Separating Structure from Noise in Large Graphs Using the Regularity Lemma

Marco Fiorucci[a,*], Francesco Pelosin[a], Marcello Pelillo[a,b]

[a]*DAIS, Ca Foscari University, Via Torino 155, Venice, Italy*
[b]*ECLT, Ca Foscari University, S. Marco 2940, Venice, Italy*

Abstract

How can we separate structural information from noise in large graphs? To address this fundamental question, we propose a graph summarization approach based on Szemerédi's Regularity Lemma, a well-known result in graph theory, which roughly states that every graph can be approximated by the union of a small number of random-like bipartite graphs called "regular pairs". Hence, the Regularity Lemma provides us with a principled way to describe the essential structure of large graphs using a small amount of data. Our paper has several contributions: (i) We present our summarization algorithm which is able to reveal the main structural patterns in large graphs. (ii) We discuss how to use our summarization framework to efficiently retrieve from a database the top-k graphs that are most similar to a query graph. (iii) Finally, we evaluate the noise robustness of our approach in terms of the reconstruction error and the usefulness of the summaries in addressing the graph search task.

1. Introduction

Recent years are characterized by an unprecedented quantity of available network data which are produced at an astonishing rate by an heterogeneous variety of interconnected sensors and devices. This high-throughput generation calls for the development of new effective methods to store, retrieve, understand and process massive network data. To tackle this challenge, we introduce a framework for summarizing large graphs based on *Szemerédi's Regularity Lemma*[1], which roughly states that any sufficiently large graph can almost entirely be partitioned into a bounded number of random-like bipartite graphs, called *regular pairs*. The partition resulting from the Regularity Lemma gives rise to a summary, called *reduced graph*, which inherits many of the essential structural properties of the original graph [2, 3].

---

*Corresponding author
Email address:* `marco.fiorucci@unive.it` (Marco Fiorucci)

In this paper, we posit that the Regularity Lemma can be used to summarize large graphs revealing its main structural patterns, while filtering out noise, which is common in any real-world networks. In its original form, the lemma is an existential predicate, but during the last decades various constructive algorithms have been proposed [4, 5, 6]. However, despite being polynomial in the size of the underlying graph, all these algorithms have a hidden tower-type dependence on an accuracy parameter. To overcome this limitation, in the last years we have proposed some simple heuristics that, most of the times, allowed us to construct a regular partition [7, 8, 9].

The main contribution of this paper is the introduction of a new heuristic algorithm which is characterized by an improvement of the summary quality both in terms of reconstruction error and of noise filtering. In particular, we first build the reduced graph of a graph G, and then we "blow-up" the reduced graph to obtain a graph G', called *reconstructed graph*, which is close to G in terms of the l_p-reconstruction error. We study the noise robustness of our approach in terms of the reconstruction error by performing an extensive series of experiments on both synthetic and real-world data. As far as the synthetic data are concerned, we generate graphs with a cluster structure, where the clusters are perturbed with different levels of noise. As far as the real-world data are concerned, we add spurious edges in accord with different noise probabilities. The aim of this series of experiments is to assess if the framework is able to separate structure from noise. In the ideal case, the distance between G and G' should be only due to the filtered noise.

Moreover, in the second part of the paper, we use our summarization algorithm to address the *graph search* problem defined under a similarity measure. The aim of graph search is to retrieve from a database the top-k graphs that are most similar to a query graph. Since noise is common in any real-world dataset, the biggest challenge in graph search is developing efficient algorithms suited for dealing with large graphs containing noise in terms of missing and adding spurious edges. In our approach, all the graphs contained in a database are compressed off-line, while the query graph is compressed on-line. Thus, graph search can be performed on the summaries, and this allows us to speed up the search process and to reduce storage space. Finally, we evaluate the usefulness of our summaries in addressing the graph search problem by performing an extensive series of experiments. In particular, we study the quality of the answers in terms of the found top-k similar graphs, and the scalability both in the size of the database and in the size of the query graphs.

Related Works. The first contribution of our paper is the introduction of a principled framework for summarizing large graphs with the aim of preserving their main structural patterns. Previous related works presented methods which mainly built summaries by grouping the vertices into subsets, such that the vertices within the same subset share some topological properties. The works in [10, 11] introduced methods for partitioning the vertices into non-overlapping clusters, so that vertices within the same cluster are more connected than vertices belonging to different clusters. A graph summary can be constructed by considering each cluster as a *supernode*, and by connecting each pair of supernodes with a *superedge* of weight equals to the sum of the cross-cluster edges. However, since graph summarization and clustering have different goals, this approach is suited only if the input graph has a strong community structure. In [12], the summary is generated by greedily grouping vertices, such that the normalized reconstruction error

between the adjacency matrix of the input graph and the adjacency matrix of the *reconstructed graph* is minimized. Since in their work they exploited heuristic algorithms, they can not give any guarantees on the quality of the summary. The work in [13] proposed a method of building a summary with quality guaranty by minimizing the l_p-*reconstruction error* between the adjacency matrix of the input graph and the adjacency matrix of the reconstructed graph. Since both approaches aim to minimize a distance measure between the input and the reconstructed graph, they are not the best choice for summarizing noisy graphs. By contrast, our goal is to develop a graph summarization algorithm which is robust against noise. For a more detailed picture on how the field has evolved previously, we refer the interested reader to the survey of Liu et al. [14].

The second contribution of our paper consists in addressing the graph search problem using the proposed summarization framework. Locating the occurrences of a query graph in a large database is a problem which has been approached in two main different ways, based on subgraph isomorphism and approximate graph matching respectively. Ullman [15] put one of the first milestones in subgraph isomorphism. He proposed an algorithm which decreases the computational complexity of the matching process by reducing the search space with backtracking. Recently, Carletti et al. [16] introduced an algorithm for graph and subgraph isomorphism which scales better than Ullmann's one. In particular, Carletti et al's algorithm, which may be considered as the state-of-the-art in exact subgraph matching, can process graphs of size up to ten thousand nodes. However, since subgraph isomorphism is a NP-complete problem, the algorithms based on exact matching are prohibitively expensive for querying against a database which contains large graphs. Moreover, due to the noise contained in any real-world networks, it is common to mismatch two graphs which have the same structure but different levels of noise. Indeed, these contributions are focused on exact matching and, even if they proposed efficient solutions, they are not noise robust. By contrast, our goal is to develop an efficient graph search algorithm which is robust against noise. Hence, approaches based on approximate graph matching are more suitable for addressing the graph search problem. Indeed, in this category lies the most effective graph similarity search algorithms. Most of the time, the searching phase is conducted under the *graph edit distance* (GED) constraint [17, 18, 19]. The graph edit distance $GED(g_1, g_2)$ is defined as the minimum number of edit operations (adding, deletion and substitution) that modify g_1 step-by-step to g_2 (or vice versa). In [18] and in [20], the authors underline the robustness of GED against noise due to its error-tolerant capability. Unfortunately, the GED computation is NP-hard, and most existing solutions adopt a *filtering-verification* technique. In particular, first, a pruning strategy is used to filter out false positive matches, and then the remaining candidates are verified by computing GED. In this context, the work of Liang and Zhao [17] represents the state-of-the-art. They provided a partition-based GED lower bound to improve the filter capability, and a multi-layered indexing approach to filter out false positives in an efficient way. Their algorithm can deal with databases with a high number of graphs, but cannot handle large graphs due to the complexity of GED computation. Instead, our algorithm is designed to scale both in the size of the databases and in the size of the graphs.

Roadmap. The paper is organized as follows. In section 2, we provide the basic concepts and notations used in the next sections as well as the formal definition of graph summary. In section 3, we present a short theoretical description of Szemerédi's Regularity Lemma
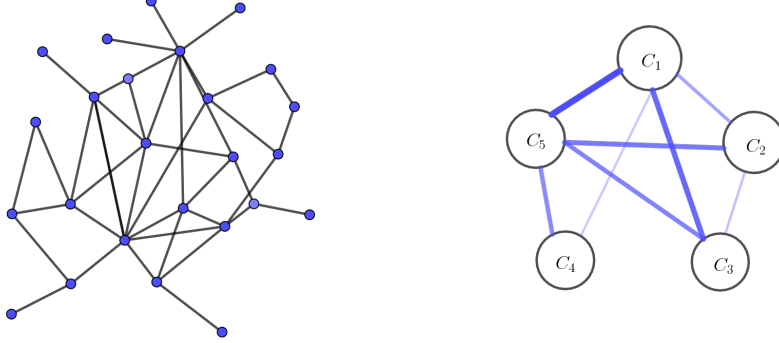
Figure 1: Example of the reduced graph (summary) construction. Left: the original graph. Right: the *reduced graph* which contains eight ε-regular classes pairs. The density of each pair is expressed by the thickness of the edge that connects the classes of that pair. If a pair is ε-irregular the corresponding classes are not connected by an edge.

and we describe how to find a regular partition. Section 4 is devoted to the description of our summarization framework, while in section 5, we introduce the formal definition of graph search problem, and we discuss how to use our framework to speed up the search process and to reduce the storage space. In section 6, we report an extensive experimental evaluation of the noise robustness of our approach, in terms of the reconstruction error, and of the usefulness of the summaries in addressing the graph search task. Finally, we drawn our conclusions in section 7.

2. Preliminaries

Let $G = (V, E)$ be an undirected graph without self-loops. The *edge density* of a pair of two disjoint vertex sets $C_i, C_j \subseteq V$ is defined as:

$$d(C_i, C_j) = \frac{e(C_i, C_j)}{|C_i|\,|C_j|} \tag{1}$$

where $e(C_i, C_j)$ denotes the number of edges of G with an endpoint in C_i and an endpoint in C_j.

Given a positive constant $\varepsilon > 0$, we say that the pair (C_i, C_j) of disjoint vertex sets $C_i, C_j \subseteq V$ is *ε-regular* if for every $X \subseteq C_i$ and $Y \subseteq C_j$ satisfying $|X| > \varepsilon\,|C_i|$ and $|Y| > \varepsilon\,|C_j|$ we have

$$|d(X, Y) - d(C_i, C_j)| < \varepsilon \;. \tag{2}$$

This means that the edges in an ε-regular pair are distributed fairly uniformly, where the deviation from the uniform distribution is controlled by the tolerance parameter ε.

DEFINITION 1 (ε-regular partition). A partition $\mathcal{P} = \{C_0, C_1, \ldots, C_k\}$, with C_0 being the exceptional set[1] is called *ε-regular* if:

---

[1]C_0 has only a technical purpose: it makes it possible that all other classes have exactly the same number of vertices.

1. it is equitable: $|C_1| = |C_2| = \cdots = |C_k|$;

2. $|C_0| < \varepsilon|V|$;

3. all but at most εk^2 of the pairs (C_i, C_j) are ε-regular, $(1 \leq i < j \leq k)$.

In this paper, we propose a summarization algorithm which, given an undirected graph $G = (V, E)$, iteratively builds a summary, called *reduced graph* (Figure 1), defined as follows.

DEFINITION 2 (Reduced graph). Given an ε-regular partition $P = \{C_1, C_2, \ldots, C_k\}$ of a graph $G = (V, E)$ and $0 \leq d' \leq 1$, the reduced graph of G is the undirected weighted graph $R = (V_R, E_R, w)$, where $V_R = P$, $E_R \subseteq V_R^2$ and $w : E_R \to \mathbb{R}$ is defined as follows:

$$w((C_i, C_j)) = \begin{cases} d(C_i, C_j) & \text{if } (C_i, C_j) \text{ is } \varepsilon\text{-regular and } d(C_i, C_j) \geq d', \\ 0 & \text{otherwise.} \end{cases}$$

We are now ready to state the Regularity Lemma which provides us a principled way to develop a summarization algorithm with the aim of separating structure from noise in a large graph.

3. Szemerédi's Regularity Lemma

In essence, Szemerédi's Regularity Lemma states that given an $\varepsilon > 0$, every sufficiently dense graph G can be approximated by the union of a bounded number of quasi-random bipartite graphs, where the deviation from randomness is controlled by the tolerance parameter ε. In other words, we can partition the vertex set V into a bounded number of classes $C_0, C_1, ..., C_k$, such that almost every pair (C_i, C_j) behaves similarly to a random bipartite graph, $(1 \leq i < j \leq k)$.

LEMMA 1 (Szemerédi's regularity lemma (1976)). For every positive real ε and for every positive integer m, there are positive integers $N = N(\varepsilon, m)$ and $M = M(\varepsilon, m)$ with the following property: for every graph $G = (V, E)$, with $|V| \geq N$, there is an ε-regular partition of G into $k + 1$ classes such that $m \leq k \leq M$.

The strength of the Regularity Lemma is corroborated by the so-called Key Lemma, which is an important theoretical result introduced by Komlos et al. [3]. It basically states that the reduced graph does inherit many of the essential structural properties of the original graph. Before presenting its original formulation, another kind of graph needs to be defined, namely the *fold graph*. Given an integer t and a graph R (which may be seen as a reduced graph), let $R(t)$ denote the graph obtained by "blowing up" each vertex j of $V(R)$ to a set A_j of t independent vertices, and joining $u \in A_x$ to $v \in A_y$ if and only if (x, y) is an edge in R. Thus, $R(t)$ is a graph in which every edge of R is replaced by a copy of the complete bipartite graph K_{tt}. The following lemma shows a link between the reduced graph R and $R(t)$.

THEOREM 1 (Key Lemma). Given $d > \varepsilon > 0$, a graph R, and a positive integer m, let us construct a graph G by performing the following steps:

1. replace every vertex of R by m vertices;

5

2. replace the edges of R with ϵ-regular pairs of density at least d.

Let H be a subgraph of $R(t)$ with h vertices and maximum degree $\Delta > 0$, and let $\delta = d - \varepsilon$ and $\varepsilon_0 = \delta^\Delta/(2 + \Delta)$. If $\varepsilon \leq \varepsilon_0$ and $t - 1 \leq \varepsilon_0 m$, then H is embeddable into G (i.e. G contains a subgraph isomorphic to H). In fact, we have:

$$||H \to G|| > (\varepsilon_0 m)^h \tag{3}$$

where $||H \to G||$ denotes the number of labeled copies of H in G.

Hence, the Key Lemma provides us a theoretical guarantee on the quality of the summary built from an ϵ-regular partition. In particular, for $t = 1$, $R(t) = R$, and if the constraint on the edge density d is satisfied, the Key Lemma ensures that every small subgraph of R is also a subgraph of G. Thus, we can use the Regularity Lemma to build a summary R of G, and then we can infer structural properties of G by studying the same properties on R.

Finding Regular Partitions. The original proof of the Regularity Lemma is not constructive, but during the last decades different constructive versions have been proposed. In this paper, we focus on the Alon et al.'s [4] work. In particular, they proposed a new formulation of the Regularity Lemma which emphasizes the algorithmic nature of the result.

THEOREM 2. (Alon et al., 1994) For every $\varepsilon > 0$ and every positive integer t there is an integer $Q = Q(\varepsilon, t)$ such that every graph with $n > Q$ vertices has an ε-regular partition into $k + 1$ classes, where $t \leq k \leq Q$. For every fixed $\varepsilon > 0$ and $t \geq 1$ such partition can be found in $O(M(n))$ sequential time, where $M(n) = O(n^{2.376})$ is the time for multiplying two $n \times n$ matrices with $0, 1$ entries over the integers. It can also be found in time $O(\log n)$ on an Exclusive Read Exclusive Write Parallel Random Access Machine(EREW PRAM) with a polynomial number of parallel processors.

A sketch of the proof is then presented. Let H be a bipartite graph with color classes A and B, with $|A| = |B| = n$. Let us define the average degree $\bar{d}$ of H as:

$$\bar{d}(A, B) = \frac{1}{2n} \sum_{i \in A \cup B} deg(i)$$

where $deg(i)$ is the degree of vertex i.

For two distinct vertices $y_1, y_2 \in B$ the *neighbourhood deviation* of y_1 and y_2 is defined as:

$$\sigma(y_1, y_2) = |N(y_1) \cap N(y_2)| - \frac{\bar{d}^2}{n}$$

where $N(x)$ is the set of neighbours of vertex x. For a subset $Y \subset B$ the *deviation* of Y is defined as:

$$\sigma(Y) = \frac{\sum_{y_1, y_2 \in Y} \sigma(y_1, y_2)}{|Y|^2}$$

Let $0 < \varepsilon < 1/16$, it can be proved that, if there exists $Y \subset B$, $|Y| > \varepsilon n$ such that $\sigma(Y) \geq \frac{\varepsilon^3}{2} n$, then at least one of the following cases occurs:

1. $\bar{d} < \varepsilon^3 n$ (H is ε-regular);

2. there exists in B a set of more than $\frac{1}{8}\varepsilon^4 n$ vertices whose degrees deviate from $\bar{d}$ by at least $\varepsilon^4 n$ (H is ε-irregular);

3. there are subsets $A' \subset A$, $B' \subset B$, $|A'| \geq \frac{\varepsilon^4}{n} n$, $|B'| \geq \frac{\varepsilon^4}{n} n$ such that $|\bar{d}(A', B') - \bar{d}(A, B)| \geq \varepsilon^4$ (H is ε-irregular).

Note that one can easily check if 1 holds in time $O(n^2)$. Similarly, it is trivial to check if 2 holds in $O(n^2)$ time, and in case it holds to exhibit the required subset of B establishing this fact. If the first two conditions are not verified, the 3 condition must be checked. To this end, we have to find the subsets A', B', called *certificates*, that witness the irregularity of the bipartite graph H. To address this task, we first select a subset of B whose vertex degrees "deviate" the most from the average degree $\bar{d}$ of H. More formally: for each $y_0 \in B$ with $|deg(y_0) - \bar{d}| < \varepsilon^4 n$ we find the vertex set $B_{y_0} = \{y \in B \,|\, \sigma(y_0, y) \geq 2\varepsilon^4 n\}$. The proof provided by Alon et al. guarantees the existence of at least one such y_0 for which $|B_{y_0}| \geq \frac{\varepsilon^4}{4} n$. Thus, the subsets $B' = B_{y_0}$ and $A' = N(y_0)$ are the required certificates. These two subsets represent the collection of vertices that contribute more to the irregularity of the pair (A, B). The sets $\bar{A}' = A \setminus A'$, $\bar{B}' = B \setminus B'$ are called *complements*. Since the computation of the quantities $\sigma(y, y')$, for y, $y' \in B$, can be done by squaring the adjacency matrix of H, the overall complexity of this algorithms is $O(M(n)) = O(n^{2.376})$.

Before reporting Alon et al.'s algorithm, we present a measure to assess the goodness of a partition of the vertex set V of a graph G, which was introduced by Szemerédi [1]. Given a partition $\mathcal{P} = \{C_0, C_1, \ldots, C_k\}$ of a graph $G = (V, E)$, the index (*sze_idx*) of the partition $\mathcal{P}$ is defined as follows:

$$sze\_ind(\mathcal{P}) = \frac{1}{k^2} \sum_{s=1}^{k} \sum_{t=s+1}^{k} d(C_s, C_t)^2$$

Since $0 \leq d(C_s, C_t) \leq 1$, it can be seen that $sze\_ind(\mathcal{P}) \leq \frac{1}{2}$. Szemerédi proved that if a partition $\mathcal{P}$ violates the regularity condition, then it can be refined by a new partition $\mathcal{P}'$ such that $sze\_ind(\mathcal{P}') > sze\_ind(\mathcal{P})$. Finally, we can now present Alon et al.'s algorithm, which provides a way to find an ε-regular partition. The procedure is divided into two main steps: in the first step all the constants needed during the next computation are set; in the second one, the partition is iteratively created. An iteration is called *refinement step*, because, at each iteration, the current partition is closer to a regular one.

Alon et al.'s Algorithm.

1. Create the initial partition: arbitrarily divide the vertices of G into an equitable partition $\mathcal{P}_1$ with classes $C_0, C_1, \ldots, C_b$ where $|C_i| = \lfloor \frac{n}{b} \rfloor$.

2. Check Regularity: for every pair (C_r, C_s) of $\mathcal{P}_i$, verify if it is ε-regular or find two certificates $A' \subset C_r$, $B' \subset C_s$, $|A'| \geq \frac{\varepsilon^4}{16}|C_1|$, $|B'| \geq \frac{\varepsilon^4}{16}|C_1|$ such that $|\bar{d}(A', B') - \bar{d}(C_s, C_t)| \geq \varepsilon^4$.

3. Count regular pairs: if there are at most $\varepsilon\binom{k_i}{2}$ pairs that are not ε-regular, then stop. $\mathcal{P}_i$ is an ε-regular partition.

4. Refine: apply a refinement algorithm and obtain a partition $\mathcal{P}'$ with $1 + k_i 4^{k_i}$ classes, such that $sze\_ind(\mathcal{P}') > sze\_ind(\mathcal{P})$.

5. Go to step 2.

Even if the above mentioned algorithm has polynomial worst case complexity in the size of G, there is a hidden tower-type dependence on an accuracy parameter. Unfortunately, Gowers [21] proved that this tower function is necessary in order to guarantee a regular partition for *all* graphs. This implies that, in order to have a faithful approximation, the original graph size should be astronomically big. This has typically discouraged researchers from applying regular partitions to practical problems, thereby confining them to the purely theoretical realm.

To make the algorithm truly applicable, [9], and later [22, 7], instead of insisting on provably regular partitions, proposed a few simple heuristics that try to construct an approximately regular partition. In the next section, we present a new heuristic algorithm which is characterized by an improvement of the summary quality both in terms of reconstruction error and of noise filtering.

4. The Summarization Algorithm

The main limitations which prevent the application of Alon et al.'s algorithm to practical problems concern Step 2 and Step 4. In particular, in Step 2 the algorithm checks the regularity of all classes pairs by using the three conditions previously described. Given a pair (C_r, C_s), condition 1 verifies if it is ε-regular, otherwise conditions 2 and 3 are used to obtain the *certificates* C'_r and C'_s that witness the irregularity. The main obstacle concerning the implementation of condition 3 is the necessity to scan over *almost all possible subsets* of C_s. To make the implementation of condition 3 feasible, given a class C_s, we select in a *greedy way* a set $Y' \subseteq C_s$ with the highest deviation $\sigma(Y')$ (the deviation is defined in 3). To do so, the nodes of C_s are sorted by bipartite degree, and Y' is built by adding $\frac{\varepsilon^4}{4}n$ nodes with the highest degree. At each iteration of the greedy algorithm, the node with a degree that deviates most from the average degree is added to the candidate certificate Y'. This last operation is repeated until the subset C'_s, that satisfies condition 3, is found. This almost guarantees to put in the candidate certificate the nodes that have a *connectivity pattern* that deviates most from the one characterizing the majority of the nodes which belong to C_s.

As far Step 4 is concerned, here an irregular partition $\mathcal{P}^i_\varepsilon$ is *refined* by a new partition $\mathcal{P}^{i+1}_\varepsilon$, such that the partition index sze_idx is increased. This step poses the main obstacle towards a practical version of Alon et al.'s algorithm involving the creation of an exponentially large number of subclasses at each iteration. Indeed, as we have said, Step 2 finds all possible irregular pairs in the graph. As a consequence, each class may be involved with up to $(k_i - 1)$ irregular pairs, k_i being the number of classes in the current partition $\mathcal{P}^i_\varepsilon$, thereby leading to an exponential growth. To avoid the problem, for each class, one can limit the number of irregular pairs containing it to at most one, possibly chosen randomly among all irregular pairs. This simple modification allows one to divide the classes into a constant, rather than exponential, number of subclasses l (typically $2 \leq l \leq 7$). Despite the crude approximation this seems to work well in practice.

The devised algorithm takes as input two main parameters, the tolerant parameter ε and the minimum compression rate c_min, that acts as a stopping criterion in the refinement process, and returns an approximated ϵ-regular partition. Its pseudocode is reported in Algorithm 1, where the procedure APPROXALONCERTIFICATES, based on the two heuristics described above, takes as input a partition $\mathcal{P}_\epsilon^i$ and returns the number of irregular pairs of $\mathcal{P}_\epsilon^i$. In the next paragraph, we describe the REFINEMENT procedure which refines a partition $\mathcal{P}_\epsilon^i$ into a partition $\mathcal{P}_\epsilon^{i+1}$. Its pseudocode is reported in Algorithm 2. The overall complexity of our summarization algorithm is $O(M(n)) = O(n^{2.376})$, which is dominated by the verification of condition 3.

Algorithm 1 The Summarization Algorithm

1: **procedure** APPROXALON(ε, c_min, $G = (V, E)$)
2:     partitions = empty list
3:     $\mathcal{P}_\varepsilon^1$ = Create initial random partition from G
4:     **while** True **do**
5:         #irr_pairs = APPROXALONCERTIFICATES($\mathcal{P}_\varepsilon^i$)
6:         **if** #irr_pairs $> \varepsilon \binom{k}{2}$ or COMPRESSRATE($\mathcal{P}_\varepsilon^i$) < c_min **then**
7:             break
8:         **else**
9:             $\mathcal{P}_\varepsilon^{i+1}$ = REFINEMENT($\mathcal{P}_\varepsilon^i$)
10:             **if** $\mathcal{P}_\varepsilon^{i+1}$ is ε-regular **then**
11:                 partitions.add($\mathcal{P}_\varepsilon^{i+1}$)
12:             **else**
13:                 break
14:     Select best partition $\mathcal{P}^*$ with maximum sze_idx from list partitions

---

Given a partition $\mathcal{P}_\epsilon^i = \{C_0, C_1, \ldots, C_{k_i}\}$, the REFINEMENT procedure starts by randomly selecting a class C_i, then iteratively processes all the others.

- If C_i is ε-regular with all the others, the procedure sorts the nodes of C_i by their *internal degree*, i.e. the degree calculated with respect to the nodes of the same class, obtaining the following sorted sequence of nodes $v_1, v_2, v_3, v_4, v_5, v_6, \cdots, v_{|C_i|}$. The next step splits (UNZIP) this sequence into two sets $C_i^1 = \{v_1, v_3, v_5, \cdots, n_{|C_i|-1}\}$ and $C_i^2 = \{v_2, v_4, v_6, \cdots, v_{|C_i|}\}$. The latter sets are part of the refined partition $\mathcal{P}_\varepsilon^{i+1}$.

- If C_i forms an irregular pair with other classes, the heuristic selects the candidate C_j that shares the most similar internal structure with C_i by maximizing $S = d(C_i, C_j) + (1 - |d(C_i, C_i) - d(C_j, C_j)|)$, where $d(C_i, C_i) = e(C_i, C_i)/|C_i|^2$ is the *internal density*.

  After selecting the best matching class C_j, we are ready to split the pair (C_i, C_j) in 4 new classes $C_i^1, C_i^2, C_j^1, C_j^2$ based on the internal densities of the certificates C_i' and C_j'.

  - In particular, a SPARSIFICATION procedure is applied when the internal density of a certificate is below a given threshold. This procedure randomly splits the certificate into two new classes. In order to match the equi-cardinality

Algorithm 2 Refinement step performed at the i-th iteration of the summarization algorithm 1. Statements 5,10 and 12 may add a node to C_0. $\mathcal{P}_\varepsilon^i$ is the partition at iteration i of the summarization algorithm

---

1: **procedure** REFINEMENT($\mathcal{P}_\varepsilon^i$)
2:      **for** each class C_i in $\mathcal{P}_\varepsilon^i$ **do**
3:          **if** C_i is ε-regular with all the other classes **then**
4:              $C_i = $ SORT_BY_INDEGREE(C_i)
5:              $C_i^1, C_i^2 = $UNZIP($C_i$)
6:          **else**
7:              Select C_j with most similar internal structure
8:              Get certificates (A', B') and complements $(\bar{A}', \bar{B}')$ of C_i, C_j
9:              **if** $d(A', A') < 0.5$ **then**
10:                $C_i^1, C_i^2 = $ SPARSIFICATION($A', \bar{A}' \cup \bar{B}'$)
11:              **else**
12:                $C_i^1, C_i^2 = $ DENSIFICATION($B', \bar{B}' \cup \bar{B}'$)
13:              Perform step 9,10,11,12 for B'
14:      **if** $|C_0| > \varepsilon n$ and $|C_0| > |\mathcal{P}_\varepsilon^{i+1}|$ **then**
15:          Uniformly distribute nodes of C_0 between all the classes
16:      **else**
17:          **return** $(\mathcal{P}_\varepsilon^{i+1}, irregular)$
18:      **return** $(\mathcal{P}_\varepsilon^{i+1}, regular)$

---

        property, the new classes are filled up to $|C_i|/2$ by adding the remaining nodes from the corresponding complement. We choose the nodes that share the minimum number of connections with the new classes.

    – On the other hand, if the internal density of a certificate is above a given threshold, then a DENSIFICATION procedure is applied. In particular, the heuristics sorts the nodes of the certificate by their internal degree and UNZIP the set into two new classes. Also in this case, we fill the new sets up to $|C_i|/2$ by adding the remaining nodes from the corresponding complement by choosing the nodes which share the major number of connections with the new classes.

5. Graph Search Using Summaries

In this section, we discuss how to use our summarization framework to efficiently address the graph search problem defined under a similarity measure. The aim of graph search is to retrieve from a database the top-k graphs that are most similar to a query graph.

Problem Definition. We consider a graph database $\mathcal{D}$ containing a high number of simple undirected graphs $g_j \in \mathcal{D}$, $j = 1 \ldots |\mathcal{D}|$, and, for the sake of generality, we allow the edges to be weighted.

PROBLEM 1 (Graph search). Given a graph database $\mathcal{D} = \{g_1, g_2, \cdots, g_{|D|}\}$, a query graph q, and a positive integer k, the graph similarity search problem is to find the top-k graphs in $\mathcal{D}$ that are most similar to q according to a similarity measure.

As far as the similarity measure is concerned, the most used one is the *graph edit distance (GED)* due to its generality, broad applicability and noise robustness [17, 20]. However, since the GED computation is NP-hard, it is not suited to deal with large graphs. To overcome this limitation, we use the *spectral distance* [23], which is computed by comparing the eigenvalues of the two graphs being matched. The choice of this measure is motivated by the work of Van Dam and Haemers [24], who show that graphs with similar spectral properties generally share similar structural patterns. In this paper, we introduce a slightly modified version of the spectral distance to increase its range of applicability to pairs of graphs that violate the assumption of the Theorem 1 in [23], which assumes a precise order between the eigenvalues of the two graphs being matched. To this aim, we simply compute the absolute value of the difference between the *i-th* eigenvalue of the first graph with the *i-th* one of the second graph.

DEFINITION 3 (Spectral distance). Given two simple undirected weighted graphs $G_1 = (V_1, W_1)$ with $|V_1| = n_1$, and $G_2 = (V_2, W_2)$ with $|V_2| = n_2$. Let us denote the corresponding spectra as $0 = \lambda_1^{(1)} \leq \lambda_2^{(1)}, \leq \cdots \leq \lambda_{n_1}^{(1)}$ and $0 = \lambda_1^{(2)} \leq \lambda_2^{(2)}, \leq \cdots \leq \lambda_{n_2}^{(2)}$. We may assume without loss of generality that $n_2 > n_1$. The spectral distance is then defined as follows

$$SD(G_1, G_2, l) = \frac{1}{n_1} \left(\sum_{i=1}^{l} |\lambda_i^{(2)} - \lambda_i^{(1)}| + \sum_{i=l+1}^{n1} |\lambda_{i+n_2-k}^{(2)} - \lambda_i^{(1)}| \right) \tag{4}$$

where, l controls which part of the spectra are being matched. In particular, eigenvalues of G_1 are compared with the head and tail eigenvalues of the G_2.

Using The Summaries. In our approach, all the graphs contained in a database are summarized off-line, while the query graph is summarized on-line by means of our summarization framework. Thus, graph search can be performed on graph summaries, and this allows us to speed up the search process and to reduce the storage space. In particular, for each graph g_j of a database $\mathcal{D}$, we store two different quantities: the summary r_j of g_j and the eigenvalues eig_{r_j} of r_j. We then summarized on-line the query graph q obtaining its summary r_q. Finally, we compute the spectral distance between r_q and each summary $r_j \in \mathcal{D}$. The desired top-k graphs will be obtained by selecting, from $\mathcal{D}$, the k graphs corresponding to the k smallest value of the spectral distance previously computed. The pseudocode of our approach to graph search is reported in Algorithm 3.

6. Experimental Evaluation

In this section, we evaluate our summarization algorithm both on synthetic graphs and on real-world networks to assess:

- the ability of the proposed algorithm to separate structure from noise;

- the usefulness of the summaries in retrieving from a database the top-k graphs that are most similar to a query graph.

Algorithm 3 Graph Search Using The Summaries

1: **procedure** ADDGRAPHTODATABASE($g, \mathcal{D}$)
2:     $r = $ Summarize g
3:     $eig_r = $ Calculate the eigenvalues of the adj. matrix of r
4:     Store (r, eig_r) in $\mathcal{D}$
5: **procedure** 2-STAGEGRAPHSEARCH($q, \mathcal{D}$)
6:     $r_q = $ Summarize q
7:     $eig_{r_q} = $ Calculate the eigenvalues of the adj. matrix of r_q
8:     $sd\_array = \emptyset$
9:     **for** r_j in $\mathcal{D}$ **do**
10:        $sd = $ Spectral Distance($r_j, r_q, eig_{r_j}, eig_{r_q}$)
11:        Append sd to sd_array
12:     Order sd_array
13:     **return** first k results of sd_array and their relative graphs.

Figure 2: The adjacency matrix of an undirected synthetic graph of 2000 vertices. The graph is generated by corrupting 5 cliques as described in Algorithm 4. In particular, the *intra-cluster noise probability* is 0.2 and the *inter-cluster noise probability* is 0.4.

Experimental Settings. In our experiments we used both synthetic graphs and real-world networks. We generated synthetic graphs with a cluster structure, where the clusters are perturbed with different levels of noise. In particular, each graph is generated by adding spurious edges between cluster pairs and by dropping edges inside each cluster. Figure 2 provides a concrete example with a visual explanation. The pseudocode of the algorithm used to generate the synthetic datasets is reported in Algorithm 4.

As far as the real-world networks are concerned, we used two different datasets which have been taken from two famous repositories: the Stanford Large Network Dataset Collection SNAP [25] and the Konect repository of the University Koblenz-Landau Konect [26]. In particular, we used the following networks: Facebook [27], Email-Eu-core [28][29],

Openflights[30], and Reactome [31]. Our algorithm is implemented in Python 3.6.3 [2] and the experiments are performed on an Intel Core i5 @ 2.60GHz HP Pavilion 15 Notebook with 8GB of RAM (DDR3 Synchronous 1600 MHz) running Arch-Linux with kernel version 4.14.4-1.

Algorithm 4 Synthetic graph generator. *Input parameters*: n is the size of the desired graph G; num_c is the number of clusters contained in G; η_1 is the probability of adding a spurious edge between a pair of clusters (inter-cluster noise probability); η_2 is the probability of dropping an edge inside a cluster (intra-cluster noise probability). *Output*: G.

1: **procedure** SYNTHGRAPHGEN(n, num_c, η_1, η_2)
2:     G = Generate Erdős Rényi graph of size n using η_1 as edge probability
3:     $clust\_dim = n/num\_c$
4:     **for** i in num_c **do**
5:         Select $clust\_dim$ nodes from G and create cluster c_i with them
6:         For each edge in c_i drop it with probability η_2
7:     **return** G

6.1. Graph Summarization

We performed experiments on both synthetic graphs and on real-world networks to assess the ability of the proposed algorithm to separate structure from noise. As evaluation criterion, we used the reconstruction error, which is expressed in terms of normalized l_p norm computed between the similarity matrix of an input graph G and the similarity matrix of the corresponding reconstructed graph G'.

DEFINITION 4 (Reconstruction error). Given the similarity matrix of the input graph A_G and the similarity matrix of the reconstructed graph $A_{G'}$, the reconstruction error is defined as follows:

$$l_p(\mathbf{A_G}, \mathbf{A_{G'}}) = (\sum_{i=1}^{n} \sum_{j=1}^{n} (A_G(i, j) - A_{G'}(i, j))^p)^{\frac{1}{p}}$$

We decided to use the reconstruction error in order to compare our results with the ones presented by Riondato et al. [13], who evaluated the summary quality using this measure. This choice is due to the fact that their algorithm summarizes a graph by minimizing the reconstruction error. However, they pointed out that the reconstruction error has some shortcomings. In particular, given an unweighted graph G, it is possible to produce an uninteresting summary with only one supenode corresponding to the vertex set and l_1 reconstruction error at most n^2. On the other hand, if we obtained an useful summary, where each pair of vertices belonging to a supernode share an high number of common neighbors, then we get a low (say $o(n^2)$) l_1 reconstruction error: this is a desirable behavior because low values of l_1 correspond to high quality summaries. Unfortunately, such low values are often obtained only with summaries having an high

---

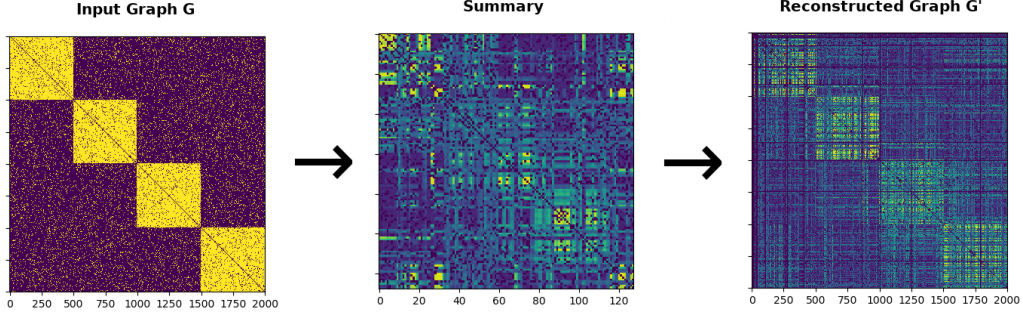[2]The implementation is available from https://github.com/MarcoFiorucci/graph-summarization-using-regular-partitions

Figure 3: We summarized an input graph G by using Algorithm 1. We then "blow up" the summary to obtain the reconstructed graph G'.

number of supernodes. This prevents to adopt the reconstruction error as a general measure to assess the summary quality.

As far the summarization and reconstruction steps are concerned, we proceeded, in all the experiments, in the following way: we applied our summarization algorithm (see Algorithm 1) to summarize an input graph G. We then "blow-up" the summary in order to obtain the reconstructed graph G', which preserves the main structure carried by the input graph (Figure 3).

Noise Robustness Evaluation. We study the ability of the proposed algorithm to separate structure from noise in graphs performing an extensive series of experiments on both synthetic graphs and real-world networks. As far synthetic graph experiments are concerned, we generated a graph G by corrupting the clusters of GT in the following way: we added spurious edges between each cluster pair with probability η_1, and we dropped edges inside each cluster with probability η_2 (see algorithm 4). As far as the real-world networks experiments are concerned, we added spurious edges with probability *noise probability* to a original graph GT obtaining an input graph G.

In the ideal case, the distance between G and the corresponding reconstructed graph G' should be only due to the filtered noise, while the distance between GT an G' should be closed to zero. Hence, we computed the reconstruction error $l_2(G', GT)$ to assess the robustness of our summarization framework against noise.

Experiment 1. We generated synthetic graphs of different sizes, spanning from 10^3 up to 10^4 nodes. We synthesized 250 graphs by considering, for each of the 10 different sizes, all the 25 combinations of the following noise probabilities:

- the probability η_1 of adding a spurious edge between a pair of clusters, called *inter-cluster noise probability*, which assumes values in $\{0.1, 0.2, 0.3, 0.4, 0.5\}$;

- the probability η_2 of dropping an edge inside each cluster, called *intra-cluster noise probability*, which assumes values in $\{0.1, 0.2, 0.3, 0.4, 0.5\}$.

Let's consider a synthetic graph $G_{n,(\eta_1,\eta_2)}$, where n is its size, and (η_1, η_2) corresponds to one of the 25 pairs of noise probabilities. For each $G_{n,(\eta_1,\eta_2)}$ we obtained the reconstructed graph $G'_{n,(\eta_1,\eta_2)}$, and we then computed the reconstruction error $l_2(G'_{n,(\eta_1,\eta_2)}, GT)$. Given
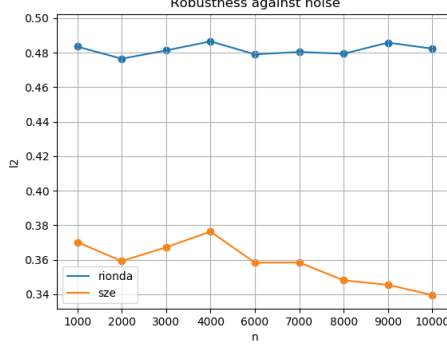
14

Figure 4: The plot shows the medians computed, for each size n, from the 25 values of $l_2(G'_{n,(\eta_1,\eta_2)}, GT)$, where (η_1, η_2) corresponds to one of the 25 pairs of noise probabilities. The curve "rionda" is obtained by using Riondato et al.'s algorithm [13], while the curve "sze" is obtained by applying our summarization framework.

a size n, we computed the median m_n of $\{l_2(G'_{n,(0.1,0.1)}, GT),\ l_2(G'_{n,(0.1,0.2)}, GT), \cdots,$ $l2(G'_{n,(0.5,0.5)}, GT)\}$. We reported in figure 4 the 10 medians computed using our summarization framework and the corresponding medians obtained by applying Riondato et al.'s algorithm [13]. We can see how our framework outperforms the state-of-the-art summarization algorithm in terms of robustness against noise.

Experiment 2. The aim of this experiment is to study separately the robustness against the inter-cluster and the intra-cluster noise. Let's consider the probability of dropping an edge inside each cluster η_2 equals to 0.2 and the graph size n equals to 10^4. To asses the inter-cluster noise robustness, we generated synthetic graphs $G_{10^4,(\eta_1,\,0.2)}$, where η_1 assumes values in $\{0.1,\ 0.15,\ 0.2,\ 0.25,\ 0.3,\ 0.35,\ 0.4,\ 0.45,\ 0.5\}$, and we computed the reconstruction errors $l_2(G'_{10^4,(\eta_1,\,0.2)}, GT)$. As far the intra-cluster noise is concerned, we chose the probability of adding a spurious edges between each pair of clusters $\eta_1 = 0.2$, and the graph size $i = 10^4$. We then generated synthetic graphs $G_{10^4,(0.2,\,\eta_2)}$, where η_2 assumes values in $\{0.1,\ 0.15,\ 0.2,\ 0.25,\ 0.3,\ 0.35,\ 0.4,\ 0.45,\ 0.5\}$, and we computed the reconstruction errors $l_2(G'_{10^4,(0.2,\,\eta_2)}, GT)$.

Figure 5 illustrates the comparison between our results with those obtained by applying Riondato et al.'s algorithm [13]. This results are in accord to those presented in figure 4, and provides an experimental verification of the ability of our method to separate structure from noise in graphs.

Experiment 3. We added spurious edges with probability *noise probability* to an original real-world network GT obtaining an input graph G. The *noise probability* assumes values in $\{0.01,\ 0.02,\ 0.03,\ 0.04,\ 0.05,\ 0.06,\ 0.07,\ 0.08,\ 0.09,\ 0.1\}$. We applied this procedure on real-word networks, which have been taken from the Stanford Large Network Dataset Collection SNAP [25] and from the Konect repository of the University Koblenz-Landau Konect [26]. Since our framework is based on the Regularity Lemma, which is suited to deal only with dense graphs, we expect to obtain low quality summaries from sparse real-world networks. However, as shown in figure 6, our framework outperforms the state-of-the-art summarization algorithm in terms of robustness against noise providing good quality summary even on sparse real-world networks. In particular,
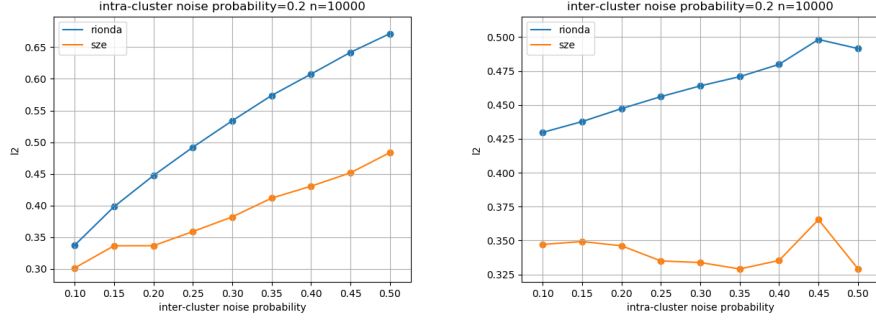
15

Figure 5: The plot on the left represents $l_2(G'_{10^4,(0.2,\eta_2)},GT)$ versus the intra-noise probability. The plot on the right represents $l_2(G'_{10^4,(\eta_1,0.2)},GT)$ versus the inter-noise probability. The curve "rionda" is obtained by using Riondato et al.'s algorithm, while the curve "sze" is obtained by applying our summarization framework.

we can see how the quality increases with the size of the input graph, which is in accord with the assumptions of the Regularity Lemma.
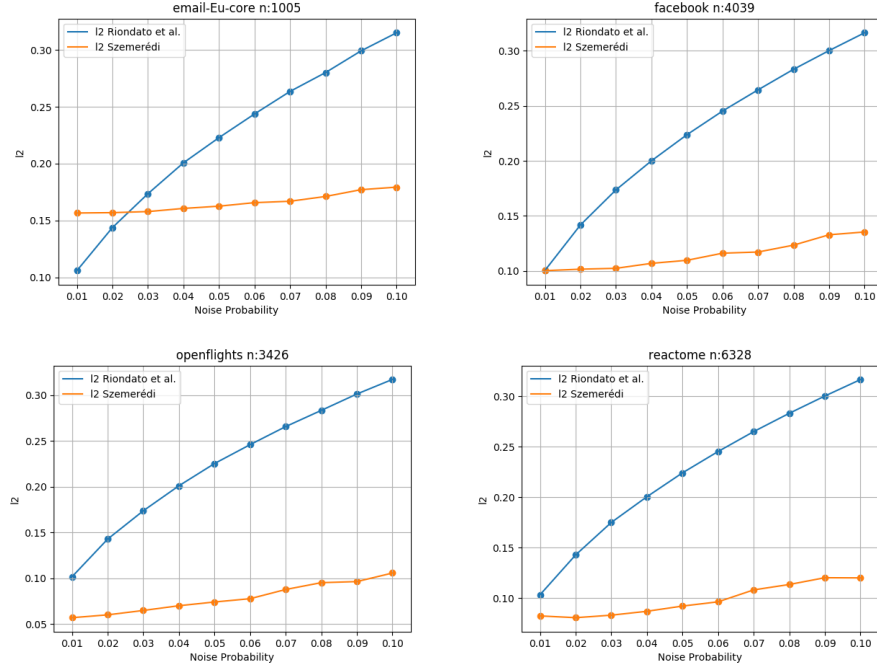

Figure 6: These plots represent the median of the $l_2(G',GT)$ versus the noise probability. We run 20 experiments for each value of the noise probability. The curve "rionda" is obtained by using Riondato et al.'s algorithm, while the curve "sze" is obtained by applying our summarization framework.

6.2. Graph Search

We performed extensive experiments on synthetic datasets to assess the usefulness of the summaries in retrieving, from a database, the top-k graphs that are most similar to a query graph. To this end, we evaluate the quality of the answer in terms of the found top-k similar graphs, and we evaluate the scalability both in the size of the database and in the size of the graphs.

Quality Evaluation. We conducted the following experiment: we compared graph search on the summaries with the baseline approach, in which the spectral distance is computed between no preprocessed graphs. The aim of the experiment is to show that pre-summarizing the graphs in the databases increases the noise robustness of the search process. We created a database $\mathcal{D}$ contained synthetic graphs, which have different structures corrupting with different levels of noise (see algorithm 4). In particular, each graph is generated by combining the following three factors: five different possible number of clusters $\{4, 8, 12, 16, 20\}$, six different possible levels of intra-cluster noise probability $\{0.05, 0.1, 0.15, 0.2, 0.25, 0.3\}$ and six different possible levels of inter-cluster noise probability $\{0.05, 0.1, 0.15, 0.2, 0.25, 0.3\}$. Given a size n, we generated 180 graphs considering all the possible combinations of these three parameters. As described in Algorithm 3, we stored in $\mathcal{D}$ the eigenvalues of the 180 synthetic graphs, their summaries and the corresponding eigenvalues. Finally, we grouped the database graphs into five groups. Each group ω_i, with $i = 1, 2, 3, 4, 5$, is composed by 36 graphs that are generated by corrupting the same cluster structure with different combinations of intra-cluster and inter-cluster noise probability. Hence, all the graphs belonging to a given group ω_i are similar, since they have the same main structure.

More formally, we constructed a set Q of five query graphs by randomly sampling one graph from each group ω_i. Then, we first computed the spectral distance between $q_i \in Q$ and every graph in the database $\mathcal{D}$. We then calculated the $AP@k$ for each query q_i by considering relevant the graphs belonging to ω_i i.e. the same group of q_i. Finally, we computed the $MAP@k$ score by averaging the *average precision* $AP@k(q_i)$ of the five graphs in Q.

We repeated the same procedure using the summaries of the 180 synthetic graphs contained in $\mathcal{D}$. The aim of this experiment is to compare the quality obtained using our approach with that obtained by computing the spectral distance between original graphs. We performed the experiment by considering the following graph sizes $n = 1500, 2000, 3000, 7000$.

The $AP@k(q_i)$ and the $MAP@k$ are defined as follows.

DEFINITION 5 (Average precision). Given a query $q \in Q$, a set of relevant graphs ω_i (graphs that share the same structure with q). Let us consider the output top-k graphs in a database $\mathcal{D}$ ordered by crescent spectral distance. We define the average precision at k as follows.

$$AP@k(q) = \frac{1}{|w_i|} \cdot \sum_{j=1}^{k} Precision(j) \cdot Relevance(j) \tag{5}$$

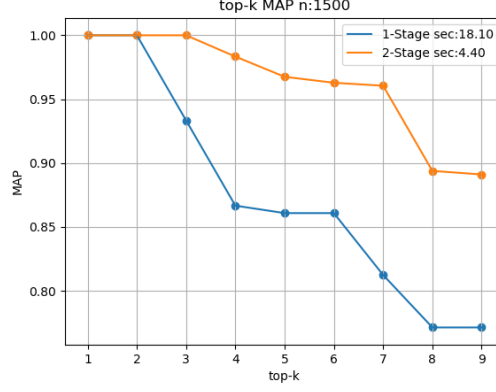where $Precision(j)$ is the relevant proportion of the found top-k graphs, while $Relevance(j)$

Figure 7: The MAP@k of the top-k graphs given as output in a database of 180 graphs. The size of the graphs contained in the database is $n = 1500$. Two-stage is referred to the use of our pre-summarization approach to address the graph search problem, while one-stage is referred to the search on the original graphs.

is 1 if the considered graph is part of ω_i and is 0 otherwise. Finally, $|\omega_i|$ is the number of relevant graphs.

DEFINITION 6 (Mean average precision). Given a query set Q, the mean average precision is defined as follows.

$$MAP(Q) = \frac{1}{|Q|} \cdot \sum_{q_i \in Q} AP@k(q_i) \tag{6}$$

In particular, the higher is the value of the $MAP \in [0, 1]$, the higher is the quality of the proposed graph search algorithm. Figures 7, 8, 9, 10 show that the proposed summarization based approach improved the query quality.
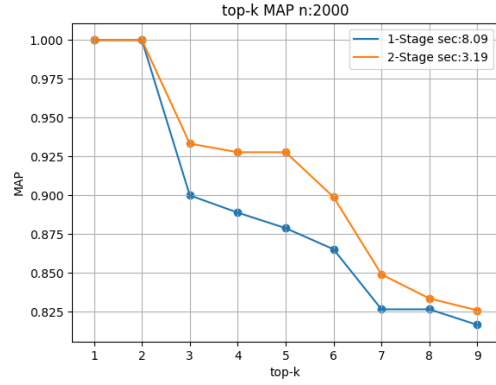

Figure 8: The MAP@k of the top-k graphs given as output in a database of 180 graphs. The size of the graphs contained in the database is $n = 2000$. Two-stage is referred to the use of our pre-summarization approach to address the graph search problem, while one-stage is referred to the search on the original graphs.
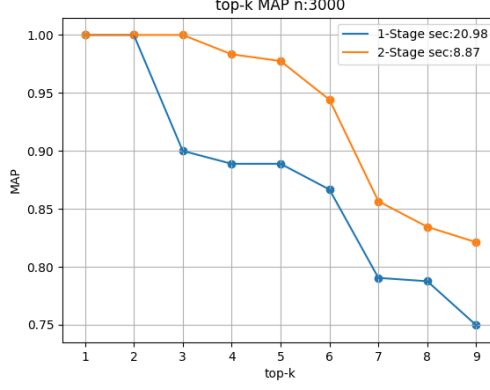
18

Figure 9: The MAP@k of the top-k graphs given as output in a database of 180 graphs. The size of the graphs contained in the database is $n = 3000$. Two-stage is referred to the use of our pre-summarization approach to address the graph search problem, while one-stage is referred to the search on the original graphs.
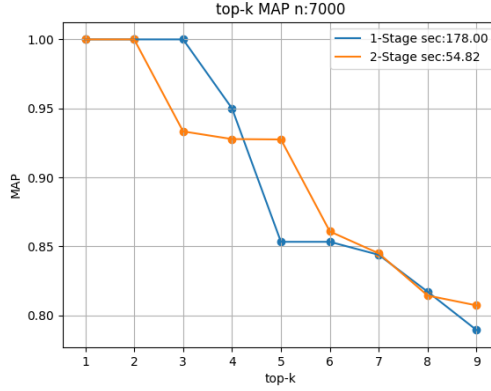

Figure 10: The MAP@k of the top-k graphs given as output in a database of 180 graphs. The size of the graphs contained in the database is $n = 7000$. Two-stage is referred to the use of our pre-summarization approach to address the graph search problem, while one-stage is referred to the search on the original graphs.

Scalability*.* In order to evaluate the scalability of our approach, we conducted two different experiments. In the first one, we investigated the time required to perform a single query as the dimension of the database $\mathcal{D}$ grows. In the second one, we investigated the query time in function of the size of the query graph.

In the first experiment, we fixed the size of all the graphs to be $n = 2000$. We then generated the graphs in $\mathcal{D}$ using all the possible combinations of the following factors: three different numbers of clusters $\{4, 12, 20\}$, six different levels of intra-cluster noise probability $\{0.05, 0.1, 0.15, 0.2, 0.25, 0.3\}$, and six different levels of inter-cluster noise probability $\{0.05, 0.1, 0.15, 0.2, 0.25, 0.3\}$. The combination of these three parameters allow us to generate 108 graphs. We then copied them enough times to reach a database cardinality spanning from 10^3 up to 10^4 graphs.
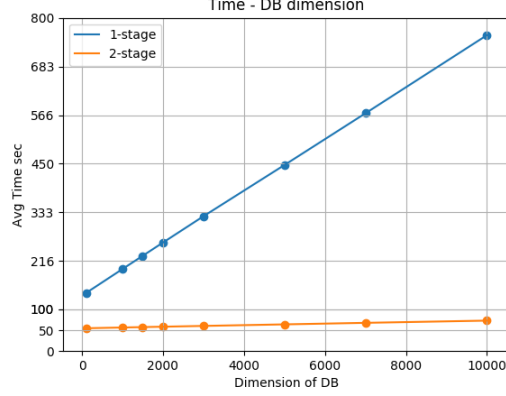
19

Figure 11: This plot shows the time expressed in seconds to perform a query as we increase the size of the database. Two-stage is referred to the use of our pre-summarization approach to address the graph search problem, while one-stage is referred to the search on the original graphs.

The query time is calculated as follows:

$$t = t\_s(q) + t\_eig(r_q) + t\_SD(eig_{r_q}, eig_{r_j}) \quad j = 1, \ldots, |\mathcal{D}|. \tag{7}$$

where $t\_s(q)$ is the time required to obtain the summary r_q of the query graph q; $t\_eig(r_q)$ is the time required to calculate the eigenvectors of r_q; and $t\_SD(eig_{r_q}, eig_{r_j})$ is the time required to calculate the spectral distances between r_q and each graph summary r_j contained in $\mathcal{D}$. We reported in Figure 11, the computed time t versus the cardinality of the database $\mathcal{D}$.

In the second experiment, we generated different databases $\mathcal{D}_i$, containing 10000 graphs. All the graphs in $\mathcal{D}_i$ have the same size and have been created analogously as the previous experiment. We then constructed a query graph q_i of the same size of the graphs in $\mathcal{D}_i$, and we measured the query time t_i as we did for the previous experiment. We reported in Figure 12, the computed t_i versus the size of the graph query q_i. Figures 11, 12 provide us an experimental verification of the scalability of our approach both in the size of the database and in the size of the query graph.

7. Conclusions

In this work, we introduced a graph summarization approach based on the Regularity Lemma, which provides us with a principled way to describe the essential structure of large graphs using a small amount of data. We have successfully validated our framework both on synthetic and real-world graphs showing that our algorithm surpasses the state-of-the-art graph summarization methods in terms of noise robustness. In the second part of the paper, we presented an algorithm to address the graph similarity search problem exploiting our summaries. In particular, the proposed method is tailored for efficiently dealing with databases containing a high number of large graphs, and, moreover, it is robust against noise, which is always presented in real-world data. This achievement seems of particular interest since, to the best of our knowledge, we are the first to devise
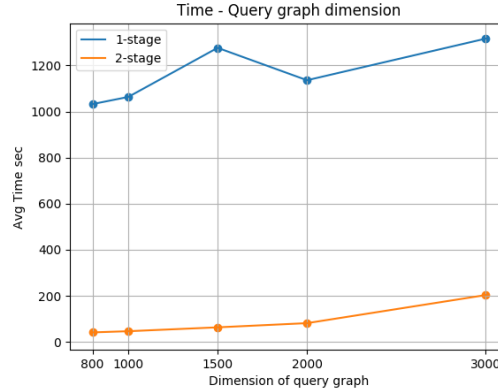
20

Figure 12: This plot shows the time (expressed in minutes) for retrieving the top-k graphs in a database composed of 10000 graphs as we increase the dimension of the query graph. Two-stage is referred to the use of our pre-summarization approach to address the graph search problem, while one-stage is referred to the search on the original graphs.

a graph search algorithm which satisfies all the above requirements together. A weak point of our summarization algorithm is related to its time complexity, which prevents the application of our framework to networks of millions of nodes. This demands for the development of efficient approaches suited to deal with large sparse graphs. In this direction, it would be a good idea to develop a heuristic based on the version of the Weak Regularity Lemma introduced by Fox et al. [32], as well as designing a distributed version of the regular decomposition algorithm introduced by Reittu et al. [33], who studied the linkage among the Regularity Lemma, the Stochastic Block Model and the Minimum Description Length. We think that the notion of regular partition will allow us to tackle the scalability issue faced by graph-based approaches [34, 35]. This would pave the way for a principled approach to massive network data analysis by combining modern graph theory and combinatorics with machine learning and pattern recognition.

References

References

[1] E. Szemerédi, On sets of integers containing no k elements in arithmetic progression, Acta Arithm. 27 (1975) 199–245.

[2] J. Komlós, A. Shokoufandeh, M. Simonovits, E. Szemerédi, The regularity lemma and its applications in graph theory, in: G. B. Khosrovshahi, A. Shokoufandeh, A. Shokrollahi (Eds.), Theoretical Aspects of Computer Science: Advanced Lectures, Springer, Berlin, 2002, pp. 84–112.

[3] J. Komlos, M. Simonovits, Szemerédi's Regularity Lemma and its applications in graph theory, DIMACS Technical Report.

[4] N. Alon, R. Duke, H. Lefmann, V. Rodl, R. Yuster, The algorithmic aspects of the regularity lemma, Journal of Algorithms 16 (1) (1994) 80 – 109.

[5] A. Czygrinow, V. Rdl, An algorithmic regularity lemma for hypergraphs, SIAM Journal on Computing 30 (4) (2000) 1041–1066.

[6] A. M. Frieze, R. Kannan, A simple algorithm for constructing szemere'di's regularity partition, Electr. J. Comb. 6.

[7] M. Fiorucci, A. Torcinovich, M. Curado, F. Escolano, M. Pelillo, On the interplay between strong regularity and graph densification, in: Graph-Based Representations in Pattern Recognition - 11th

21

IAPR-TC-15 International Workshop, GbRPR 2017, Anacapri, Italy, May 16-18, 2017, Proceedings, 2017, pp. 165–174.

[8] M. Pelillo, I. Elezi, M. Fiorucci, Revealing structure in large graphs: Szemerdis regularity lemma and its use in pattern recognition, Pattern Recognition Letters 87 (2017) 4 – 11, advances in Graph-based Pattern Recognition.

[9] A. Sperotto, M. Pelillo, Szemerédi's regularity lemma and its applications to pairwise clustering and segmentation, in: Energy Minimization Methods in Computer Vision and Pattern Recognition, 6th International Conference, EMMCVPR 2007, Ezhou, China, August 27-29, 2007, Proceedings, 2007, pp. 13–27.

[10] S. E. Schaeffer, Graph clustering, Computer Science Review 1 (1) (2007) 27 – 64.

[11] M. E. J. Newman, M. Girvan, Finding and evaluating community structure in networks, Phys. Rev. E 69 (2) (2004) 026113.

[12] K. LeFevre, E. Terzi, Grass: Graph structure summarization, Proceedings of the 2010 SIAM International Conference on Data Mining (2010) 454–465.

[13] M. Riondato, D. García-Soriano, F. Bonchi, Graph summarization with quality guarantees, Data Min. Knowl. Discov. 31 (2) (2017) 314–349.

[14] Y. Liu, A. Dighe, T. Safavi, D. Koutra, Graph Summarization: A Survey, ACM Computing Surveys V (N).

[15] J. R. Ullmann, An algorithm for subgraph isomorphism, J. ACM 23 (1) (1976) 31–42.

[16] V. Carletti, P. Foggia, A. Saggese, M. Vento, Challenging the time complexity of exact subgraph isomorphism for huge and dense graphs with vf3, IEEE Transactions on Pattern Analysis and Machine Intelligence 40 (4) (2018) 804–818.

[17] Y. Liang, P. Zhao, Similarity search in graph databases: A multi-layered indexing approach, in: 2017 IEEE 33rd International Conference on Data Engineering (ICDE), 2017, pp. 783–794.

[18] W. Zheng, L. Zou, X. Lian, D. Wang, D. Zhao, Efficient graph similarity search over large graph databases, IEEE Transactions on Knowledge and Data Engineering 27 (4) (2015) 964–978.

[19] W. Zheng, L. Zou, X. Lian, D. Wang, D. Zhao, Graph similarity search with edit distance constraint in large graph databases, in: Proceedings of the 22Nd ACM International Conference on Information & Knowledge Management, CIKM '13, ACM, 2013, pp. 1595–1600.

[20] S. Zhang, J. Yang, W. Jin, Sapper: Subgraph indexing and approximate matching in large graphs, Proc. VLDB Endow. 3 (1-2) (2010) 1185–1194.

[21] T. Gowers, Lower bounds of tower type for Szemerédi's uniformity lemma, Geom Func. Anal. 7 (2) (1997) 322–337.

[22] G. Sárközy, F. Song, E. Szemerédi, S. Trivedi, A prectical regularity partitioning algorithm and its applications in clustering, arXiv:1209.6540v1 [math.CO] (2012).

[23] Y. Jin, J. F. JáJá, Network summarization with preserved spectral properties, CoRR abs/1802.04447.

[24] E. R. van Dam, W. H. Haemers, Which graphs are determined by their spectrum?, Linear Algebra and its Applications 373 (2003) 241 – 272, combinatorial Matrix Theory Conference (Pohang, 2002).

[25] J. Leskovec, A. Krevl, SNAP Datasets: Stanford large network dataset collection, `http://snap.stanford.edu/data` (Jun. 2014).

[26] J. Kunegis, KONECT – The Koblenz Network Collection, in: Proc. Int. Conf. on World Wide Web Companion, 2013, pp. 1343–1350.

[27] J. Leskovec, J. J. Mcauley, Learning to Discover Social Circles in Ego Networks, in: F. Pereira, C. J. C. Burges, L. Bottou, K. Q. Weinberger (Eds.), Advances in Neural Information Processing Systems 25, Curran Associates, Inc., 2012, pp. 539–547.

[28] H. Yin, A. R. Benson, J. Leskovec, D. F. Gleich, Local Higher-Order Graph Clustering, in: Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining - KDD '17, 2017, pp. 555–564.

[29] J. Leskovec, J. Kleinberg, C. Faloutsos, Graph Evolution: Densification and Shrinking Diameters, ACM Trans. Knowl. Discov. Data 1 (1).

[30] Openflights network dataset – KONECT (Sep. 2016).

[31] G. Joshi-Tope, M. Gillespie, I. Vastrik, P. D'Eustachio, E. Schmidt, B. de Bono, B. Jassal, G. R. Gopinath, G. R. Wu, L. Matthews, S. Lewis, E. Birney, L. Stein, Reactome: A knowledgebase of biological pathways, Nucleic Acids Research 33 (DATABASE ISS.).

[32] J. Fox, L. M. Lovász, Y. Zhao, A fast new algorithm for weak graph regularity, CoRR abs/1801.05037 (2018) 1–13.

[33] H. Reittu, L. Leskela, T. Raty, M. Fiorucci, Analysis of large sparse graphs using regular decomposition of graph distance matrices, in: 2018 IEEE International Conference on Big Data (Big Data),

2018, pp. 3784–3792.

[34] H. Bunke, K. Riesen, Recent advances in graph-based pattern recognition with applications in document analysis, Pattern Recognition 44 (5) (2011) 1057 – 1067.

[35] M. Vento, A long trip in the charming world of graphs for pattern recognition, Pattern Recogn. 48 (2) (2015) 291–301.