on Wikitalk) since SSumM cannot exactly control the summary size.

6 CONCLUSION

In this paper, we propose a novel SpecSumm algorithm for graph summarization via node aggregation. We motivate the use of the top-k largest in magnitude eigenvectors of the adjacency matrix to reduce the dimensionality of the problem, while also maintaining the relevant objective-specific information. We additionally provide a greedy reassignment heuristic to further improve the summary quality. We conduct extensive experiments on 11 real graphs to show that SpecSumm yields upto 22.5% and 76.1% higher quality summaries compared to S2L and SSumM and is up to 200× faster than S2L. Given its efficacy and simplicity, SpecSumm can scale to massive graphs and be easily deployed in real-world applications.

ACKNOWLEDGMENTS

Arpit Merchant would like to thank Ananth Mahadevan and Sachith Pai for useful suggestions regarding Ocsa and SSumM. Michael Mathioudakis is supported by University of Helsinki and Academy of Finland Projects MLDB (322046) and HPC-HD (347747).

REFERENCES

[1] Emmanuel Abbe, Afonso S. Bandeira, and Georgina Hall. 2016. Exact Recovery in the Stochastic Block Model. *IEEE Trans. Inf. Theory* 62, 1 (2016), 471–487.

[2] James Baglama and Lothar Reichel. 2005. Augmented Implicitly Restarted Lanczos Bidiagonalization Methods. *SIAM J. Sci. Comput.* 27, 1 (2005), 19–42.

[3] Maham Anwar Beg, Muhammad Ahmad, Arif Zaman, and Imdadullah Khan. 2018. Scalable Approximation Algorithm forÂăGraph Summarization. In *Advances in Knowledge Discovery and Data Mining*. Springer, Cham, 502–514.

[4] Paolo Boldi, Marco Rosa, Massimo Santini, and Sebastiano Vigna. 2011. Layered Label Propagation: A Multiresolution Coordinate-Free Ordering for Compressing Social Networks. In *Proceedings of the 20th International Conference on World Wide Web (WWW '11)*. ACM, New York, NY, USA, 587–596.

[5] Paolo Boldi and Sebastiano Vigna. 2004. The Webgraph Framework I: Compression Techniques. In *Proceedings of the 13th International Conference on World Wide Web (WWW '04)*. ACM, New York, NY, USA, 595–602.

[6] Giorgos Bouritsas, Andreas Loukas, Nikolaos Karalias, and Michael Bronstein. 2021. Partition and Code: learning how to compress graphs. *Advances in Neural Information Processing Systems* 34 (2021), 18603–18619.

[7] Gregory Buehrer and Kumar Chellapilla. 2008. A Scalable Pattern Mining Approach to Web Graph Compression with Communities. In *Proceedings of the 2008 International Conference on Web Search and Data Mining (WSDM '08)*. ACM, New York, NY, USA, 95–106.

[8] Flavio Chierichetti, Ravi Kumar, Silvio Lattanzi, Michael Mitzenmacher, Alessandro Panconesi, and Prabhakar Raghavan. 2009. On Compressing Social Networks. In *Proceedings of the 15th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD '09)*. ACM, New York, NY, USA, 219–228.

[9] Graham Cormode and S. Muthukrishnan. 2005. An improved data stream summary: the count-min sketch and its applications. *J. Algorithms* 55, 1 (2005), 58–75.

[10] Laxman Dhulipala, Igor Kabiljo, Brian Karrer, Giuseppe Ottaviano, Sergey Pupyrev, and Alon Shalita. 2016. Compressing Graphs and Indexes with Recursive Graph Bisection. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD '16)*. ACM, New York, NY, USA, 1535–1544.

[11] Cody Dunne and Ben Shneiderman. 2013. Motif Simplification: Improving Network Visualization Readability with Fan, Connector, and Clique Glyphs. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI '13)*. ACM, New York, NY, USA, 3247–3256.

[12] Wenfei Fan, Jianzhong Li, Xin Wang, and Yinghui Wu. 2012. Query Preserving Graph Compression. In *Proceedings of the 2012 ACM SIGMOD International Conference on Management of Data (SIGMOD '12)*. ACM, New York, NY, USA, 157–168.

[13] Wenfei Fan, Yuanhao Li, Muyang Liu, and Can Lu. 2021. Making Graphs Compact by Lossless Contraction. In *Proceedings of the 2021 International Conference on Management of Data (SIGMOD '21)*. ACM, New York, NY, USA, 472–484.

[14] Robert Görke, Pascal Maillard, Christian Staudt, and Dorothea Wagner. 2010. Modularity-Driven Clustering of Dynamic Graphs. In *Experimental Algorithms*. Springer, Berlin, Heidelberg, 436–448.

[15] Mahdi Hajiabadi, Jasbir Singh, Venkatesh Srinivasan, and Alex Thomo. 2021. Graph Summarization with Controlled Utility Loss. In *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining (KDD '21)*. ACM, New York, NY, USA, 536–546.

[16] Kifayat-Ullah Khan, Waqas Nawaz, and Young-Koo Lee. 2015. Set-based approximate approach for lossless graph summarization. *Computing* 97, 12 (2015), 1185–1207.

[17] Jihoon Ko, Yunbum Kook, and Kijung Shin. 2020. Incremental Lossless Graph Summarization. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining (KDD '20)*. ACM, New York, NY, USA, 317–327.

[18] Danai Koutra, U Kang, Jilles Vreeken, and Christos Faloutsos. 2015. Summarizing and Understanding Large Graphs. *Stat. Anal. Data Min.* 8, 3 (2015), 183–202.

[19] K. Ashwin Kumar and Petros Efstathopoulos. 2018. Utility-Driven Graph Summarization. *Proc. VLDB Endow.* 12, 4 (2018), 335–347.

[20] Kyuhan Lee, Hyeonsoo Jo, Jihoon Ko, Sungsu Lim, and Kijung Shin. 2020. SSumM: Sparse Summarization of Massive Graphs. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining (KDD '20)*. ACM, New York, NY, USA, 144–154.

[21] Kristen LeFevre and Evimaria Terzi. 2010. GraSS: Graph Structure Summarization. In *Proceedings of the 2010 SIAM International Conference on Data Mining (SDM)*. SIAM, 454–465.

[22] Jure Leskovec, Daniel Huttenlocher, and Jon Kleinberg. 2010. Predicting Positive and Negative Links in Online Social Networks. In *Proceedings of the 19th International Conference on World Wide Web (WWW '10)*. ACM, New York, NY, USA, 641–650.

[23] Jure Leskovec, Jon M. Kleinberg, and Christos Faloutsos. 2007. Graph evolution: Densification and shrinking diameters. *ACM Trans. Knowl. Discov. Data* 1, 1, Article 2 (2007), 41 pages.

[24] Jure Leskovec, Kevin J. Lang, Anirban Dasgupta, and Michael W. Mahoney. 2009. Community Structure in Large Networks: Natural Cluster Sizes and the Absence of Large Well-Defined Clusters. *Internet Math.* 6, 1 (2009), 29–123.

[25] Yuzhi Liang, Chen Chen, Yukun Wang, Kai Lei, Min Yang, and Ziyu Lyu. 2020. Reachability preserving compression for dynamic graph. *Inf. Sci.* 520 (2020), 232–249.

[26] Yike Liu, Tara Safavi, Abhilash Dighe, and Danai Koutra. 2018. Graph Summarization Methods and Applications: A Survey. *ACM Comput. Surv.* 51, 3, Article 62 (2018), 34 pages.

[27] 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 (KDD '10)*. ACM, New York, NY, USA, 533–542.

[28] Saket Navlakha, Rajeev Rastogi, and Nisheeth Shrivastava. 2008. Graph Summarization with Bounded Error. In *Proceedings of the 2008 ACM SIGMOD International Conference on Management of Data (SIGMOD '08)*. ACM, New York, NY, USA, 419–432.

[29] Amin Emamzadeh Esmaeili Nejad, Mansoor Zolghadri Jahromi, and Mohammad Taheri. 2021. Graph compression based on transitivity for neighborhood query. *Inf. Sci.* 576 (2021), 312–328.

[30] Jorge Nocedal and Stephen J. Wright. 1999. *Numerical Optimization*. Springer, Berlin, Heidelberg.

[31] Bryan Perozzi, Rami Al-Rfou, and Steven Skiena. 2014. DeepWalk: Online Learning of Social Representations. In *Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD '14)*. ACM, New York, NY, USA, 701–710.

[32] 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 Eleventh ACM International Conference on Web Search and Data Mining (WSDM '18)*. ACM, New York, NY, USA, 459–467.

[33] Matteo Riondato, David García-Soriano, and Francesco Bonchi. 2017. Graph summarization with quality guarantees. *Data Min. Knowl. Discov.* 31, 2 (2017), 314–349.

[34] Benedek Rozemberczki, Carl Allen, and Rik Sarkar. 2021. Multi-Scale attributed node embedding. *J. Complex Networks* 9, 2 (2021), 22 pages. cnab014.

[35] Benedek Rozemberczki and Rik Sarkar. 2020. Characteristic Functions on Graphs: Birds of a Feather, from Statistical Descriptors to Parametric Models. In *Proceedings of the 29th ACM International Conference on Information & Knowledge Management (CIKM '20)*. ACM, New York, NY, USA, 1325–1334.

[36] Amin Sadri, Flora D. Salim, Yongli Ren, Masoomeh Zameni, Jeffrey Chan, and Timos Sellis. 2017. Shrink: Distance preserving graph compression. *Inf. Syst.* 69 (2017), 180–193.

[37] David Sculley. 2010. Web-Scale k-Means Clustering. In *Proceedings of the 19th International Conference on World Wide Web (WWW '10)*. ACM, New York, NY, USA, 1177–1178.

[38] Prithviraj Sen, Galileo Namata, Mustafa Bilgic, Lise Getoor, Brian Gallagher, and Tina Eliassi-Rad. 2008. Collective Classification in Network Data. *AI Mag.* 29, 3 (2008), 93–106.

[39] Kijung Shin, Amol Ghoting, Myunghwan Kim, and Hema Raghavan. 2019. SWeG: Lossless and Lossy Summarization of Web-Scale Graphs. In *The World Wide Web Conference (WWW '19)*. ACM, New York, NY, USA, 1679–1690.

[40] Nan Tang, Qing Chen, and Prasenjit Mitra. 2016. Graph Stream Summarization: From Big Bang to Big Crunch. In *Proceedings of the 2016 International Conference on Management of Data (SIGMOD '16)*. ACM, New York, NY, USA, 1481–1496.

[41] Yuanyuan Tian, Richard A. Hankins, and Jignesh M. Patel. 2008. Efficient Aggregation for Graph Summarization. In *Proceedings of the 2008 ACM SIGMOD International Conference on Management of Data (SIGMOD '08)*. ACM, New York, NY, USA, 567–580.

[42] Hannu Toivonen, Fang Zhou, Aleksi Hartikainen, and Atte Hinkka. 2011. Compression of Weighted Graphs. In *Proceedings of the 17th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD '11)*. ACM, New York, NY, USA, 965–973.

[43] Zaiwen Wen and Wotao Yin. 2013. A feasible method for optimization with orthogonality constraints. *Math. Program.* 142, 1-2 (2013), 397–434.

[44] Donghui Yan, Ling Huang, and Michael I. Jordan. 2009. Fast Approximate Spectral Clustering. In *Proceedings of the 15th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD '09)*. ACM, New York, NY, USA, 907–916.

[45] Jaewon Yang and Jure Leskovec. 2012. Defining and Evaluating Network Communities Based on Ground-Truth. In *Proceedings of the ACM SIGKDD Workshop on Mining Data Semantics (MDS '12)*. ACM, New York, NY, USA, Article 3, 8 pages.

[46] Quinton Yong, Mahdi Hajiabadi, Venkatesh Srinivasan, and Alex Thomo. 2021. Efficient Graph Summarization Using Weighted LSH at Billion-Scale. In *Proceedings of the 2021 International Conference on Management of Data (SIGMOD '21)*. ACM, New York, NY, USA, 2357–2365.