Graph Summarization via Node Grouping: A Spectral Algorithm

Arpit Merchant
aprit.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 computation → 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...
We categorize previous studies into three broad classes based on whether they apply the reassignment heuristic, with and without OCSA. 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, 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 up to 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 up to 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. 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 [13, 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 cannot 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.
3 PRELIMINARIES

Consider an unweighted, undirected graph \( G = (V, E) \) where \( V \) is a set of \( n \) nodes and \( E \) is a set of \( m \) edges. We denote its adjacency matrix by \( A \in \{0, 1\}^{n \times n} \). A \( k \)-partition of \( V \) is defined as \( V = \{V_1, \ldots, V_k\} \) such that \( \forall i \neq j \in [k], V_i \cap V_j = \emptyset \) and \( \bigcup_{i=1}^{k} V_i = V \). 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 \( Z_V = X_V(X_V^T X_V)^{-1/2} \) be the associated normalized membership matrix where \( Z_V^T Z_V = I \).

Given a \( k \)-partition \( V' \) of \( V \), let \( V \times V' \) denote the set of all superedges between every pair of subsets in \( V \). Then, a \( k \)-summary of \( G \) is defined as a weighted, supergraph \( S_G, V' = \{V, V \times V'\} \) 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:

\[
A_S(V_i, V_j) := \sum_{u \in V_i, v \in V_j} A(u, v) / |V_i| \cdot |V_j|,
\]

where \( A_S \) is called the density matrix of \( S \).

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:

\[
\text{tr} \left( \left(Z^T A Z \right)^2 \right) = \sum_{j=1}^{k} \text{tr} \left( z_j^T A z_j \right)^2 + \sum_{i=1, i \neq j}^{k} \sum_{(l,j)} \left( z_j^T A z_l \right)^2.
\]

A trivial lower bound for \( F_Z \) 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.

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]**

\[
\begin{align*}
\text{arg max}_Z & \quad \text{tr} \left( \left(Z^T A Z \right)^2 \right) \\
\text{s.t.} & \quad Z^T Z = 1 \text{ where } Z = X (X^T X)^{-1/2} \\
& \quad X \in \mathbb{R}^{n \times k}
\end{align*}
\]

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 \( k = n \), \( Z = [e_1, \ldots, e_k] \) optimally solves Problem 2 where \( e_i \) are the eigenvectors of \( A \).

For general values of \( k \), we write the objective in vector form. Let 

\[
T_1 = \sum_j \text{tr} \left( z_j^T A z_j \right)^2
\]

\[
T_2 = \sum_{i=1, i \neq j}^{k} \sum_{(l,j)} \left( z_j^T A z_l \right)^2.
\]

A trivial lower bound for \( F_Z \) 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:

\[
\text{tr} \left( \left(Z^T A Z \right)^2 \right) \geq \sum_{j=1}^{k} \lambda_j^2,
\]

where, for \( j \in [k], \lambda_j \) is the \( j \)-th largest (in magnitude) eigenvalue of \( A \). Further, this lower bound is achieved when \( Z = [e_1, \ldots, e_k] \) where each \( e_j \) is the eigenvector corresponding to \( \lambda_j \).
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 $e_1$, i.e.,
$$\arg \max_{Z} T_1 = \arg \max_{Z} \text{tr}(Z^TAZ) = e_1$$

(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 $a_{e_1}, \ldots, a_{e_k}$ of $A$.

$$\arg \max_{Z} T_1 = \arg \max_{Z} \sum_{j=1}^{k} (z_j^TAz_j)^2 = [e_1, \ldots, e_k]$$

(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 $i, j \in [k], a_i \neq a_j, (e_i^TAe_j)^2 = (e_i^T\lambda e_j)^2 = 0$. 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, $\text{tr}[(Z^TAZ)^2] \leq \sum_{j=1}^{n} \lambda_j^2$ (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 $T_1$ is optimal for Problem 2.

$$\exists Z, \text{ s.t. } Z^TZ = I, \text{ and } T_2 = \sum_{j=1}^{k} \sum_{i \in [k]} (z_j^TAz_i)^2 > 0$$

(7)

In other words, Lemma 4.5 implies that there exist 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.

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

The set of feasible solutions $M = \{Z \in \mathbb{R}^{n \times k} : Z^TZ = I\}$ is called a Steifel 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 $Z(t)$.

**Algorithm 2: Ocsa (Relaxed Problem)**

1. **Input**: Adjacency matrix $A$ of graph $G$; summary size $k$; error tolerance $\epsilon$; number of iterations $T$.
2. **Output**: Feasible solution $Z$ for Problem 2.
3. **Initial feasible solution**
4. Draw a random matrix from $\mathbb{R}^{n \times k}$ as $R$.
5. **Z(0) \leftarrow** QR-DECOMPOSITION($R$).
6. **for $t \leftarrow 1$ to $T$ do**

   1. **Preparation for gradient ascent**
   2. Compute gradients $G^{(t)}$ (cf. Equation 8).
   3. Compute $r \leftarrow$ NEWTON-LINE-SEARCH [30].
   4. Compute skew-symmetric matrix $P^{(t)}$ (cf. Equation 9).
   5. Compute new iterate $Y^{(t)}(\tau)$ (cf. Equation 11).
   6. **Update the current solution**
   7. $Z^{(t+1)} \leftarrow Z^{(t)} + \frac{\tau}{2} P^{(t)} \left( Z^{(t)} + Y^{(t)}(\tau) \right)$ (cf. Equation 10).
   8. **if** $\frac{\tau Y^{(t+1)} - Y^{(t)}}{\tau Z^{(t)}} \leq \epsilon$, **then break**.
9. **return** $Z(T)$. 

Orthogonality-Constrained Optimization Heuristic. Our algorithm, Ocsa, is directly adapted from Wen and Yin [43].
Wen and Yin [43] show that the update scheme in Equation 10 preserves orthonormality, maintains a smooth curve for \( Y(\tau) \) over \( \tau \), 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 \textit{k-Means} and Reassignment. In the first phase, SpecSumm converts the relaxed solution (obtained previously) into an integral solution using \textit{k}-means clustering. In the second (optional) phase, SpecSumm improves the \textit{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 \( a_1, \ldots, a_D \in \mathbb{R}^k \) denote this embedding of \( D \) points where \( a_i \) is the \( i \)-th row of \( Z(T) \). To create supernodes, we use the continuous \textit{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:

\[
\min_{c_1, \ldots, c_k} \sum_{i=1}^{D} \| a_i - c_{l(i)} \|_2^2
\]

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 \textit{k-Means} alone is that it does not directly optimize the objective, \( F_Z \), in Problem 1. To improve the quality of the summary returned by \textit{k-Means}, we propose \textit{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 \( V \) 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 \( (F_Z(v)) \). 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 \( F_Z(v) \). Otherwise, we do not reassign \( v \). At each step, and thus after \( T \) rounds, this ensures that \textit{Reassignment} returns a feasible solution that is at least as good as the solution obtained from \( k \)-\textit{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 \textit{k-Means} is \( O(mkt_2) \) 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 \( O(mkt_1 + nkkt_2) \). However, the widespread use and study of each of the components involved in SpecSumm indicates that scaling summarization to massive graphs is feasible.

### 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],

| Dataset     | \(|V|\) | \(|E|\) | \(d_{avg}\) | \(\rho\) | \(D\) | \(C\) |
|-------------|--------|--------|-------------|--------|------|------|
| 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 |
We construct graph summaries of size \( k \in \{ 50, 100, 200, 400, 600 \} \) 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 Seulley [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 (\( F_Z \)) achieved by LM-EigVecs and Ocsa as a function of \( k \). As expected, \( F_Z \) 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 \( F_Z \) 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 \( F_Z \) 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 \( F_Z \) 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, $F_Z (\times 10^3)$, 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        |     | 3.22| 3.21| 3.20| 3.19| 3.18| 3.17|
| S2L          |     | 3.17| 3.16| 3.15| 3.14| 3.13| 3.12|
| DeepWalk-OcSa-KM |   | 3.10| 3.09| 3.08| 3.07| 3.06| 3.05|
| SpecSumm     |     | 3.03| 3.02| 3.01| 3.00| 2.99| 2.98|
| SpecSumm-R   |     | 2.88| 2.87| 2.86| 2.85| 2.84| 2.83|

(a) SBM

| Algorithm     | $k$ | 20  | 40  | 60  | 80  | 100 | 120 |
|---------------|-----|-----|-----|-----|-----|-----|-----|
| SSumM        |     | 2.27| 2.21| 1.98| 1.82| 1.79| 1.77|
| S2L          |     | 1.83| 1.76| 1.70| 1.66| 1.64| 1.62|
| DeepWalk-OcSa-KM |   | 1.58| 1.52| 1.49| 1.45| 1.42| 1.40|
| SpecSumm     |     | 1.23| 1.18| 1.14| 1.11| 1.09| 1.07|
| SpecSumm-R   |     | 0.97| 0.92| 0.88| 0.85| 0.82| 0.80|

(b) PPI

| Algorithm     | $k$ | 20  | 40  | 60  | 80  | 100 | 120 |
|---------------|-----|-----|-----|-----|-----|-----|-----|
| SSumM        |     | 1.31| 1.25| 1.20| 1.16| 1.13| 1.11|
| S2L          |     | 1.04| 0.99| 0.95| 0.92| 0.90| 0.88|
| DeepWalk-OcSa-KM |   | 0.76| 0.72| 0.68| 0.65| 0.63| 0.61|
| SpecSumm     |     | 0.49| 0.45| 0.42| 0.39| 0.36| 0.34|
| SpecSumm-R   |     | 0.22| 0.20| 0.18| 0.16| 0.14| 0.13|

(c) LastFM-Asia

| Algorithm     | $k$ | 20  | 40  | 60  | 80  | 100 | 120 |
|---------------|-----|-----|-----|-----|-----|-----|-----|
| SSumM        |     | 0.70| 0.65| 0.60| 0.56| 0.53| 0.51|
| S2L          |     | 0.53| 0.49| 0.46| 0.43| 0.41| 0.39|
| DeepWalk-OcSa-KM |   | 0.38| 0.35| 0.32| 0.29| 0.27| 0.25|
| SpecSumm     |     | 0.17| 0.15| 0.13| 0.11| 0.10| 0.09|
| SpecSumm-R   |     | 0.03| 0.02| 0.01| 0.01| 0.00| 0.00|

(d) Blogcatalog

| Algorithm     | $k$ | 20  | 40  | 60  | 80  | 100 | 120 |
|---------------|-----|-----|-----|-----|-----|-----|-----|
| SSumM        |     | 0.31| 0.25| 0.21| 0.18| 0.16| 0.14|
| S2L          |     | 0.26| 0.21| 0.19| 0.16| 0.14| 0.12|
| DeepWalk-OcSa-KM |   | 0.14| 0.12| 0.10| 0.09| 0.08| 0.07|
| SpecSumm     |     | 0.05| 0.04| 0.03| 0.02| 0.02| 0.02|
| SpecSumm-R   |     | 0.01| 0.01| 0.01| 0.01| 0.01| 0.01|

(e) Facebook

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

Table 2: Objective value, $F_Z (\times 10^3)$, 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     | $k$ | 20  | 40  | 60  | 80  | 100 | 120 |
|---------------|-----|-----|-----|-----|-----|-----|-----|
| SSumM        |     | 0.33| 0.31| 0.29| 0.27| 0.25| 0.23|
| S2L          |     | 0.22| 0.20| 0.18| 0.16| 0.14| 0.13|
| DeepWalk-OcSa-KM |   | 0.32| 0.30| 0.28| 0.26| 0.24| 0.23|
| SpecSumm     |     | 0.49| 0.48| 0.47| 0.46| 0.45| 0.44|
| SpecSumm-R   |     | 0.58| 0.57| 0.56| 0.55| 0.54| 0.54|

(f) Email-Enron

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

Graph Summarization via Node Grouping: A Spectral Algorithm WSDM '23, February 27-March 3, 2023, Singapore, Singapore

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 $F_Z$ value than another summary must have a smaller $l_2$-loss as well. Beyond aggregate measures such as $F_Z$, 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 $F_Z$ during each iteration (Line 10, Algorithm 3). The low efficiencies make both

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-sparifies the original graph by minimizing aggregate (over entire A) error and destroying topological structure. The results for $l_2$-reconstruction errors
Figure 5: Trade-off between number of eigenvectors (d) and summary size (k) for the trace objective value (F_Z) for Amazon, YouTube, and Wikitalk. Darker shades of blue and red represent higher quality and longer running times, respectively.

Figure 6: Comparison between SpecSumm and SSumM in terms of F_Z and construction time as a function of k.

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 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 F_Z values. Conversely, constructing smaller summaries from larger number of eigenvectors results in even lower values of 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 F_Z and runtimes, respectively. While SSumM runs up to 3 orders of magnitude faster, its summary quality is (up to 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 up to 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 OCSD and SSumM. Michael Mathioudakis is supported by University of Helsinki and Academy of Finland Projects MLDB (322046) and HPC-HD (347747).
