Data Mining and Knowledge Discovery manuscript No.
(will be inserted by the editor)

Graph Summarization with Quality Guarantees

Matteo Riondato · David García-Soriano · Francesco Bonchi

Received: date / Accepted: date

Abstract We study the problem of graph summarization. Given a large graph we aim at producing a concise lossy representation (a *summary*) that can be stored in main memory and used to approximately answer queries about the original graph much faster than by using the exact representation.

In this work we study a very natural type of summary: the original set of vertices is partitioned into a small number of supernodes connected by superedges to form a complete weighted graph. The superedge weights are the edge densities between vertices in the corresponding supernodes. To quantify the dissimilarity between the original graph and a summary, we adopt the *reconstruction error* and the *cut-norm error*. By exposing a connection between graph summarization and geometric clustering problems (i.e., k-means and k-median), we develop the *first polynomial-time approximation algorithms* to compute the best possible summary of a certain size under both measures.

We discuss how to use our summaries to store a (lossy or lossless) compressed graph representation and to approximately answer a large class of queries about the original graph, including adjacency, degree, eigenvector centrality, and triangle and subgraph counting. Using the summary to answer queries is very efficient as the running time to compute the answer depends on the number of supernodes in the summary, rather than the number of nodes in the original graph.

Keywords Graph analysis · Cut-norm · Approximation algorithms · Summarization · Regularity Lemma

---

A preliminary version of this work appeared in the proceedings of IEEE ICDM'14 [29].

M. Riondato
Two Sigma Investments, New York, NY, USA (part of the work performed during an internship at Yahoo Labs Barcelona and while affiliated to Brown University)
E-mail: matteo@twosigma.com

D. García-Soriano
Yahoo Labs, Barcelona, Spain
E-mail: davidgs@yahoo-inc.com

Francesco Bonchi
Yahoo Labs, Barcelona, Spain
E-mail: bonchi@yahoo-inc.com

1 Introduction

Data analysts in several application domains (e.g., social networks, molecular biology, communication networks, and many others) routinely face graphs with millions of vertices and billions of edges. In principle, this abundance of data should allow for a more accurate analysis of the phenomena under study. However, as the graphs under analysis grow, mining and visualizing them become computationally challenging tasks. In fact, the running time of most graph algorithms grows with the size of the input (number of vertices and/or edges): executing them on huge graphs might be impractical, especially when the input is too large to fit in main memory.

Graph summarization speeds up the analysis by creating a *lossy concise representation of the graph* that fits into main memory. Answers to otherwise expensive queries can then be computed using the summary without accessing the exact representation on disk. Query answers computed on the summary incur in a minimal loss of accuracy. When multiple graph analysis tasks can be performed on the same summary, the cost of building the summary is amortized across its life cycle. Summaries can also be used for privacy purposes [22], to create easily interpretable visualizations of the graph [28], and to store a compressed representation of the graph, either lossless or lossy (as we do in Sect. 5).

The biggest challenge in graph summarization is developing efficient algorithms to build *high-quality* summaries, i.e., summaries that closely resemble the original graphs while respecting the space requirements specified by the user.

1.1 Background and related work

A wide variety of concepts have been explored in the literature under the name of graph summarization, each tailored to a different application.

LeFevre and Terzi [22] propose to use an enriched "supergraph" as a summary, associating an integer to each supernode (a set of vertices) and to each superedge (an edge between two supernodes), representing respectively the number of edges (in the original graph) between vertices in the supernode and between the two sets of vertices connected by the superedge, respectively. From this lossy representation one can infer an *expected adjacency matrix*, where the expectation is taken over the set of *possible worlds* (i.e., graphs that are compatible with the summary). Thus, from the summary one can derive approximated answers for graph properties queries, as the expectation of the answer over the set of possible worlds. The GraSS algorithm by LeFevre and Terzi [22] follows a greedy heuristic resembling an agglomerative hierarchical clustering using Ward's method [35] and as such can not give any guarantee on the quality of the summary. We propose algorithms to compute summaries of *guaranteed quality* (a constant or polynomial factor from the optimal). This theoretical property is also verified empirically (see Sect. 6): our algorithms build more representative summaries and are much more efficient and scalable than GraSS in building those summaries.

Navlakha et al [28] propose a summary consisting of two components: a graph of "supernodes" (sets of nodes) and "superedges" (sets of edges), and a table of "corrections" representing the edges that should be removed from or added to the naïve reconstruction of the summary in order to obtain the exact graph. A different approach followed by Tian et al [32] and Liu et al [23], for graphs with labelled vertices, is to create "homogeneous" supernodes, i.e., to partition vertices so that vertices in the same set have, as much as possible, the same attribute values. These contributions are mostly focused on storing a highly compressed version of the graph. By contrast, our goal is to develop a summary that, while small enough to be stored in limited space (e.g., in main memory), can also be used to compute approximate but fast answers to queries about the original graph.

Toivonen et al [33] propose an approach for graph summarization tailored to weighted graphs, with the goal of creating a summary that preserves the distances between vertices. Fan et al [16] present two different summaries for classes of graph queries, one for reachability queries and one for graph patterns. These proposals are highly query-specific, while our summaries are general-purpose and can be used to answer different types of queries (see Sect. 5).

Graph summarization must not be confused with *graph clustering*, whose goal is to find groups of vertices that are highly connected within their group and are well-separated from other groups [30]. Graph summarization instead aims to group together vertices that have similar connection patterns with the rest of the graph.

Although graph summarization can be used for storing a concise representation of the graph, it is different from classic *graph compression* [8–10] which is concerned with developing algorithms and data structures to efficiently store and retrieve an *exact* representation of the graph using the minimum possible space. Despite the effectiveness achieved by existing methods for graph compression, these may not be sufficient to store the graph in main memory, and do not solve the problem of speeding up the computation of graph properties by analyzing a concise representation.

1.2 Contributions and Roadmap

In this work we study the graph summarization problem according to a definition that generalizes the one by LeFevre and Terzi [22], *devising the first polynomial-time approximation algorithms*. More in details, our contributions are as follows:

- In Sect. 2 we formalize the problem of graph summarization, and discuss several error measures: the ℓ_p*-reconstruction error* (previously employed by LeFevre and Terzi [22]) and the *cut-norm error*, a measure rooted in extremal graph theory [25] whose use we propose to address certain shortcomings of the ℓ_p errors.
- In Sect. 3, we draw a connection between graph summarization and geometric clustering (i.e., k-median/means) that allows us to develop the first algorithms to build k-summaries (summaries with k supernodes) of guaranteed quality according to the ℓ_1 and ℓ_2-reconstruction errors. Our algorithms compute a *constant-factor approximation* to the best k-summary and run in time roughly $n \cdot k$. Previously, only heuristics with unbounded approximation factors and high running times were known [22].
- In Sect. 4 we present the approximation algorithm for the cut-norm case. We exploit a connection between high-quality summaries that minimize the cut-norm error and *weakly-regular partitions* of a graph [17], a variant of the concept used in the Szemerédi Regularity Lemma [31]. The algorithm achieves a

polynomial approximation guarantee. Ours is also the first algorithm to find an approximation of the *best* weakly-regular k-partition.
- Several applications are discussed in Sect. 5. First we show that a summary with low cut-norm error provides approximation algorithms for graph partitioning problems such as *MaxCut* and *correlation clustering*. We then discuss the use of our summaries for storing a compressed (lossy or lossless) representation of the graph. Finally, we show how to use the summary to accurately answer important classes of queries about the original graph, including adjacency queries and triangle counting.
- An extensive experimental evaluation of the performance of our algorithms for summary construction and query answering, and a comparison with existing methods [22] complete our work (Sect. 6).

We believe that the connections with theoretical results that we exploit in this paper might foster further insights to solve important problems in graph mining. The reduction from approximate graph summarization to k-means/median clustering (Sect. 3) is of independent interest. One of the advantages of this approach is that it allows to leverage the large body of work on the latter topic; for example, one could use parallel implementations of k-means [7] to parallelize the summary construction. The Szemerédi Regularity Lemma [31] and its numerous variants (which we exploit in Sect. 4) can be a powerful addition to the toolkit of the graph mining algorithm designer.

2 Problem definition

We consider a simple, undirected[1] graph $G = (V, E)$ with $|V| = n$. In the rest of the paper, the key concepts are defined from the standpoint of the symmetric adjacency matrix A_G of G. For the sake of generality we allow the edges to be weighted (so the adjacency matrix is not necessarily binary) and we allow self-loops (so the diagonal of the adjacency matrix is not necessarily all-zero).

Graph summaries. Given a graph $G = (V, E)$ and $k \in \mathbb{N}$, a k-summary $\mathcal{S}$ of G is a *complete undirected weighted* graph $\mathcal{S} = (V', V' \times V')$ that is uniquely identified by a k-partition V' of V (i.e., $V' = \{V_1, \ldots, V_k\}$, s.t. $\cup_{i\in[1,k]} V_i = V$ and $\forall i, j \in [1, k], i \neq j$, it holds $V_i \cap V_j = \emptyset$). The vertices of $\mathcal{S}$ are called *supernodes*. There is a *superedge* e_{ij} for each unordered pair of supernodes (V_i, V_j), including (V_i, V_i) (i.e., each supernode has a self-loop e_{ii}). Each superedge e_{ij} has a weight, corresponding to the density of edges between V_i and V_j:

$$d_G(i, j) = d_G(V_i, V_j) = e_G(V_i, V_j)/(|V_i||V_j|),$$

where for any two sets of vertices $S, T \subseteq V$, we denote

$$e_G(S, T) = \sum_{i\in S, j\in T} A_G(i, j) \ .$$

Observe that if S and T overlap, each non-loop edge with both endpoints in $S \cap T$ is counted twice for $e_G(S, T)$.

[1] We discuss the case of directed graphs in Sect. 3.5.

We define the *density matrix* of $\mathcal{S}$ as the $k \times k$ matrix $A_{\mathcal{S}}$ with entries $A_{\mathcal{S}}(i,j) = d_G(i,j)$, $1 \le i,j \le k$. For each $v \in V$, we also denote by $s(v)$ the unique element w of $\mathcal{S}$ (i.e., a supernode) such that $v \in w$. The density matrix $A_{\mathcal{S}} \in \mathbb{R}^{k \times k}$ can be *lifted* to the matrix $A^{\uparrow}_{\mathcal{S}} \in \mathbb{R}^{n \times n}$ defined as

$$A^{\uparrow}_{\mathcal{S}}(v,w) = A_{\mathcal{S}}(s(v), s(w)) \ .$$

We justify the use of the lifted matrix in Sect. 3.1. Our lifted matrix is slightly different from the *expected adjacency matrix* defined by LeFevre and Terzi [22], which can also be computed from our summary (see Sect. 5). The difference affects the values of entries in diagonal blocks. In Section 3.4 we show that our algorithms also approximate the partition that minimizes the error from the expected adjacency matrix.

Partition-constant matrices. Given a k-partition $\mathcal{P} = \{S_1, \ldots, S_k\}$ of $[n]$, we say that a symmetric $n \times n$ matrix M with real entries is $\mathcal{P}$*-constant* if the $S_i \times S_j$ submatrix of M is *constant*, $1 \le i,j \le k$. More formally, M is $\mathcal{P}$-constant if for all pairs (i,j), $1 \le i,j \le k$, there is a constant $c_{ij} = c_{ji}$ such that $M(p,q) = c_{ij}$ for each pair (p,q) where $p \in S_i$ and $q \in S_j$. We also say that M is k*-constant*, to highlight the size of the partition. It should be clear from the definition that the lifted adjacency matrix of a k-summary $\mathcal{S}$ of a graph G is $\mathcal{P}_S$-constant for the partition $\mathcal{P}_{\mathcal{S}}$ of the nodes of G into the supernodes of $\mathcal{S}$.

An input graph, a possible summary, and the corresponding lifted matrix are exemplified in Figure 1 and Table 1.

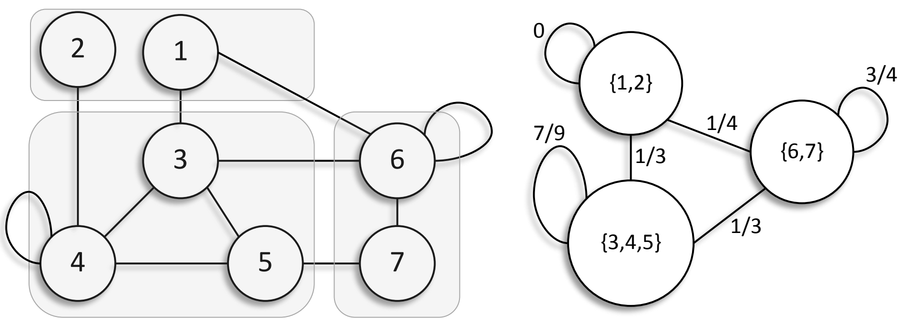

Fig. 1: A graph G (left) and one possible summary $\mathcal{S}$ (right).

Problem definition. The number of possible summaries is huge (there is one for each possible partitioning of V), so we need efficient algorithms to find the summary that best resembles the graph. This goal is formalized in Problem 1, which is the focus of this work.

Problem 1 (Graph Summarization) Given a graph $G = (V,E)$ with $|V| = n$, and $k \in \mathbb{N}$, find the k-summary $\mathcal{S}^*$, such that $A^{\uparrow}_{\mathcal{S}^*}$ minimizes the error $\mathsf{err}(A_G, A^{\uparrow}_{\mathcal{S}^*})$ for some error function $\mathsf{err} : \mathbb{R}^{n \times n} \times \mathbb{R}^{n \times n} \to [0, \infty)$.

	1	2	3	4	5	6	7
1	0	0	1/3	1/3	1/3	1/4	1/4
2	0	0	1/3	1/3	1/3	1/4	1/4
3	1/3	1/3	7/9	7/9	7/9	1/3	1/3
4	1/3	1/3	7/9	7/9	7/9	1/3	1/3
5	1/3	1/3	7/9	7/9	7/9	1/3	1/3
6	1/4	1/4	1/3	1/3	1/3	3/4	3/4
7	1/4	1/4	1/3	1/3	1/3	3/4	3/4

Table 1: The lifted matrix $A_{\mathcal{S}}^{\uparrow}$ corresponding to the $\mathcal{S}$ in Figure 1.

The function err expresses the dissimilarity between the original adjacency matrix of G and the lifted matrix obtained from the summary. Different definitions for err are possible and different algorithms may be needed to find the optimal summary $\mathcal{S}^*$ according to different measures. In the following section we present some of these measures and discuss their properties.

2.1 The ℓ_p-reconstruction error

Let $p \in \mathbb{R}$, $p \geq 1$. Given a graph G with adjacency matrix A_G and a summary $\mathcal{S}$ with lifted adjacency matrix $A_{\mathcal{S}}^{\uparrow}$, the *$\ell_p$-reconstruction error* of $\mathcal{S}$ is defined as the entry-wise p-norm of the difference between A_G and $A_{\mathcal{S}}^{\uparrow}$:

$$\mathsf{err}_p(A_G, A_{\mathcal{S}}^{\uparrow}) = \|A_G - A_{\mathcal{S}}^{\uparrow}\|_p = \left(\sum_{i=1}^{|V|}\sum_{j=1}^{|V|} |A_G(i,j) - A_{\mathcal{S}}^{\uparrow}(i,j)|^p\right)^{1/p} .$$

For example, consider the graph G and the summary $\mathcal{S}$ represented in Figure 1, whose lifted adjacency matrix is presented in Table 1. We have:

$$\mathsf{err}_1(A_G, A_{\mathcal{S}}^{\uparrow}) = \frac{329}{18} = 18.2\overline{7} \text{ and } \mathsf{err}_2(A_G, A_{\mathcal{S}}^{\uparrow}) = \sqrt{\frac{11844}{1296}} = 3.023059525 \ . \quad (1)$$

If A_G has entries in $[0,1]$ then $\mathsf{err}_p(A_G, A_{\mathcal{S}}^{\uparrow}) \in [0, n^{2/p}]$. Of special interest for us are the ℓ_1 and ℓ_2-reconstruction errors. These are very natural measures of the quality of a summary: e.g., the algorithms by LeFevre and Terzi [22] try to minimize the ℓ_1-reconstruction error.

Computational considerations. The ℓ_p norms can be computed in time $O(n^2)$ if $p = O(1)$. If the original graph $G = (V, E)$ is *unweighted*, it is possible to compute the ℓ_p-reconstruction errors in time $O(k^2)$ from the k-summary $\mathcal{S} = (V', V' \times V')$ itself. Indeed, given the partition $V' = \{V_1, \ldots, V_k\}$ of V, let $\alpha_{ij} = e_G(V_i, V_j)/(|V_i||V_j|)$ denote the superedge densities, for $i, j \in [k]$. A simple calculation shows that, for $p \geq 1$,

$$\mathsf{err}_p(A_{\mathcal{S}}^{\uparrow}, A_G)^p = \sum_{i,j \in [k]} |V_i||V_j|\alpha_{ij}(1-\alpha_{ij})\left((1-\alpha_{ij})^{p-1} + \alpha_{ij}^{p-1}\right) .$$

From this we also get that, for any summary $\mathcal{S}$,

$$\mathsf{err}_2(A^{\uparrow}_{\mathcal{S}}, A_G)^2 = 2 \cdot \mathsf{err}_1(A^{\uparrow}_{\mathcal{S}}, A_G) \ . \tag{2}$$

For example, this is the case for the errors reported in (1). Thus, a partition that minimizes the ℓ_2-reconstruction error also minimizes the ℓ_1-reconstruction error. A similar statement holds approximately (up to constant factors) for the ℓ_p and ℓ_q-reconstruction errors where $1 \leq p, q \leq O(1)$, so for unweighted graphs the exact choice of p is not crucial.

Discussion. Despite their naturality, these measures have some shortcomings. Consider an unweighted graph G. One can easily produce an uninteresting summary with only one supernode corresponding to V and ℓ_1-reconstruction error at most n^2. On the other hand, a low (say $o(n^2)$) ℓ_1-reconstruction error would imply that most pairs of vertices in any supernode have almost exactly the same neighborhoods in G: a summary with such low error would be very accurate and useful. Unfortunately, herein lies the main drawback of reconstruction error: for many graphs, k-summaries may achieve such a low reconstruction error only for very high values of k, often close to n.

As a simple idealized example, consider an Erdős-Renyi graph G drawn from the distribution $\mathcal{G}_{n,1/2}$, where each pair of vertices is linked by an edge with probability $1/2$. Given any two vertices in G, the size of the symmetric difference between their sets of neighbours is $\frac{n}{2} \pm \sqrt{n \log n}$ with high probability, and we incur in a $\frac{n}{2} - o(n)$ cost whenever we put two vertices into the same supernode. Hence, even for $k = n/2$, the reconstruction error of any k-summary must be as high as $n^2/9$.

As far as the input graph is not random, one may hope to find *structure* to exploit for building the summary. From this perspective it might be useful to think about the adjacency matrix A_G of the input graph G as a sum $A_G = S + E$ where S is a "structured" part captured by the summary, and E is a "residual" matrix representing the error of the summary. The "smaller" E is, the better the summary. In the next section we propose a measure that tries to quantify the residual error not represented by the summary. This measure is based on the *cut norm*, which plays an essential role in the theory of graph limits and property testing [25].

2.2 The cut-norm error

The *cut norm* of an $n \times m$ matrix A is the maximum absolute sum of the entries of any of its submatrices [17]:

$$\|A\|_{\square} = \max_{S,T \subseteq [n]} |A(S,T)| = \max_{S,T \subseteq [n]} \left| \sum_{i \in S, j \in T} A_{i,j} \right| \ .$$

The *cut distance* between two $n \times n$ matrices A and B is then defined by $\mathsf{err}_{\square}(A, B) = \|A - B\|_{\square}$. The *cut-norm error* of a summary $\mathcal{S}$ with respect to the graph G is therefore

$$\mathsf{err}_{\square}(A_G, A^{\uparrow}_{\mathcal{S}}) = \max_{S,T \subseteq V} |e_G(S,T) - e_{\mathcal{S}^{\uparrow}}(S,T)|,$$

which can be interpreted as the maximum absolute difference between the sum of weights of the edges between any two sets of vertices in the original graph and the same sum in the lifted summary. The cut norm is NP-hard to compute, but can be approximated up to a constant factor via SDP [3].

3 Summarization with reconstruction error

We now show a close connection between graph summarization and well-studied geometric clustering problems (k-median/means).

3.1 The best matrix for a given partition

We now justify the use of the ($\mathcal{P}$-constant) lifted adjacency matrix in our analysis. We ask the following question:
Given $p \geq 1$, a graph $G = (V, E)$ (without loss of generality, let $V = [n]$) with adjacency matrix A_G, and a partition $\mathcal{P} = \{S_1, \ldots, S_k\}$ of $[n]$, what $\mathcal{P}$-constant matrix $B^{*,p}_{\mathcal{P}}$ minimizes $\left\|A_G - B^{*,p}_{\mathcal{P}}\right\|_p$?

As we now show, the lifted adjacency matrix is a good (constant) approximation to the matrix that answers this question.

Clearly it suffices to consider each pair of supernodes S_i, S_j separately. For a fixed pair, let X be a random variable representing the weight (in G) of an edge (x, y) drawn uniformly at random from $S_i \times S_j$. We are looking for the real number a_p that minimizes $\mathbb{E}[|X - a_p|^p]$. The value a_p is known as the *p-predictor* of X. Then $B^{*,p}_{\mathcal{P}}(x, y)$ is exactly the p-predictor of the uniform distribution over the multiset $M(x, y) = \{A_G(v, w) \mid v \in s(x), y \in s(y)\}$, where $s(v)$ denotes the unique element of $\mathcal{P}$ (i.e., a set) to which v belongs. It is well-known that the 1-predictor of X is its median, and its 2-predictor is its expectation [36]. In other words,

$$
\begin{aligned}
B^{*,1}_{\mathcal{P}}(x, y) &= \mathrm{median}(\{A_G(v, w) \mid (v, w) \in s(v) \times s(w)\}) \\
B^{*,2}_{\mathcal{P}}(x, y) &= \sum_{(v,w) \in s(v) \times s(w)} A_G(v, w)/(|s(x)||s(y)|) \ ,
\end{aligned}
$$

where the argument to median is understood to be a multiset.

Note that $B^{*,2}_{\mathcal{P}} = A^{\uparrow}_{\mathcal{S}_{\mathcal{P}}}$, the lifted adjacency matrix of the summary $\mathcal{S}_{\mathcal{P}}$ corresponding to $\mathcal{P}$. In general, $B^{*,1}_{\mathcal{P}} \neq B^{*,2}_{\mathcal{P}}$, but we can still use $A^{\uparrow}_{\mathcal{S}_{\mathcal{P}}}$ for the case $p = 1$ because it still provides a good approximation to $B^{*,1}_{\mathcal{P}}$, as shown in the following lemma which is a corollary of Lemma 2.

Lemma 1 *We have:*

$$
\left\|A_G - B^{*,1}_{\mathcal{P}}\right\|_1 \leq \left\|A_G - B^{*,2}_{\mathcal{P}}\right\|_1 \leq 2 \cdot \left\|A_G - B^{*,1}_{\mathcal{P}}\right\|_1
$$

and

$$
\left\|A_G - B^{*,2}_{\mathcal{P}}\right\|_2 \leq \left\|A_G - B^{*,1}_{\mathcal{P}}\right\|_2 \leq \sqrt{2} \cdot \left\|A_G - B^{*,2}_{\mathcal{P}}\right\|_2 .
$$

Lemma 2 *Let X be a random variable with median m and expectation μ. Then*

$$\mathbb{E}[|X - m|] \leq \mathbb{E}[|X - \mu|] \leq 2 \cdot \mathbb{E}[|X - m|], \tag{3}$$

and

$$\mathbb{E}[|X - \mu|^2] \leq \mathbb{E}[|X - m|^2] \leq 2 \cdot \mathbb{E}[|X - \mu|^2] \ .$$

Proof The first inequality of each line follows from the fact that m is the 1-predictor and μ the 2-predictor. Now we bound the deviation between mean and median. Observe that

$$|\mu - m| = |\mathbb{E}[X] - m| = |\mathbb{E}[X - m]| \leq \mathbb{E}[|X - m|] \leq \mathbb{E}[|X - \mu|] \leq \sigma,$$

where $\sigma = \sqrt{\mathrm{Var}[X]}$ is the standard deviation of X and the last inequality is Cauchy-Schwarz. This yields the other two inequalities:

$$\mathbb{E}[|X - \mu|] = \mathbb{E}[|X - m + m - \mu|] \leq \mathbb{E}[|X - m|] + |m - \mu| \leq 2 \cdot \mathbb{E}[|X - m|],$$

and, since $\mathbb{E}[|X - \mu|^2] = \mathrm{Var}[X] = \sigma^2$,

$$\mathbb{E}[|X - m|^2] = \mathrm{Var}[X] + (m - \mu)^2 \leq 2 \cdot \mathbb{E}[|X - \mu|]^2 \ .$$

3.2 Connection with ℓ_p^p clustering

We now show how the graph summarization problem is related to the ℓ_p^p clustering problem. In the ℓ_p^p clustering problem, we are given n points $a_1, \ldots, a_n \in \mathbb{R}^d$ and we need to find k *centroids* $c_1, \ldots, c_k \in \mathbb{R}^d$ so as to minimize the *cost* function $\sum_n \left\|a_i - c_{l(k)}\right\|_p^p$, where $l(i)$ is the centroid closest to a_i in the ℓ_p metric. When $p = 2$, this is the k-means problem with ℓ_2 (Euclidean) metric; when $p = 1$, this is the k-median problem with ℓ_1 metric. We consider the *continuous* version in which the centroids are allowed to be arbitrary points in $\mathbb{R}^d$.

Any choice of centroids $c_1, \ldots, c_k$ gives rise to a partition $\mathcal{P}$ of $[n]$ that groups together points having the same closest centroid (assuming a scheme to break ties). Conversely, for any partition $\mathcal{P} = \{S_1, \ldots, S_k\}$ there is an optimal (i.e., minimizing the ℓ_p^p cost given $\mathcal{P}$) choice $c_1^*, \ldots, c_k^*$ of centroids: c_i^* is the coordinate-wise mean of the vectors in S_i when $p = 2$, and their coordinate-wise median when $p = 1$.

We show the following connections between clustering and summarization with respect to the ℓ_2 and ℓ_1-reconstruction error respectively.

Theorem 1 *Let $\bar{\mathcal{S}}$ be the k-summary induced by the partition of the rows of A_G with the smallest continuous ℓ_2^2 cost, and let $\mathcal{S}^*$ be the optimal k-summary for G with respect to the ℓ_2-reconstruction error. The ℓ_2-reconstruction error of $\bar{\mathcal{S}}$ is a 4-approximation to the best ℓ_2-reconstruction error:*

$$\mathsf{err}_2(A_G, A_{\bar{\mathcal{S}}}^{\uparrow}) \leq 4 \cdot \mathsf{err}_2(A_G, A_{\mathcal{S}^*}^{\uparrow}) \ .$$

Theorem 2 *Let $\hat{\mathcal{S}}$ be the k-summary induced by the partition of the rows of A_G with the smallest continuous ℓ_1 cost, and let $\mathcal{S}^\dagger$ be the optimal k-summary for G with respect to the ℓ_1-reconstruction error. The ℓ_1-reconstruction error of $\bar{\mathcal{S}}$ is an 8-approximation to the best ℓ_1-reconstruction error:*

$$\mathsf{err}_1(A_G, A_{\hat{\mathcal{S}}}^{\uparrow}) \leq 8 \cdot \mathsf{err}_1(A_G, A_{\mathcal{S}^\dagger}^{\uparrow}) \ .$$

Before we can prove these theorems we need some additional definitions and lemmas.

Smoothing projections and lifted matrices. Let $\mathcal{P} = \{S_1, \ldots, S_k\}$ be a partition of $[n]$ and let $\mathbf{s}_i$ be the n-dimensional vector associated to S_i such that the j^{th} entry of $\mathbf{s}_i$ is 1 if $j \in S_i$, and 0 otherwise. Write $\mathbf{v}_i = \mathbf{s}_i/\sqrt{|S_i|}$. Since $\|\mathbf{v}_i\| = 1$ and $S_i \cap S_j = \emptyset$ for $i \neq j$, the vectors $\{\mathbf{v}_i\}_{i\in[k]}$ are orthonormal. A sequence of vectors $\mathbf{v}_1, \ldots, \mathbf{v}_k \in \mathbb{R}^n$ is *partition-based* if they arise in this way from a partition of $[n]$. We say that a linear operator $P : \mathbb{R}^n \to \mathbb{R}^n$ is *smoothing* if it can be written as $P = \sum_{i=1}^k \mathbf{v}_i\mathbf{v}_i^\intercal$ for a partition-based set of vectors $\mathbf{v}_1, \ldots, \mathbf{v}_k$. Since $P^2 = P$, $P^\intercal = P$ and $P\mathbf{v}_i = \mathbf{v}_i$, it follows that P is the orthogonal projection onto the subspace generated by $\mathbf{v}_1, \ldots, \mathbf{v}_k$. It is also easy to check that a $n \times n$ matrix A is $\mathcal{P}$-constant if and only if $PAP = A$.

Given a k-summary $\mathcal{S}$ of G, let $\mathcal{P}_\mathcal{S} = \{S_1, \ldots, S_k\}$ be the partition of $[n]$ corresponding to $\mathcal{S}$. Consider the smoothing projection P arising from $\mathcal{P}_\mathcal{S}$ as described above.

Lemma 3 $A^{\uparrow}_{\mathcal{S}} = PA_GP$.

Proof Let $\mathbf{v}_1, \ldots, \mathbf{v}_k$ be the partition-based vectors arising from $\mathcal{P}_\mathcal{S}$. Recall that the entry $A^{\uparrow}_{\mathcal{S}_\mathcal{P}}(p, q)$ of the lifted adjacency matrix $A^{\uparrow}_{\mathcal{S}_\mathcal{P}}$, where $p \in S_i$ and $q \in S_j$, is the density $d_G(S_i, S_j) = \mathbf{s}_i^\intercal A_G\mathbf{s}_j/(|S_i||S_j|)$. Therefore

$$\begin{aligned}
A^{\uparrow}_{\mathcal{S}} &= \sum_{i,j\in[k]} d_G(S_i, S_j)\mathbf{s}_i^\intercal\mathbf{s}_j = \sum_{i,j\in[k]} \frac{\mathbf{s}_i^\intercal A_G\mathbf{s}_j}{|S_i||S_j|}\mathbf{s}_i^\intercal\mathbf{s}_j \\
&= \sum_{i,j\in[k]} (\mathbf{v}_i^\intercal A_G\mathbf{v}_j)\mathbf{v}_i^\intercal\mathbf{v}_j = \sum_{i,j\in[k]} \mathbf{v}_i^\intercal(\mathbf{v}_i^\intercal A_G\mathbf{v}_j)\mathbf{v}_j = PA_GP \enspace .
\end{aligned}$$

To prove Theorem 1 and Theorem 2 we also make use of the three following technical lemmas.

Lemma 4 *Let $P\colon \mathbb{R}^n \to \mathbb{R}^n$ be an orthogonal projection and let $\|\cdot\|$ denote a matrix norm that is* (1) *invariant under transposition and negation ($\|X\| = \|-X\| = \|X^\intercal\|$); and* (2) *contractive under P (for any $n \times n$ matrix X, $\|XP\| \leq \|X\|$). Then for any symmetric or skew-symmetric*[2] *matrix A, it holds that*

$$\frac{\|A - AP\|}{2} \leq \|A - PAP\| \leq 2\,\|A - AP\| \,.$$

Proof Using that $P^2 = P$ and the triangle inequality for $\|\cdot\|$, we have

$$\begin{aligned}
\|A - AP\| &= \|A - PAP + PAP - AP\| \\
&\leq \|A - PAP\| + \|PAP - AP\| = \|A - PAP\| + \|(PAP - A)P\| \\
&\leq \|A - PAP\| + \|PAP - A\| = 2\,\|A - PAP\| \enspace .
\end{aligned}$$

Observe that if A is symmetric ($A^\intercal = A$) then $(A - AP)^\intercal = A^\intercal - P^\intercal A^\intercal = A - PA$, whereas if A is skew-symmetric ($A^\intercal = A$) then $(A - AP)^\intercal = -(A - PA)$; either way, $\|A - AP\| = \|A - PA\|$. Therefore

$$\begin{aligned}
\|A - PAP\| &= \|A - AP + AP - PAP\| = \|A - AP\| + \|(A - PA)P\| \\
&\leq \|A - AP\| + \|A - PA\| = \|A - AP\| + \|A - AP\| = 2\,\|A - AP\| \enspace .
\end{aligned}$$

[2] A skew-symmetric matrix (also known as antisymmetric or antimetric matrix) is a square matrix A whose transpose is also its negative: $-A = A^\intercal$.

Lemma 5 *The ℓ_p norms ($p \geq 1$) satisfy the conditions of Lemma 4 for any smoothing projection P.*

Proof Invariance under transposition and negation is trivial, so we only need to check the second condition. To see it, write X by columns: $X = (x_1 \mid \cdots \mid x_n)$. Then $PX = (Px_1 \mid \cdots \mid Px_n)$ and $\|XP\|_p^p = \|PX\|_p^p = \sum_i \|Px_i\|_p^p$, so it suffices to show that $\|Py\|_p \leq \|y\|_p$ for $p = 1, 2$ and all $y \in \mathbb{R}^n$. The reader can verify that this follows from the power mean inequality:

$$\left(\frac{|\sum_{i=1}^m y_i|}{m}\right)^p \leq \frac{\sum_{i=1}^m |y_i|^p}{m} \quad .$$

The following result is an easy consequence of the definitions of smoothing projection and cost of a clustering (i.e., of a choice of centroids).

Lemma 6 *The ℓ_2^2 cost of the clustering associated with a partition $\mathcal{P}$ of the rows of a matrix $A \in \mathbb{R}^{n \times n}$ is $\|A - AP\|_2^2$, where P is the smoothing projection arising from $\mathcal{P}$.*

We are now ready to prove Theorems 1 and 2, showing the connection between summarization and geometric clustering.

Proof (Proof of Theorem 1) Let P denote a smoothing projection arising from an arbitrary k-partition. Let $P_{\bar{\mathcal{S}}}$ be the smoothing projection induced by $\bar{\mathcal{S}}$. By Lemma 6, $\|A_G - A_G P_{\bar{\mathcal{S}}}\|_2 \leq \|A_G - A_G P\|_2$. Let $P_{\mathcal{S}^*}$ be the smoothing projection associated with the partition which minimizes the ℓ_2-reconstruction error (i.e., the one induced by $\mathcal{S}^*$). Using Lemmas 4 and 5,

$$\begin{aligned}\mathsf{err}_2(A_G, A_{\bar{\mathcal{S}}}^{\uparrow}) &= \|A_G - P_{\bar{\mathcal{S}}} A_G P_{\bar{\mathcal{S}}}\|_2 \leq 2\,\|A_G - A_G P_{\bar{\mathcal{S}}}\|_2 \leq 2\,\|A_G - A_G P_{\mathcal{S}^*}\|_2 \\ &\leq 4\,\|A_G - P_{\mathcal{S}^*} A_G P_{\mathcal{S}^*}\|_2 = 4 \cdot \mathsf{err}_2(A_G, A_{\mathcal{S}^*}^{\uparrow}) \quad .\end{aligned}$$

This concludes the proof for Theorem 1.

The proof for Theorem 2 follows the same steps but Lemma 1 must be taken into account, which results in an additional factor of 2 in the approximation guarantee.

3.3 An efficient algorithm for summarization

Theorems 1 and 2 show that building a graph summary of guaranteed quality with regard to the ℓ_1 or ℓ_2-reconstruction error can be approximately reduced to solving a clustering instance. We now turn our attention to how to do this efficiently.

Both k-median and k-means are NP-hard [2, 13, 26], but admit constant-factor approximation algorithms that run in time polynomial in the number of points (n), clusters (k), and the dimension of the space where the points reside [6, 20, 27]. In order to use these algorithms for our purposes, we need to take care of the following bottlenecks: costly pairwise distances computation, high dimensionality, and large number of points. Exact computation of all pairwise distances between the rows of the adjacency matrix can be rather expensive: in the ℓ_2 norm, computing all distances between n points in $\mathbb{R}^d$ is equivalent to multiplying a matrix with its transpose.[3] We can avoid this by using approximate distances computed efficiently from a *sketch* of the adjacency matrix, i.e., a matrix with the same number of rows but a logarithmic number of columns [19];[4] this also reduces the number of dimensions from n to $O(\log n)$. In exchange for this speedup, the analysis needs to factor in the additional error, and the fact that the approximate distances we work with will not satisfy the triangle inequality, whereas the clustering algorithms we use are designed for metric spaces.

The $O(1)$-approximation algorithm by Mettu and Plaxton [27] can be used with the approximate distances computed from the sketch, but it runs in time $\widetilde{O}(n^2)$. To improve this we use a result by Aggarwal et al [1] which adaptively selects $O(k)$ of the rows of the sketch so that the optimal k-median/means solution obtained by clustering these rows gives a set of centers that can be used to obtain a constant-factor approximation to the clustering problem for all the rows. By running the algorithm of Mettu and Plaxton [27] on the resulting $O(k) \times O(\log n)$ matrix, we obtain a constant factor approximation in time $\widetilde{O}(m + nk)$, where m is the number of edges (or half the number of non-zero entries in A_G, if G is weighted). We formalize this intuition in Theorem 3, while Algorithm 1 presents the pseudocode.

Algorithm 1: Graph summarization with ℓ_p-reconstruction error

Input : $G = (V, E)$ with $|V| = n$, $k \in \mathbb{N}$, $p \in \{1, 2\}$
Output: A $O(1)$-approximation to the best k-summary for G under the ℓ_p-reconstruction error
// Create the $n \times O(\log n)$ sketch matrix [19]
$S \leftarrow$ `createSketch`$(A_G, O(\log n), p)$
// Select $O(k)$ rows from the sketch [1]
$R \leftarrow$ `reduceClustInstance`(A_G, S, k)
// Run the approximation algorithm by Mettu and Plaxton [27] to obtain a partition.
$\mathcal{P} \leftarrow$ `getApproxClustPartition`(p, k, R, S)
// Compute the densities for the summary
$D \leftarrow$ `computeDensities`$(\mathcal{P}, A_G)$
return $(\mathcal{P}, D)$

Theorem 3 *Let $p \in \{1, 2\}$. There exists an algorithm to compute an $O(1)$-approximation to the best k-summary under the ℓ_p-reconstruction error in time $\widetilde{O}(m + nk)$ with high constant probability.*

Proof **Approximate distance computation.** Indyk [19] gives a small-space streaming algorithm for maintaining sketches of vectors so as to allow approximate computation of their ℓ_p distance, where $p \in (0, 2)$. It is shown in [19, Theorem 2] that, for $p \in \{1, 2\}$ and $\epsilon, \delta \in (0, 1)$, one can sketch n-dimensional vectors whose entries are bounded by poly(n) so that:

[3] If $v_1, \ldots, v_n \in \mathbb{R}^d$, then $\|v_i - v_j\|_2^2 = \|v_i\|_2^2 + \|v_j\|_2^2 - 2\langle v_i, v_j\rangle$. Since the quantities $\|v_i\|_2^2$ can be easily precomputed, the problem reduces to computing all inner products $\langle v_i, v_j\rangle$. These form the entries of $AA^\intercal$, where A is the $n \times d$ matrix with rows $v_1, \ldots, v_n$.

[4] For ℓ_2, we can also use the Johnson-Lindenstrauss transform [21].

– The sketch $C(v)$ of $v \in \mathbb{R}^n$ uses space $t = O(\log(n/(\delta\epsilon))\log(1/\delta)/\epsilon^2)$ and can be computed in time $O(1+t \cdot nnz(v))$, where $nnz(v)$ is the number of non-zero entries of v.
– Given $C(v)$ and $C(w)$, one can estimate $\|v-w\|_p$ up to a factor of $1+\epsilon$ with probability $1-\delta-1/n^3$ in time $O(t)$.

We compute the sketch of all the rows of A_G with $\epsilon = 1$ and $\delta = 1/(200n^2)$. The result is an $n \times t$-sized sketched matrix S, where $t = O(\log n)$. The running time of this step is $O(n + t \cdot nnz(A_G)) = \widetilde{O}(m+n)$. For every pair of rows of A_G, we can use S to estimate their ℓ_p distance up to a factor of 2 with probability $1-1/(200n^2)-1/n^3$ in time $O(\log n)$. Hence the estimation is good simultaneously for all pairs of rows with probability $0.99 - 1/n$.

Approximate pseudometrics. Now let d_{ij} denote the ℓ_p distance between rows i and j of A_G, and let $\tilde{d}_{ij}$ denote their approximation based on the sketch S. The true distances d_{ij} give rise to a pseudometric on the rows of A_G. (They are not necessarily a metric as d_{ij} may be zero for $i = j$ if we have two identical rows in A_G.) On the other hand, the approximate distances do not satisfy the triangle inequality, which may pose a problem for algorithms for metric k-median. Nevertheless, using the fact that $\frac{\tilde{d}_{ij}}{d_{ij}} \in [1/2, 2]$, one may verify that they form a 4-approximate pseudometric, i.e., they satisfy the following axioms with $\lambda = 4$:

– $\tilde{d}_{ij} \geq 0$
– $\tilde{d}_{ii} = 0$,
– $\tilde{d}_{ij} = \tilde{d}_{ji} = 0$,
– $\tilde{d}_{ij} \leq \lambda(\tilde{d}_{ik} + \tilde{d}_{kj})$.

The p-th powers of the distances in a λ-approximate pseudometric yield a $\lambda^p 2^{p-1}$-approximate pseudometric; this is a consequence of the arithmetic mean-geometric mean inequality. Moreover, the largest ratio between two non-zero distances between rows of A_G is bounded above by $\mathrm{poly}(n)$. Taking these observations together we conclude that the numbers $\tilde{d}_{ij}^p$ give rise to an $O(1)$-approximate pseudometric with distance ratio $\mathrm{poly}(n)$. Under this condition, most known metric k-median/k-means algorithms, including the ones detailed below, guarantee a constant-factor approximation [1, 20].

Clustering. Aggarwal et al [1, Theorems 1 & 4] show that given a k-means or k-median clustering instance S in which distance computations can be performed in time $O(t)$, one can select a subset X of $r = O(k)$ points from S and compute a weight assignment $w : X \to \mathbb{R}$ such that

– With 99% probability, if $C \in \binom{X}{k}$[5] is an α-approximate minimizer of

$$\rho(C) = \sum_{x \in X} \min_{c \in C} w(c) \cdot \|x - c\|_p^p ,$$

then C is also an $O(\alpha)$-approximate minimizer of

$$\rho'(C) = \sum_{x \in S} \min_{c \in C} \|x - c\|_p^p .$$

– X and w can be computed in time $O(nkt \log n)$.

[5] We denote as $\binom{X}{k}$ the set of k-subsets of X, i.e., the subsets of X of size k.

Here we say that $C \in \binom{X}{k}$ is an α-approximate minimizer of function $f : \binom{X}{k} \to \mathbb{R}^+$ if $f(C) \leq \alpha \cdot f(C')$ for all $C' \in \binom{X}{k}$.

This means that one can solve a *weighted* k-median instance with $O(k)$ points, and derive from it a nearly optimal solution to the original n-point instance by keeping the same set of centers and assigning every other point to its closest center; the last step takes time $O(nkt)$. To solve the smaller instance, we may use the algorithm by Mettu and Plaxton [27], which generalizes to weighted k-median in a straightforward manner. The running time of [27] on an instance with $O(k)$ points and an approximate pseudometric with distance ratio r is $O(k^2 + k \log r)$. Hence we obtain a $O(1)$-approximation to the clustering of the $O(k)$ rows of S (where $r = \mathrm{poly}(n)$) in time $O(nk(\log n)^2 + k^2 + kn + nk \log n) = \widetilde{O}(nk)$.

Finally, observe that the distance approximation properties of $\tilde{d}_{ij}$ guarantee that the optimal clustering of the rows of S is also a $O(1)$-approximation to the optimal clustering of the rows in the original adjacency matrix, so an $O(1)$-approximation to the former is a $O(1)$-approximation to the latter. Given the partition defined by this nearly-optimal clustering, we can then compute the densities in time $O(m + k^2) = O(m + nk)$, completing the proof.

3.4 Relationship with the expected adjacency matrix

Let $\mathcal{P}$ be a k-partition with associated smoothing projection P (see Section 3.2). Let us assume A is the adjacency matrix of a loopless, unweighted graph and let $B = PAP$. Let C be the expected adjacency matrix of A with respect to $\mathcal{P}$, as defined by LeFevre and Terzi [22]. Note that $C_{ij} = B_{ij}$ whenever i and j belong to different supernodes.

Denote by n_i the size of the ith supernode of $\mathcal{P}$, and by e_i the number of internal edges within the supernode. Write $\alpha_i = 2e_i/(n_i(n_i - 1)) \in [0, 1]$. Direct computation reveals that

$$|B - A|_1 = |C - A|_1 + \sum_{i=1}^{k} 2(n_i - 1)\alpha_i^2,$$

implying

$$|C - A|_1 \leq |B - A|_1 \leq |C - A|_1 + 2n - k.$$

It follows that if C^* is the optimal solution to the problem of finding a k-partition whose expected adjacency matrix minimizes the ℓ_1 reconstruction error with A, and $\hat{B}$ is a constant-factor approximation to the problem of finding a k-partition whose lifted matrix minimizes the ℓ_1 reconstruction error with A, then

$$|\hat{B} - A|_1 \leq O(|C^* - A|_1) + O(n).$$

That is, the partition returned by our method is, up to a small additive term of $O(n)$, within a constant factor of optimal in the sense of LeFevre and Terzi [22].

3.5 Summaries for directed graphs

For directed graphs we employ *two* summaries rather than one. We decompose the adjacency matrix A into the sum of a symmetric matrix $B = (A + A^\intercal)/2$ and a skew-symmetric matrix $C = (A - A^\intercal)/2$. Our algorithms then build k-summaries $\mathcal{S}_B$ and $\mathcal{S}_C$ for B and C respectively. Since the quality guarantees of our algorithms also apply to skew-symmetric matrices, $(\mathcal{S}_B, \mathcal{S}_C)$ is a high-quality summary for the directed graph. Indeed, suppose we lift these summaries to obtain two matrices $B^\uparrow_{\mathcal{S}_B}$ and $C^\uparrow_{\mathcal{S}_C}$ and let $\hat{A}$ be the sum of $B^\uparrow_{\mathcal{S}_B}$ and $C^\uparrow_{\mathcal{S}_C}$, then $\mathsf{err}(A, \hat{A}) \leq \mathsf{err}(B, B^\uparrow_{\mathcal{S}_B}) + \mathsf{err}(C, C^\uparrow_{\mathcal{S}_C})$.

4 Summarization with the cut-norm error

One of the cornerstone results in graph theory is the Szemerédi regularity lemma [31]: the vertex set of *any* graph can be partitioned into classes of approximately the same size in such a way that the number of edges between vertices in different classes behaves almost randomly. Such a partition can be found in polynomial time [4] and would be a powerful basis for a graph summary, offering equally-sized parts and well-behaved densities. Unfortunately, the number of classes can be astronomically large [18]. Therefore, we focus on *weak regularity*, a variant which requires substantially smaller partitions [17].

4.1 Weak regularity and index

We now introduce two important concepts: *weak regularity* (related to the cut norm) and *index of a partition* (related to the reconstruction error).

Given a graph $G = (V, E)$, a partition $\mathcal{P} = \{V_1, \ldots, V_k\}$ of V is ε*-weakly regular* if the cut-norm error of the lifted adjacency matrix of the summary induced by $\mathcal{P}$ is at most εn^2 [17]. Equivalently, if for any pair of sets $S, T \subseteq V$, we have

$$\left| e(S,T) - \sum_{i=1}^{k} \sum_{j=1}^{k} d_G(V_i, V_j) |S \cap V_i||T \cap V_j| \right| \leq \varepsilon n^2.$$

Any unweighted graph with average degree $\alpha = 2|E|/|V|^2 \in [0,1]$ admits an ε-weakly regular partition with $2^{O(\alpha \log(1/\alpha)/\varepsilon)^2}$ parts [17]. This bound is tight [12].

Note, however, that any particular graph of interest may admit much smaller partitions. Dellamonica et al [14, 15] proved that the optimal deterministic algorithm to build a weakly ε-regular partition with at most $2^{1/\mathrm{poly}(\varepsilon)}$ parts takes time $c(\varepsilon)n^2$.

The *index* of a partition $\mathcal{P} = \{V_1, \ldots, V_k\}$ is defined as

$$\mathsf{ind}(\mathcal{P}) = \sum_{i,j \in [k]} \frac{|V_i||V_j|}{n^2} d_G^2(V_i, V_j) = \mathbb{E}_{u,v \in V}[d_G^2(s(u), s(v))].$$

The index is nonnegative and is bounded by the average density of the graph. It is used as a potential function measuring progress towards a regular partition in all proofs of the standard regularity lemma. Interestingly, on unweighted graphs the sum of the index of $\mathcal{P}$ and the normalized squared ℓ_2-reconstruction error of $\mathcal{P}$ is exactly the average density of the graph. Accordingly, finding the k-partition that maximizes the index is equivalent to finding the k-summary whose lifted adjacency matrix minimizes the ℓ_1-reconstruction error, implying a close relationship between the two problems.

4.2 Relationship with clustering

To the best of our knowledge, the problem of constructing a partition of a given size k that achieves the smallest possible cut-norm error for a given input graph G has not been considered before. Existing algorithms [17] take an error bound ε and always produce a partition of size $2^{\Theta(1/\varepsilon^2)}$, irrespective of whether much smaller partitions with the same error exist for G.

Our algorithm to build a quasi-optimal summary that minimizes the cut-norm error works by computing the k-median clustering of the columns of the *square* of the adjacency matrix. After squaring the adjacency matrix, it follows the same steps as the one we presented in Sect. 3.3 (see also Alg. 2).

We rely on the following result of Lovász:

Theorem 4 (Theorem 15.32[25]) *Consider the following distance function δ between nodes of a graph $G=(V,E)$ with $|V|=n$ and adjacency matrix A_G:*

$$\delta(u,v)=\frac{\sum_{w\in V}|A_G^2(u,w)-A_G^2(v,w)|}{n^2} .$$

1. *If $\mathcal{P}=\{S_1,\ldots,S_k\}$ is weakly ε-regular for G, then we can select a node v_i from each class S_i such that the cost of the k-median solution $\{v_1,\ldots,v_k\}$ according to δ is at most 4ε.*
2. *If the k-median cost of a set of centroids $S=\{v_1,\ldots,v_k\}\subseteq V$ according to δ is ε, then the clustering induced by S forms a weakly $8\sqrt{\varepsilon}$-regular partition for G.*

We then have the following result connecting the cost (error) of the weakly-regular partition induced by the optimal k-median solution according to the δ distance function to the cost of the optimal (minimum error) weakly-regular partition with k classes.

Corollary 1 *Let OPT_p be the cost (error) of the optimal (minimum error) weakly-regular partition of V with k classes. The clustering induced by the optimal k-median solution forms a partition that is a weakly-regular partition with cost at most $16/\sqrt{\mathsf{OPT}_p}$.*

Proof Let OPT_c be the cost of the optimal k-median solution. We know from Theorem 4, point 1, that $\mathsf{OPT}_c\leq 4\mathsf{OPT}_p$. Then

$$\frac{8\sqrt{\mathsf{OPT}_c}}{\mathsf{OPT}_p}\leq\frac{8\sqrt{4\mathsf{OPT}_p}}{\mathsf{OPT}_\mathsf{p}}\leq\frac{16}{\sqrt{\mathsf{OPT}_p}} .$$

After squaring the adjacency matrix, we can proceed as in Section 3.3. Algorithm 2 presents the pseudocode for summarization w.r.t. the cut-norm error. The following is an immediate consequence.

Theorem 5 *Let ε^* be the smallest ε such that there is a weakly ε-regular k-partition of G. Then we can find an $O(\sqrt{\varepsilon^*})$-weakly regular partition in time $O(n^\omega)$.*

Here $\omega < 2.3727$ stands for the time bound in matrix multiplication [37]; we assume $\omega > 2$. The time complexity bound is dominated by the cost of squaring the matrix and does not depend on ε^*. We observe that one could find a partition with the same guarantee in randomized time $O(n^2 \cdot \mathrm{poly}(1/\varepsilon^*))$ by performing a search on ε^* and using sampling to estimate the distances $\delta(u, v)$ up to $\mathrm{poly}(\varepsilon^*)$ additive accuracy.

Algorithm 2: Graph summarization with cut-norm error

Input : Graph $G = (V, E)$ with $|V| = n$, integer $k > 0$
Output: An approximation to the best k-summary for G under the cut-norm error

```
// Square the adjacency matrix
Q ← A_G^2
// Create the n × O(log n) sketch matrix [19]
S ← createSketch(Q, O(log n), 1)
// Select O(k) rows from the sketch [1]
R ← reduceClustInstance(Q, S, k)
// Run the approximation algorithm [27] to obtain a partition.
𝒫 ← getApproxClustPartition(1, k, R, S)
D ← computeDensities(𝒫, A_G)
return (𝒫, D)
```

5 Using the summary

In this section we show how our summaries can be used *(1)* in partitioning problems, *(2)* for compression, and *(3)* to accurately answer queries about the original graph.

5.1 Partitioning

By definition, a summary with low cut-norm error (i.e, a weakly regular partition of a graph) gives good approximations to the sum of the weights of the edges between *any* pair of vertex sets. This helps devising approximation algorithms for certain graph partitioning problems by working on the summary. For example, Frieze and Kannan [17] give a PTAS for MaxCut restricted to dense graphs, by finding a small weakly regular partition, and then using brute force to find the optimal weighted max cut of the summary, which can then be translated back into a nearly maximum cut in the original graph. A similar technique has been recently applied to devise sublinear algorithms for *correlation clustering* [11].

5.2 Compression

The summary, seen as a partition of the vertices and a collection of densities, can be used as a lossy compressed representation of the graph. It is described by a partition of the vertices into supernodes and a collection of densities. The partition (a function from $[n]$ to $[k]$) can be stored in $O(n \log k)$ bits. The densities are determined by the total weight of edges between each pair of sets, which takes $O(k^2 \ell \log n)$ bits, where ℓ is the number of bits required to store each edge weight. Therefore the number of bits required to describe a summary is $O(n \log k + k^2 \ell \log n)$.[6]

One can also compute a *lossless* compressed representation of the original graph. The idea is to use the summary as a *model* and to encode the data, i.e., the original graph, given the model (see [22]). The number of additional of bits to encode the data for unweighted graphs is

$$\sum_{i,j} \left(-A_G(i,j) \log_2 A_S^{\uparrow}(i,j) - (1 - A_G(i,j)) \log_2 \left(1 - A_S^{\uparrow}(i,j)\right)\right) .$$

5.3 Query answering

Due to the lossy nature of summary, given a summary, there exists a number of *possible worlds*, i.e., possible graphs, that might have originated that summary. Therefore, following [22] we adopt an *expected-value semantics* for approximate query answering: the answer to a query on the summary is the expectation of the exact answer over all graphs that may have resulted in that summary, considered all equally likely under the principle of indifference. In particular, LeFevre and Terzi [22] define an *expected adjacency matrix* $\bar{A}$ which is slightly different from the lifted matrix $A_S^{\uparrow}$ we defined in Sect. 2 but can be computed from it as follows[7]:

- If two vertices i and j belong to different supernodes in the summary, then $\bar{A}(i,j) = A_S^{\uparrow}(i,j)$.
- If i and j belong to the same supernode S_ℓ, and $i \neq j$, then $\bar{A}(i,j) = A_S^{\uparrow} \cdot |S_\ell|/(|S_\ell| - 1)$.
- If $i = j$, then $\bar{A}(i,j) = 0$.

Under the expected-value semantics, computing the answers to many important class of queries is straightforward. For instance, the *existence probability of an edge* (u,v) (or is expected weight, in case of weighted graphs) is $\bar{A}(u,v)$. The *weighted degree* of v is $\sum_{i=1}^{n} \bar{A}(v,i)$. Similarly, the *weighted eigenvector centrality* can be expressed as $\sum_{i=1}^{n} \bar{A}(v,i)/2|E|$.

It is worth remarking that the average error of adjacency queries is the ℓ_1-reconstruction error, while the average error of degree queries is always bounded by the ℓ_1-reconstruction error divided by n. Hence in these cases it is easy to prove worst-case bounds on the average error incurred when computing the answer from the summary.

We next show how to answer more complex queries involving the number of triangles.

The problem of counting the number of triangles (cycles of size 3) in a graph and more generally of computing the distribution of connected subgraphs is a fundamental problem in graph analysis [34]. These quantities are useful to understand

[6] Further space-saving can be achieved by storing only densities above a certain threshold using adjacency lists; the superedges removed increase the reconstruction error.

[7] Minor modifications are needed if self-loops are allowed.

the structure of the graph, to compute additional properties, and as a fingerprint of the graph. Using our graph summary for answering triangle counting queries in the expected-value semantics is straightforward. Let n_i be the number of vertices in the i-th supernode and let π_{ij} be defined as follows for $1 \leq i, j \leq k$: $\pi_{ij} = d_{ij}$ if $i \neq j$, and $\pi_{ij} = \frac{d_{ij} n_i}{n_i - 1}$ if $i = j$.

Lemma 7 *The expected number of triangles is*

$$\mathbb{E}[\triangle] = \sum_{i=1}^{k} \left(\binom{n_i}{3} \pi_{ii}^3 + \sum_{j=i+1}^{k} \left(\pi_{ij}^2 \left(\binom{n_i}{2} n_j \pi_{ii} + \binom{n_j}{2} n_i \pi_{jj} \right) + \right.\right.$$
$$\left.\left. + \sum_{w=j+1}^{k} n_i n_j n_w \pi_{ij} \pi_{jw} \pi_{wj} \right) \right) . \tag{4}$$

It can be computed in time $O(k^3)$.

Proof The thesis follows from repeated applications of the linearity of expectation.

The expected number of triangles is the sum of the expected numbers of three groups of triangles, which differ by the distribution of their vertices across the supernodes in the summary:

Group 1 triangles with all three vertices in the same supernode;
Group 2 triangles with two vertices in one supernode and one vertices in a different one;
Group 3 triangles with vertices in three different supernodes.

From linearity of expectation, we have that the expected number of triangles in each group is the sum of the expectations of the number of triangles in each possible choice of supernodes compatible with the definition of the group: single supernodes for group 1, pairs of supernodes for group 2, and triplets of supernodes for group 3.

The expected number of triangles within each choice of supernodes is the sum of the expectations over the possible choices of three vertices from the chosen supernodes (according to the group definition). The number of choices of three vertices depends on the number of vertices in each chosen supernodes and on the group definition.

For each triplet of nodes we can model the presence of a triangle using a Bernoulli random variable that takes value one if the triplet forms a triangle. The expectation of this random variable is clearly the product of the quantities π_{ij} (or π_{ii}) between the supernodes which the vertices of the corresponding triangle belong to.

By summing all the terms we obtain (4). It is immediate to see that the complexity is $O(k^3)$, which only depends on the number of supernodes in the summary.

The same approach can be used to develop formulas for the expected distribution of subgraphs of any size. Care must be taken to avoid counting the same occurrence of a subgraph multiple times due to isomorphisms. We can also use the summary to count the expected number of triangles a given vertex belongs to.

Corollary 2 *Let v be a vertex and let, without loss of generality, V_1 be the supernode it belongs to. The expected number of triangles that v belongs to is*

$$\pi_{11}^3 \binom{|V_1|-1}{2} + \sum_{j=2}^{k} (\pi_{1j}|V_j| (\pi_{11}\pi_{1j}(|V_i|-1)+ \\ + \pi_{jj}\pi_{1j}\frac{|V_j|-1}{2} + \sum_{w=j+1}^{k} \pi_{1w}\pi_{jw}|V_w|)) \ . \tag{5}$$

The *triangle density* of a graph is the ratio between the number of triangles in the graph over the number of triplets of vertices, independently of their connectivity. The results above allow us also to compute the expected triangle density from the summary.

Corollary 3 *Let $\mathbb{E}[\triangle]$ be the expected number of triangles from (4). Then the expected triangle density is*

$$\frac{6\mathbb{E}[\triangle]}{n(n-1)(n-2)} \ . \tag{6}$$

6 Experimental Evaluation

In this section we report the results of our experimental evaluation which has the following goals:

1. to characterize the structure of the summaries built by our algorithms;
2. to evaluate the quality of the summaries in terms of the reconstruction errors and the cut-norm error and of their usefulness in answering queries;
3. to compare the performances of our algorithms with those of `GraSS` [22].

Datasets and implementations We used real graphs from the SNAP repository[8] and the iRefIndex protein interaction network[9] (Table 2).

As we considered all the graphs *unweighted and undirected*, the ℓ_1-reconstruction error is half the *squared* ℓ_2-reconstruction error (see (2)), hence we only report the results for the ℓ_2-reconstruction error (divided by n for normalization) and the cut-norm error (divided by n^2).

We use two variants of the k-median/k-means clustering procedure at the core of our methods: the constant-factor approximation algorithm by Arya et al [6] and the classic Lloyd's iterative approach [24] with `k-means++` initialization that guarantees an $O(\log k)$ approximation factor [5]. We denote the different variants as follows: "`S2A`" is the algorithm for the ℓ_2-reconstruction error using the approximation algorithm by Arya et al [6][10], "`S2L`" uses Lloyd's algorithm for the k-median step for ℓ_2-reconstruction error, and "`SCL`" does the same for the cut-norm error. There is no "`SCA`" variant because the approximation algorithm would

[8] `http://snap.stanford.edu/data/`

[9] `http://irefindex.org`

[10] For speed reasons, we modified the algorithm by Arya et al [6] to try only a limited number of local improvements and did not run it to completion. It could otherwise achieve even better approximations.

require computing the fourth power of the adjacency matrix, which is too costly. Computing the exact cut-norm error is very expensive so we estimate it using a sample of the graph and the approximation algorithm by Alon and Naor [3]. Our implementations do not use dimensionality reduction: we tested it and found that it brings little to no performance improvement for the very sparse graphs that we consider. Our algorithms are implemented[11] in C++11 and the experiments are performed on a 4-core AMD Phenom II X4 955 with 16GB of RAM running GNU/Linux 3.12.9. Each algorithm is run 5 times for each combination of parameters and input graph.

6.1 Summary characterization

We studied the structure of the summaries created by our algorithms in terms of the distribution of the sizes of the supernodes, the distributions of the internal and cross densities, the (reconstruction or cut-norm) error of the generated summaries, and the running time of our algorithms. We report the results for `S2L` in Table 2. The behaviors for `S2A` are extremely similar and not reported. The behavior for the cut-norm error is reported in Table 3.

Supernode size In Table 2 and Table 3 we do not report the minimum size since this was always 1 in all cases. This is interesting: in order to minimize the errors it may actually be convenient to create a supernode containing a single vertex. Nevertheless there are also large supernodes containing hundreds or thousands of vertices, which helps explain the relatively large standard deviation. As k grows, the standard deviation shrinks faster than the average size (n/k), suggesting that supernode sizes become more uniform. The behavior is similar for the two errors, but we can see that the distribution is actually different, reflecting the different requirements.

Density The minimum internal density was 0 in all our tests, as a consequence of aforementioned fact that there are supernodes of size 1 and that the graphs had no self-loops. On the other hand, there are supernodes whose corresponding induced subgraphs are quite dense, almost cliques (a clique would correspond to a value of 100 in the “max” column). The minimum and maximum cross densities are not reported in Tables 2 and 3 because they were respectively 0 and 1 in all cases. While the latter fact is expected from the presence of supernodes of size 1, the former suggests that some supernodes are effectively *independent* from each other, i.e., there are no edges connecting them. The cross densities are very small but their distribution has a large standard deviation, a fact related to the presence of cross-densities equal to 1. Both internal and cross densities decrease as k grow.

Reconstruction errors The ℓ_2-reconstruction error in Table 2 shrinks *linearly* as k grows. This was very consistent across the five runs for all cases: standard deviation (not reported) was less than 10^{-4}. Interestingly, for fixed k, the *normalized* error becomes smaller as the size of the graph grows.

[11] The implementation is available from `http://cs.brown.edu/~matteo/graphsumm.tar.bz2`.

		Size		Internal Density ($\times 10^2$)			Cross Density ($\times 10^2$)		ℓ_2-rec. err. ($\times 10^2$)	Time (s)	
Graph	k	stdev	max	avg	stdev	max	avg	stdev	avg	avg	stdev
Facebook	500	29.99	719	41.43	32.34	97.14	1.77	11.49	6.56	1.17	0.00
$\lvert V\rvert = 4\,039$	750	19.00	556	34.77	31.86	94.72	1.65	11.42	6.18	1.85	0.00
$\lvert E\rvert = 88\,234$	1000	15.84	597	28.59	31.22	94.79	1.56	11.42	5.81	2.67	0.02
	1250	11.09	382	23.52	29.64	95.15	1.44	11.18	5.42	3.53	0.01
	1500	8.77	206	19.72	28.02	94.18	1.37	11.07	5.01	4.48	0.01
Iref	1000	203.44	6542	7.40	20.05	96.29	0.88	8.58	3.07	8.22	0.18
$\lvert V\rvert = 12\,231$	2000	118.33	5848	4.54	15.64	93.75	0.57	7.17	2.73	15.03	0.08
$\lvert V\rvert = 281\,903$	3000	76.66	4568	2.79	12.13	90.9	0.41	6.18	2.41	23.27	0.09
	4000	51.78	3475	1.84	9.73	85.71	0.31	5.43	2.08	33.18	0.11
	5000	34.78	2485	1.17	7.64	85.71	0.23	4.79	1.78	44.11	0.10
Enron	6000	81.58	6817	15.56	26.88	87.5	0.22	4.62	0.96	115.3	0.21
$\lvert V\rvert = 36\,692$	8000	56.78	5325	14.08	25.76	87.5	0.15	3.82	0.83	172.88	0.36
$\lvert E\rvert = 183\,831$	10000	42.10	4041	12.58	24.52	87.5	0.1	3.27	0.72	253.1	15.66
	12000	27.57	2635	11.16	23.24	88.88	0.08	2.86	0.63	305.38	20.67
	14000	23.98	2398	9.77	21.77	87.5	0.06	2.54	0.54	349.31	17.25
Epinions1	7500	434.60	38015	3.27	13.13	90.00	0.3	5.45	0.81	384.35	0.69
$\lvert V\rvert = 75\,879$	10000	332.45	34596	2.71	11.82	90.00	0.21	4.52	0.73	525.82	0.59
$\lvert E\rvert = 405\,740$	12500	265.58	31070	2.23	10.6	88.88	0.15	3.85	0.66	675.15	1.09
	15000	208.43	26026	1.90	9.74	87.50	0.11	3.35	0.60	870.80	1.04
Sign-epinions	7500	934.11	87464	3.41	13.54	92.30	0.42	6.31	0.67	683.9	9.8
$\lvert V\rvert = 131\,828$	10000	781.22	79371	2.70	12.03	90.90	0.30	5.41	0.61	901.21	3.21
$\lvert E\rvert = 711\,210$	12500	642.54	72588	2.25	10.92	88.88	0.22	4.67	0.57	1140.23	3.86
	15000	543.04	67884	1.91	10.01	90.00	0.17	4.10	0.53	1449.68	2.91
Stanford	2000	2481.49	113572	28.05	32.57	97.95	0.08	2.73	0.48	389.57	19.10
$\lvert V\rvert = 281\,903$	4000	1355.57	94216	25.90	31.94	97.61	0.05	2.29	0.44	616.23	51.23
$\lvert E\rvert = 1\,992\,636$	6000	1007.47	83444	24.25	31.52	97.61	0.04	1.98	0.42	970.74	9.04
	8000	834.61	73622	22.56	31.09	97.61	0.03	1.84	0.40	1230.16	47.52
	10000	658.67	65659	21.32	30.63	97.61	0.03	1.70	0.38	1604.85	81.47
Amazon0601	2000	7479.31	351920	37.79	28.91	90.9	0.01	0.90	0.53	1921.29	76.42
$\lvert V\rvert = 403\,394$	4000	5006.09	323770	37.53	29.2	90.9	0.00	0.81	0.52	2646.85	49.12
$\lvert E\rvert = 2\,443\,408$	6000	3766.88	306673	36.97	29.69	90.9	0.00	0.76	0.52	3419.72	76.94
	8000	3053.54	278468	36.78	29.99	90.9	0.00	0.73	0.51	4215.82	33.32

Table 2: Distributions of supernode size, internal and cross densities, normalized ℓ_2-reconstruction error, and runtime for summaries built with S2L. The average supernode size is n/k by definition. The minimum supernode size was always 1. The minimum internal density was always 0. The minimum and maximum cross densities were respectively always 0 and 1.

Cut-norm error Again, the cut-norm error (SCL algorithm) shrinks approximately linearly as k grows. We attribute the slight deviation from linearity to the fact that we are not computing the exact cut-norm error, but rather an approximation.

Runtime The running time grows approximately linearly with k, but for larger graphs and larger k its standard deviation across the five runs grows. By looking at the leftmost four columns of Table 4 we can see that, as expected, S2A builds summaries with a slightly smaller ℓ_2-reconstruction error than S2L but takes between 3 to 9 times longer. The times for SCL are much higher because squaring the original adjacency matrix results in a much denser matrix. The high standard deviation of the running times for SCL are also due to the approximation algorithm involved in the computation of the cut-norm error.

Graph	k	Size stdev	Size max	Internal Density ($\times 10^2$) avg	Internal Density ($\times 10^2$) stdev	Internal Density ($\times 10^2$) max	Cross Density ($\times 10^2$) avg	Cross Density ($\times 10^2$) stdev	cut-norm err. ($\times 10^4$) avg	Time (s) avg	Time (s) stdev
Facebook	500	18.90	351	35.05	29.90	96.17	1.53	9.78	24.98	3.86	0.09
	750	12.34	221	30.85	29.42	93.43	1.57	10.59	20.69	6.19	0.09
	1000	9.39	203	26.20	28.72	93.50	1.51	10.78	19.70	9.24	0.14
	1250	7.06	146	22.18	27.58	93.50	1.46	10.93	19.22	12.54	0.20
	1500	5.87	141	18.82	26.53	93.07	1.37	10.85	18.12	16.56	0.05
Iref	1000	184.73	6995	0.12	2.11	50	0.37	5.80	8.57	107.21	3.46
	2000	111.81	6026	0.07	1.64	50	0.38	5.97	8.07	254.31	4.59
	3000	87.90	5807	0.07	1.66	50	0.35	5.81	6.43	380.01	70.89
	4000	20.86	1230	0.06	1.52	50	0.25	4.84	5.71	545.87	104.04
	5000	19.60	1863	0.05	1.47	50	0.21	4.51	4.79	581.29	177.52
Enron	6000	65.63	3912	3.54	13.27	87.50	0.22	4.63	1.49	1211.59	16.76
	8000	60.22	8339	4.36	14.84	87.50	0.16	3.89	1.69	1908.63	487.16
	10000	37.88	2735	5.32	16.38	87.50	0.11	3.26	1.61	2121.09	10.60
	12000	30.17	2456	6.12	17.60	87.50	0.08	2.84	1.88	2572.83	10.96
	14000	25.21	2631	6.77	18.48	87.50	0.07	2.53	2.16	3019.85	15.64
Epinions1	7500	234.57	29872	0.11	2.03	75	0.27	5.12	0.71	6306.92	1494.41
	10000	154.25	14924	0.13	2.43	83.33	0.20	4.41	0.70	8640.02	2054.76
	12500	178.53	30719	0.16	2.71	85.71	0.16	3.92	0.53	11169.40	2883.92
	15000	119.30	16597	0.18	2.79	83.33	0.12	3.42	0.55	16030.62	3694.68
Sign-epinions	7500	645.35	57295	0.11	2.15	66.67	0.35	5.86	0.55	17543.27	564.95
	10000	541.35	48479	0.09	1.81	75	0.26	5.02	0.51	25235.54	676.87
	12500	449.69	46588	0.10	2.05	91.67	0.20	4.46	0.48	33078.94	871.03
	15000	428.44	49180	0.11	2.26	92.97	0.17	4.14	0.35	41377.98	472.26
Amazon0601	2000	3983.13	217197	6.58	17.88	89.26	0.01	0.76	0.36	6466.76	1925.52
	4000	2740.82	188554	6.67	17.89	90.00	0.01	0.79	0.38	6687.60	1397.96
	6000	2083.37	203282	6.66	18.16	90.91	0.01	0.76	0.39	10229.73	4640.58
	8000	1656.28	145817	6.58	18.08	90.91	0.01	0.73	0.47	18176.86	3035.96

Table 3: Distributions of supernode size, internal and cross densities, normalized cut-norm reconstruction error, and runtime for summaries built with SCL. The average supernode size is n/k by definition. The minimum supernode size was always 1. The minimum internal density was always 0. The minimum and maximum cross densities were respectively always 0 and 1.

6.2 Query answering

We evaluated the performances of the summaries in answering queries about the structure of the graph. In Table 5 we report *(1)* the absolute error for adjacency queries, corresponding to the ℓ_1-reconstruction error; *(2)* the absolute degree error, and *(3)* the *relative* triangle density error (defined as $(\mathsf{expected} - \mathsf{exact})/\mathsf{exact}$). Results for the very large graphs are not available because computing the query error would require running the query on the original graph, and this takes an excessive amount of time (indeed, this is one of the motivations for our work). We remark that the performances on answering degree (resp. triangle density) queries are equivalent to those of answering eigenvector centrality queries (resp. triangle counting queries). In general, as expected, a decrease in k corresponds to an often-substantial increase in the query answer error. For adjacency queries, the average error (which is exactly the ℓ_1-reconstruction error) is very small, almost 0, and indeed the error was 0 for many pairs of vertices. We found though that the maximum error could be large in some rare case. This can happen when a vertex v is the only one in its supernode to have an edge to a vertex u in another supernode: if one

	S2L			S2A			SCL			
	ℓ_2-rec. err.	Time (s)		ℓ_2-rec. err.	Time (s)		Cut-norm err		Time (s)	
k	avg ($\times 10^2$)	avg	stdev	avg ($\times 10^2$)	avg	stdev	avg ($\times 10^2$)	stdev ($\times 10^2$)	avg	stdev
1000	3.07	8.22	0.18	2.89	70.18	3.33	0.08	0.01	107.21	3.46
2000	2.73	15.03	0.08	2.53	86.13	2.4	0.06	0.00	254.31	4.59
3000	2.41	23.27	0.09	2.19	102.97	3.35	0.05	0.01	380.01	70.89
4000	2.08	33.18	0.11	1.88	93.42	1.83	0.05	0.01	545.87	104.04
5000	1.78	44.11	0.1	1.59	106.75	2.04	0.04	0.01	581.29	177.52

Table 4: Errors (normalized) and runtime as function of k for Iref. Standard deviations for the ℓ_2-rec. errors are not reported because insignificant (smaller than 10^{-4} for errors of order 10^{-2}).

		Error in Query Answering				
		Adjacency ($\times 10^2$)		Degree		
Graph	k	avg	stdev	avg	stdev	Clust. Coeff.
Facebook	500	0.42	4.57	7.14	10.43	-0.31
	750	0.37	4.32	6.22	9.15	-0.28
	1000	0.33	4.05	5.38	7.96	-0.24
	1250	0.28	3.79	4.79	7.27	-0.19
	1500	0.24	3.49	4.01	6.31	-0.15
Iref	1000	0.09	2.15	6.23	11.25	-0.47
	2000	0.07	1.92	4.53	8.18	-0.32
	3000	0.05	1.68	3.27	6.1	-0.20
	4000	0.04	1.46	2.39	4.5	-0.13
	5000	0.03	1.26	1.68	3.38	-0.07
Enron	4000	0.01	0.78	2.57	4.95	-0.32
	6000	< 0.01	0.66	1.92	3.53	-0.20
	8000	< 0.01	0.57	1.49	2.65	-0.13

Table 5: Error in query answering for summaries built with S2L. For adjacency and degree queries we report the absolute error, while for the clustering coefficient we report the relative error.

or both supernodes are large, the cross density is very small, but the error in the adjacency query involving v and u is large, since they are the only connected pair. This has obviously an impact on the standard deviation that is substantially larger than the average. For degree queries, the average error is small, when compared to the average degree $2|E|/|V|$ and to the ratio between the ℓ_1-reconstruction error and n (which is an upper bound to the average degree error). The standard deviation of the error shows that it is also quite concentrated. As for clustering coefficients (i.e., triangle counting), the estimations obtained from the summary are of good quality when the ratio between the number of vertices in the graph and the number of supernodes is not too large, but grows rapidly otherwise. This is to be expected: the number of triangles (and therefore the clustering coefficient) is particularly sensitive to loss of information due to summarization. Note that we always underestimate the clustering coefficient because real-world networks have many more triangles than random graphs.

		Error in Query Answering				
		Adjacency ($\times 10^2$)		Degree		
Graph	k	avg	stdev	avg	stdev	Clust. Coeff.
Facebook	500	0.54	5.18	7.81	20.46	-0.47
	750	0.47	4.82	6.07	17.90	-0.40
	1000	0.41	4.50	5.06	14.66	-0.34
	1250	0.35	4.17	4.27	14.14	-0.28
	1500	0.30	3.83	3.53	10.19	-0.23
Iref	1000	0.12	2.40	7.35	20.45	-0.90
	2000	0.10	2.24	5.79	18.15	-0.77
	3000	0.08	1.98	4.23	16.06	-0.57
	4000	0.07	1.80	3.64	12.38	-0.50
	5000	0.05	1.56	2.55	8.91	-0.34
Enron	4000	0.02	0.94	3.32	14.11	-0.57
	6000	0.01	0.81	2.37	9.87	-0.38
	8000	0.01	0.71	1.75	8.16	-0.26

Table 6: Error in query answering for summaries built with `SCL`. For adjacency and degree queries we report the absolute error, while for the clustering coefficient we report the relative error.

6.3 Comparison with GraSS

We compared the runtime and the summary quality of `S2A` with those of the GraSS `k-GS-SamplePairs` algorithm [22] (which we refer to as "`GS`"). An implementation of `GS` from the original authors was not available, therefore we implemented it in C++11 and optimized it as much as we could in order to be able to perform a fair comparison. `GS` was originally presented as a method to build summaries by heuristically minimizing the ℓ_1-reconstruction error using the *expected adjacency matrix*, rather than the *lifted matrix* from the summary. Given the close similarity between the two, we adapted `GS` to use the lifted matrix and extended it to minimize the ℓ_2-reconstruction error. We did not pursue the adaptation of `GS` to the cut-norm error due to fact that `GS` would perform multiple computations of the cut-norm error per step, incurring in an exceedingly high cost. In order to keep the running time of `GS` within reasonable limits, we used sampled-down versions of the graphs obtained with a "forest-fire" sampling approach. In Table 7 we report the results for a sample of 500 vertices and 3969 edges of the `ego-gplus` networks. Note that `GS` takes a parameter c to quantify the number of sampled pair candidates for merging per step. We used $c \in \{0.10, 0.5, 1.0\}$. We did not use higher values for c due to the excessive running time of `GS` for high values of this parameter. It is possible to appreciate that `S2A` is several orders of magnitude faster than `GS` (which runs in $O(n^4 \cdot c)$), and its error is always smaller. In fact, due to the internal workings of `GS`, its running time is roughly independent of k unless k is close to n.

In conclusion, *our algorithm not only gives guarantees on the approximation of the optimum solution but it also builds summaries with smaller reconstruction error, much faster.*

k	Alg.	c (for GS)	ℓ_2-reconstr. Err.	Runtime (s)
10	GS	0.1	0.168	495.122
		0.5	0.155	2669.961
		0.75	0.153	4401.213
		1.0	0.153	5516.915
	S2A		0.152	0.440
25	GS	0.1	0.156	494.666
		0.5	0.145	2669.619
		0.75	0.143	4400.456
		1.0	0.142	5515.247
	S2A		0.140	0.742
50	GS	0.1	0.146	495.518
		0.5	0.136	2671.848
		0.75	0.133	4407.631
		1.0	0.133	5527.319
	S2A		0.131	0.695
100	GS	0.1	0.130	495.074
		0.5	0.120	2669.013
		0.75	0.117	4396.178
		1.0	0.116	5508.125
	S2A		0.115	0.708
250	GS	0.1	0.081	462.085
		0.5	0.072	2517.515
		0.75	0.070	4192.601
		1.0	0.069	5263.651
	S2A		0.065	0.552

Table 7: Comparison between S2A and GS on a random sample ($n = 500$) of ego-gplus (averages over five runs).

7 Conclusions

This work provides the first approximation algorithms to build quasi-optimal summaries with guaranteed quality according to various error measures. The algorithms exploit a novel connection between graph summarization and the k-median and k-means problems. One of the measures introduced (the *cut-norm error*) is particularly interesting and relates high-quality summaries to weakly-regular partitions, an concept connected to the Szemerédi regularity lemma. We believe that the this connection with theoretical results, can be a powerful addition to the algorithm designer toolkit for other problems in graph analysis and mining.

Acknowledgements Matteo Riondato was supported in part by NSF grant IIS-1247581, NIH grant R01-CA180776, and by a summer internship at Yahoo Labs Barcelona.

References

1. Aggarwal A, Deshpande A, Kannan R (2009) Adaptive sampling for k-means clustering. In: APPROX-RANDOM, pp 15–28

2. Aloise D, Deshpande A, Hansen P, Popat P (2009) NP-hardness of Euclidean sum-of-squares clustering. Mach Learn 75(2):245–248, DOI 10.1007/s10994-009-5103-0, URL `http://dx.doi.org/10.1007/s10994-009-5103-0`
3. Alon N, Naor A (2006) Approximating the cut-norm via Grothendieck's inequality. SIAM J Comput 35(4):787–803
4. Alon N, Duke RA, Lefmann H, Rödl V, Yuster R (1994) The algorithmic aspects of the regularity lemma. J of Algorithms 16(1):80–109
5. Arthur D, Vassilvitskii S (2007) k-means++: the advantages of careful seeding. In: Proc. of 18th SODA, pp 1027–1035
6. Arya V, Garg N, Khandekar R, Meyerson A, Munagala K, Pandit V (2004) Local search heuristics for k-median and facility location problems. SIAM J Comput 33(3):544–562
7. Bahmani B, Moseley B, Vattani A, Kumar R, Vassilvitskii S (2012) Scalable k-means++. PVLDB 5(7):622–633
8. Boldi P, Vigna S (2004) The webgraph framework I: compression techniques. In: WWW, ACM, pp 595–602
9. Boldi P, Santini M, Vigna S (2009) Permuting web and social graphs. Internet Mathematics 6(3):257–283
10. Boldi P, Rosa M, Santini M, Vigna S (2011) Layered label propagation: A multiresolution coordinate-free ordering for compressing social networks. In: WWW, ACM, pp 587–596
11. Bonchi F, García-Soriano D, Kutzkov K (2013) Local correlation clustering, under submission
12. Conlon D, Fox J (2012) Bounds for graph regularity and removal lemmas. Geom Func Anal 22(5):1191–1256, DOI 10.1007/s00039-012-0171-x, URL `http://dx.doi.org/10.1007/s00039-012-0171-x`
13. Dasgupta S (2008) The hardness of k-means clustering. Tech. Rep. 09-16, University of California, San Diego
14. Dellamonica DJ, Kalyanasundaram S, Martin DM, Rödl V, Shapira A (2012) A deterministic algorithm for the Frieze–Kannan regularity lemma. SIAM J Disc Math 26(1):15–29, DOI 10.1137/110846373, URL `http://epubs.siam.org/doi/abs/10.1137/110846373`, `http://epubs.siam.org/doi/pdf/10.1137/110846373`
15. Dellamonica DJ, Kalyanasundaram S, Martin DM, Rödl V, Shapira A (2015) An optimal algorithm for finding Frieze–Kannan regular partitions. Combinatorics, Probability and Computing 24:407–437, DOI 10.1017/S0963548314000200, URL `http://journals.cambridge.org/article_S0963548314000200`
16. Fan W, Li J, Wang X, Wu Y (2012) Query preserving graph compression. In: SIGMOD, ACM, pp 157–168
17. Frieze A, Kannan R (1999) Quick approximation to matrices and applications. Combinatorica 19(2):175–220
18. Gowers WT (1997) Lower bounds of tower type for Szemerédi's uniformity lemma. Geom Func Anal 7(2):322–337
19. Indyk P (2006) Stable distributions, pseudorandom generators, embeddings, and data stream computation. J ACM 53(3):307–323
20. Jain K, Vazirani VV (2001) Approximation algorithms for metric facility location and k-median problems using the primal-dual schema and Lagrangian relaxation. J ACM 48(2):274–296

21. Johnson WB, Lindenstrauss J (1984) Extensions of Lipschitz mappings into a Hilbert space. Contemp Math 26:189–206
22. LeFevre K, Terzi E (2010) GraSS: Graph structure summarization. In: SDM, SIAM, pp 454–465
23. Liu Z, Yu JX, Cheng H (2012) Approximate homogeneous graph summarization. J Inf Proc 20(1):77–88
24. Lloyd S (1982) Least squares quantization in PCM. IEEE Trans Inf Theor 28(2):129–137, DOI 10.1109/TIT.1982.1056489, URL http://dx.doi.org/10.1109/TIT.1982.1056489
25. Lovász L (2012) Large networks and graph limits, vol 60. American Mathematical Society
26. Megiddo N, Supowit KJ (1984) On the complexity of some common geometric location problems. SIAM J Comput 13(1):182–196, URL http://dblp.uni-trier.de/db/journals/siamcomp/siamcomp13.html#MegiddoS84
27. Mettu RR, Plaxton CG (2003) The online median problem. SIAM J Comput 32(3):816–832
28. Navlakha S, Rastogi R, Shrivastava N (2008) Graph summarization with bounded error. In: SIGMOD, pp 419–432
29. Riondato M, García-Soriano D, Bonchi F (2014) Graph summarization with quality guarantees. In: ICDM
30. Schaeffer SE (2007) Graph clustering. Computer Science Review 1(1):27–64
31. Szemerédi E (1976) Regular partitions of graphs. In: Problèmes Combinatoires et Théorie des Graphes, Colloq. Internat. CNRS, Univ. Orsay., pp 399–401
32. Tian Y, Hankins RA, Patel JM (2008) Efficient aggregation for graph summarization. In: SIGMOD, ACM, pp 567–580
33. Toivonen H, Zhou F, Hartikainen A, Hinkka A (2011) Compression of weighted graphs. In: KDD, ACM, pp 965–973
34. Tsourakakis CE (2008) Fast counting of triangles in large real networks without counting: Algorithms and laws. In: ICDM, IEEE, pp 608–617
35. Ward JH (1963) Hierarchical grouping to optimize an objective function. J Amer Statist Assoc 58(301):236–244
36. Williams D (1991) Probability with Martingales. Cambridge University Press, URL http://dx.doi.org/10.1017/CBO9780511813658, cambridge Books Online
37. Williams VV (Nov 2011) Breaking the Coppersmith-Winograd barrier, unpublished manuscript