Community-Preserving Generalization of Social Networks

Jordi Casas-Roma
Universitat Oberta de Catalunya
Barcelona, Spain
Email: jcasasr@uoc.edu

François Rousseau
LIX, École Polytechnique
Palaiseau, France
Email: rousseau@lix.polytechnique.fr

***Abstract*—In this paper, we tackle the problem of graph generalization in the context of privacy-preserving social network mining. By grouping together nodes that are not only similar but that also belong to the same k-shells, we better preserve the community structure of the graph, its utility in case of clustering-related applications, while still achieving some privacy level through the concept of graph generalization. We conduct empirical evaluations of our approach on synthetic and real social network data, demonstrating its utility and practical application.**

I. Introduction

With the tremendous amount of social network data recently made publicly available, privacy-preserving graph mining has become an important research issue. Indeed, before releasing these data to third parties and to the research community for subsequent analysis, it is necessary to protect the users' privacy in the first place through network anonymization techniques. Backstrom *et al.* [2] pointed out that the simplest technique for anonymizing graphs, which consists of removing the identities of the vertices before publishing the actual graph, does not always guarantee privacy because of the untouched topology of the network than can help re-identify the nodes. Therefore, developing more complex network anonymization techniques has become an important issue these past few years. To anonymize a network, one can change its topology (e. g., by adding or deleting edges) while keeping the original nodes or one can propose a summary of the graph where nodes, and a fortiori edges, have been merged into super-nodes and super-edges, technique known as *graph generalization* or graph summarization.

In this paper, we consider the second scenario and propose a novel approach to generalize a graph in order to preserve the user's privacy while retaining as much data utility, specifically the community structure in the context of clustering-specific graph mining tasks. Additionally, graph generalization is also used for summarization and dimensionality reduction in order to represent networks in a small space, so that they can be used effectively for indexing and retrieval. Furthermore, compressed graphs can be used in a variety of applications in which it is desirable to use the summary behavior in order to estimate the approximate structural properties of the network. Therefore, while our method has been initially developed for privacy-preserving graph mining, it has many more applications where maintaining the community structure would be of interest.

A. Our contributions

We developed an approach to generalize undirected and unlabeled networks. Since these graphs have no attributes nor labels on the edges, information is contained only in the structure of the graph itself – its topology – and therefore, preserving the network's structure is critical to reduce the information loss. We will present various methods to compute the similarity between vertices and define the partitions in which the vertices will be merged. The contributions of the proposed approach can be summarized as follows:

- We define a novel approach to generalize a graph by capitalizing on the concept of graph degeneracy and on the similarity between vertices of the same k-shell.
- We propose four different methods to compute the similarity between vertices.
- We conduct an empirical evaluation of these methods on several synthetic and real networks, comparing information loss based on different graph properties and also on clustering-specific processes.
- We demonstrate that our approach preserves data privacy while simultaneously achieving better data utility through the generalization process.

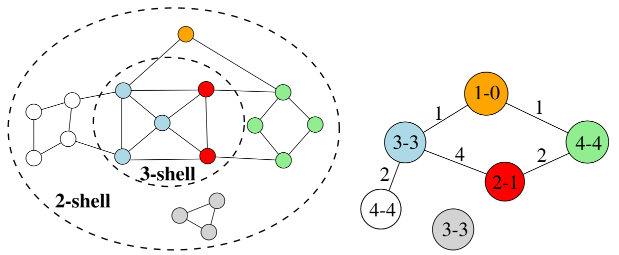
B. Outline

The rest of the paper is organized as follows. Section II reviews the state-of-the-art on graph generalization methods. Section III defines the preliminary concepts related to our approach. Section IV introduces our proposed algorithm for generalizing a graph while preserving its community structure. Section V describes the experimental framework and discusses the results we obtained in terms of information loss and data utility. Finally, Section VI concludes our paper and mentions future work directions.

II. Related work

Generalization approaches, also known as clustering-based approaches, can be essentially regarded as grouping vertices and edges into partitions called *super-vertices* and *super-edges*. The details about individuals can be hidden properly, but the graph may be shrunk considerably after anonymization,

Jordi Casas-Roma was partially supported by the Spanish MCYT and the FEDER funds under grants TIN2011-27076-C03 "CO-PRIVACY" and TIN2014-57364-C2-2-R "SMARTGLACIS".



(a) Original graph (b) Generalized graph

Fig. 1: Toy example generalization process.

which may not be desirable for analyzing local structures. The generalized graph, which contains the link structures among partitions as well as the aggregate description of each partition, can still be used to study macro-properties of the original graph. Even if it holds the properties of the original graph, it does not have the same granularity. It is interesting to underline that generalization approaches also preserve attribute and link disclosure, since two vertices from any cluster are indistinguishable based on either their relationships or their attributes.

Figure 1a shows a toy example graph, consisting on 17 vertices and 25 edges distributed into 2 k-shells (see Section III-A for definition). Node color indicates similar vertices, according to some arbitrary vertex similarity measure. A generalized version of the same graph, Figure 1b, contains 6 super-vertices and 5 super-edges. Each super-vertex incorporates the number of intra-vertices (first digit) and intra-edges (second digit).

Hay et al. [17] applied structural generalization approaches using the size of a partition to ensure vertex k-anonymity [28]. Their method obtains a vertex k-anonymous super-graph by clustering vertices into super-vertices and edges into super-edges. Each super-vertex represents at least k nodes and each super-edge represents all the edges between vertices in two super-vertices. Zheleva and Getoor [30] focused on the problem of preserving the privacy of sensitive relationships in graph data. Nerggiz and Clifton [20] presented a methodical approach to evaluate clustering-based k-anonymity algorithms using different metrics and attempted to improve precision by ignoring restrictions on generalization approaches.

Campan and Truta [9] worked on undirected networks with labeled vertices and unlabeled edges. Vertices attributes contain identifiers, quasi-identifiers and sensitive attributes. The k-anonymity model is applied to quasi-identifiers in order to achieve indistinguishable vertices from their attributes or relationships between attributes. The authors developed a method, called SaNGreeA, designed to anonymize structural information. Then, Ford et al. [15] introduced an extension to k-anonymity model that adds the ability to protect against attribute disclosure. They also presented a new algorithm, based on SaNGreeA, to enforce p-sensitive k-anonymity on social network data based on a greedy clustering approach. He et al. [18] utilized a similar anonymization method that partitions the network in a manner that preserves as much of the structure of the original social network as possible.

Bhagat et al. [5] assumed that adversaries know part of the links and nodes in the graph. The authors pointed out that merely grouping nodes into several classes cannot guarantee the privacy. For instance, one can considers the case where the nodes within one class form a complete graph via a certain interaction. Then, once the adversary knows the target is in the class, he can be sure that the target must participate in the interaction. The authors provided a safety condition to ensure that the pattern of links between classes does not leak information.

More recently, Singh and Schramm [25] took the generalization concept further and create a generalized trie structure that contains information about network sub-graphs and neighborhoods. Cormode et al. [13] studied the anonymization problem on bipartite networks. Their anonymization method can preserve the graph structure exactly by masking the mapping from entities to nodes rather than masking or altering the graph structure. Stokes and Torra [26] presented two methods for graph partitioning using similarity measures to create clusters which group vertices into partitions of k or more elements.

Finally, Sihag [24] presented a method for k-anonymization via generalization on undirected and unlabeled graphs. The author chose genetic algorithms to optimize this NP-hard problem, but this method does not seem scalable for medium or large networks.

III. Preliminary concepts

Let $G = (V, E)$ be a simple, undirected and unlabeled graph, where V is the set of vertices and E the set of edges in G. We define $n = |V|$ to denote the number of vertices and $m = |E|$ to denote the number of edges. We will use (i, j) to refer to an undirected edge from vertex v_i to v_j and $deg(v_i)$ to denote the degree of vertex v_i, i. e. its number of neighbors. Finally, we will designate by $G = (V, E)$ and $\widetilde{G} = (\widetilde{V}, \widetilde{E})$ when referring to a pair of original and generalized graphs respectively.

Our graph generalization approach relies on two main concepts related to graph structure and vertex similarity. The former corresponds to the concept of *graph degeneracy*, specifically the definition of a k-shell, and the latter involves the computation of the similarity between vertices on unlabeled networks.

A. Graph degeneracy and k-shell

The idea of a *k-degenerate* graph comes from the work of Bollobás [7, page 222] that was further extended by Seidman [23] into the notion of a *k-core*, which explains the use of *degeneracy* as an alternative denomination for k-core in the literature. Henceforth, we will be using the two terms interchangeably.

Let k be an integer. A subgraph $H_k = (V', E')$, induced by the subset of vertices $V' \subseteq V$ (and a fortiori by the subset of edges $E' \subseteq E$), is called a *k-core* if and only if $\forall v_i \in V'$, $deg_{H_k}(v_i) \geq k$ and H_k is the maximal subgraph with this property, i. e. it cannot be augmented without losing this property. In other words, the k-core of a graph corresponds to the set of maximal connected subgraphs whose vertices are at least of degree k within the subgraph.

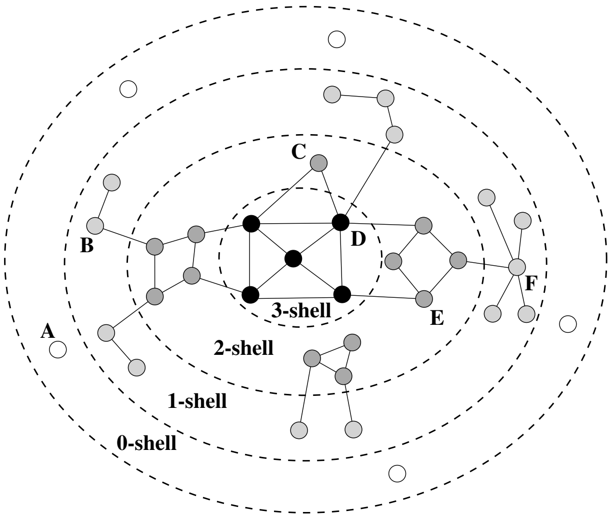


Fig. 2: Illustration of a graph G and its decomposition in disjoint k-shells.

From the k-core, Carmi *et al.* [10] defined the notion of *k-shell*, which corresponds to the subgraph induced by the set of vertices that belong to the k-core but not the $(k+1)$-core, denoted by $\mathcal{S}_k$ such that $\mathcal{S}_k = \{v_i \in G,\ v_i \in H_k \wedge v_i \notin H_{k+1}\}$

The *core number* of a vertex v_i is the highest order of a core that contains this vertex, denoted by $core(v_i)$. It is also referred as the *shell index* since the k-shell is exactly the part of the k-core that will not survive in the $(k+1)$-core. Basically, its value corresponds to how cohesive one's neighborhood is and is a measure of user engagement in a network [19]. Indeed, to belong to a k-core, a node needs at least k neighbors also meeting the same requirements, thus forming a community of "close" nodes. Again, in the case of a social network, the core number of a node would correspond to the number of close friends the user has, his inner circle that would collapse if he were to leave (through the *cascading effect* implied by the k-core condition – see the impact of the removal of node D in the 3-shell of Figure 2 for instance). Thanks to Batagelj and Zaveršnik [4], the shell index sequence of an unlabeled graph can be efficiently computed in linear time ($\mathcal{O}(n+m)$) and space ($\mathcal{O}(n)$).

Figure 2 illustrates the decomposition of a given graph G of 34 vertices and 36 edges into disjoints shells and nested cores of order 0, 1, 2 and 3. Node color indicates the shell a vertex belongs to: white for the 0-shell, light gray for the 1-shell, dark gray for the 2-shell and black for the 3-shell.

B. Vertex similarity measures

Since we are interested in grouping vertices into super-vertices, we have to define a measure of vertex similarity in order to find and select a group of vertices that shares structural properties in the network. In particular, we considered Manhattan and 2-path similarities for clustering of the neighbor sets of the vertices of a graph.

The *Manhattan similarity* [27], based on the Manhattan distance, measures how many equal neighbors the two vertices share but also how many non-neighbors they share. It is computed as follows:

$$Sim_{Manhattan}(v_i, v_j) = 1 - \frac{1}{n}\sum_{k=1}^{n} |(v_i, v_k) - (v_j, v_k)| \quad (1)$$

where $(v_i, v_k) = 1$ if $(v_i, v_k) \in E$ and $(v_i, v_k) = 0$ otherwise.

The *2-path similarity* measures the number of paths of length 2 between two vertices, as the following equation depicts:

$$Sim_{2\text{-}path}(v_i, v_j) = \frac{1}{n}\sum_{k=1}^{n}(v_i, v_k)(v_j, v_k) \quad (2)$$

where $(v_i, v_k) = 1$ if $(v_i, v_k) \in E$ and $(v_i, v_k) = 0$ otherwise.

Both similarity measures score in the range $[0, 1]$, where 0 indicates no similarity at all and 1 the maximum similarity score. As noted by Stokes and Torra [26], the Manhattan similarity measures the similarity between vertices with respect to both neighbors and non-neighbors, while the 2-path similarity only measures the similarity between vertices with respect to their neighbors, so that a common non-neighbor does not change the similarity between two vertices.

IV. Graph generalization algorithm

In this section, we present our approach designed to generalize a graph while preserving its community structure. As aforementioned, these methods can be useful for diverse purposes and we focused here on privacy-preserving data mining. Specifically, we are interested in preserving the users' privacy while keeping most of the data utility, in particular for clustering and community detection processes.

As stated previously, our approach relies on two definitions: *graph degeneracy* and *vertex similarity*. Firstly, we evaluate communities based on the concept of k-core as an efficient mean to evaluate their collaborative nature – a property not captured by the single-node metrics or by the established community evaluation metrics. We considered the graph as a set of disjoint k-shells. Such a partition provides a way to preserve the graph's clustering structure, while making the execution of the similarity measure between vertices much faster due to the smaller size of the graph's partitions. Secondly, we consider various *vertex similarity* metrics to compute the distance between vertices. Due to the fact that generalization implies merging vertices into one super-vertex, defining a similarity metric is a critical step to maintain the data utility and reduce the information loss. Here, we evaluated two similarity metrics, the Manhattan and 2-path similarities, and we also used two well-known clustering algorithms to provide groups of vertices.

Our approach is based on a three-step algorithm:

1) *Information gathering* – this step collects two types of information for every vertex in the original graph. First, we decompose the graph and assign the *shell index* to each vertex. Then, we utilize a similarity metric to define groups of one or more vertices. Vertices in the same group share some structural properties depending on the similarity metric used.

2) *Super-vertex definition* – using the information collected in the previous step, we can define which vertices will be merged into each super-vertex.

3) *Generalized graph creation* – once the super-vertices are defined, we (1) create an empty graph; (2) add the super-vertices and also the information contained in them, i. e. the number of intra-vertices and intra-edges; and finally (3) add the inter-edges between super-vertices in the generalized network.

A. Step 1 – Information gathering

The first step focuses on collecting the information needed in the next one to define the partition groups. Following the intuitions previously mentioned, we considered that a partition group must contains vertices, which (1) belong to the same k-shell of the original graph, since it will preserve the graph decomposition and also the clustering structure; and (2) share some properties regarding graph's structure, thus similarity metrics have to defined and evaluated to reduce the information loss.

Obviously, there are many ways to define and compute the similarity between vertices in a network. We have chosen two well-known vertex similarity metrics and also two graph clustering algorithms, which do not provide a similarity measure but a group partition of the whole network.

Our similarity metrics are the *Manhattan* and *2-path* similarity presented in Section III-B. As described, these metrics score a mark between 0 and 1, where 0 means no similarity at all and 1 is the maximum similarity between two vertices. According to these scores, our approach selects the maximum mark between two vertices or groups of vertices and merges them into the same group. This process is iterated until a stop condition is reached – we have used the *contraction percentage* (p), defined as the number of partitions over the total number of vertices. For instance, $p = 0.5$ implies creating, at least, $n/2$ groups of vertices.

Due to the importance of the group partitions in our approach, we also wanted to test some clustering or community detection algorithms to define the vertices' partitions. They do not provide a score, just a group partition over the whole graph. Thus, it is possible to utilize any graph clustering or community detection algorithm in this step. We have chosen the *Multilevel* [6] and *Fastgreedy* [12] algorithms (see Section V-A3 for further details).

B. Step 2 – Super-vertex definition

This second step focuses on defining the super-vertices according to the previously collected information for each vertex. For each k-shell in the graph, we merge vertices belonging to the same group partition into the same super-vertex. Additionally, we have defined a parameter to avoid merging too many vertices into one super-vertex; if a max_fusion parameter is reached, then we split the super-vertex onto two independent super-vertices. As a result of this step, a set of super-vertices is defined and each vertex is assigned to one, and only one, super-vertex.

C. Step 3 – Generalized graph creation

Finally, the third step focuses on creating the new generalized graph according to the super-vertices defined in the previous step. We start by defining an empty, undirected, edge-labeled and vertex-labeled graph $\widetilde{G} = (\widetilde{V}, \widetilde{E})$. Then, the process iterates by adding each previously defined super-vertex $sv_i \in \widetilde{V}$. Each super-vertex contains information about the number of vertices, which have merged into this super-vertex (*IntraVertices*) and also the number of edges between the vertices contained in it (*IntraEdges*). Lastly, a super-edge between two super-vertices is created if there exists an edge between two vertices contained in each of the super-vertices, i. e. $(sv_i, sv_j) \in \widetilde{E} \leftrightarrow (v_k, v_p) \in E : v_k \in sv_i \wedge v_p \in sv_j$. Super-edges contain a label indicating the number of edges between all vertices from their endpoints (*InterEdges*).

V. Experiments

In this section, we first describe the experimental framework we used to analyze and compare the information loss induced by our generalization methods. We then present the results we obtained on five network datasets.

A. Experimental framework

For each dataset, we compute the generalized graph produced by *Multilevel* and *Fastgreedy* algorithms. As aforementioned, the generalized graph is unique for each clustering algorithm. On the contrary, using our two similarity measures, we can adjust some parameters to control the granularity of the summarized graph. In these experiments, we set the *contraction percentage* to 0.5, i. e. the number of super-vertices is half the number of vertices.

1) Tested networks: We used both synthetic and real networks in our experiments. We used the *igraph*[1] library to generate two kinds of random graphs:

- *ER-1000* – Erdös-Rényi Model [14] is a classical random graph model. It defines a random graph as n vertices connected by m edges that are chosen randomly from the $n(n-1)/2$ possible edges. In our experiments, we set n to 1,000 and m to 5,000.
- *BA-1000* – Barabási-Albert Model [3], also called scale-free model, is a network whose degree distribution follows a power law. That is, for degree d, its probability density function is $P(d) = d^{-\gamma}$. In our experiments, we set the number of vertices to be 1,000 and γ=1, i. e. linear preferential attachment.

Additionally, three different real networks have been used in our experiments. Although all these sets are unlabeled, we have selected these datasets because they have different graph's properties. They are the following ones:

- *Karate* – Zachary's Karate Club [29] shows the relationships among 34 members of a karate club.
- *Polblogs* – Political blogosphere data [1] compiles the data on the links among US political blogs.
- *URV email* – the email communication network at the University Rovira i Virgili in Tarragona (Spain) [16].

[1] Available at: http://igraph.org/

2) Generic information loss evaluation: We describe the criteria that are used to quantify the information loss on summarized graphs. It is important to underline that since the number of vertices, edges and some important graph's structures change during generalization process, some metrics cannot be used to compare information between original and generalized graphs. For instance, centrality measures, like betweenness or closeness centrality, have been proved to correctly assess information loss and data utility [11], but they evaluate each vertex independently and consequently, score obtained on original and summarized graphs cannot be compared.

For each experiment, we provide information about the number of vertices and edges both in the original and summarized graphs. We also considered the degree distribution as an important feature we must preserve. Additionally, some graph's structural metrics have been included in our experiments. The first one is the *average distance* ($\overline{dist}$), which is defined as the average of the distances between each pair of vertices in the graph. The *diameter* (d) is defined as the largest minimum distance between two vertices in the graph, and *harmonic mean of the shortest distance* (h) is an evaluation of connectivity, similar to the average distance or average path length, and its inverse is also known as the global efficiency. Finally, *transitivity* (T) is one type of clustering coefficient, which measures and characterizes the presence of local loops near a vertex.

The above measures evaluate the entire graph as a unique score. We compute the error on these graph metrics as follows:

$$\epsilon_m(G, \widetilde{G}) = |m(G) - m(\widetilde{G})|, \tag{3}$$

where m is one of the graph metrics defined above, G is the original graph and $\widetilde{G}$ is the generalized graph.

3) Clustering-specific evaluation: Variations in the generic graph properties is a good way to assess the information loss but they have their limitations because they are just a proxy to the changes in data utility we actually want to measure. We define the specific information loss measures as a task-specific measure for quantifying the data utility and the information loss associated to a data publishing process. We focus on clustering-specific processes, since it is an important application for social and healthcare networks, among many others. Like generic graph measures, we compare the results obtained both by the original and the generalized graphs in order to quantify the level of noise introduced during the generalization process. This measure is specific and application-dependent, but it is necessary to test the generalized data in real graph-mining processes.

We considered the following approach to measure the clustering assessment for a particular generalization and clustering method (illustrated in Figure 3): (1) apply each of our generalization methods to the original graph G and obtain the generalized version $\widetilde{G}$; (2) apply a particular clustering method c to G and obtain clusters $c(G)$ and apply the same method to $\widetilde{G}$ to obtain $c(\widetilde{G})$; (3) compare the clusters $c(G)$ to $c(\widetilde{G})$. In relation to information loss, it is clear that the more similar $c(\widetilde{G})$ is to $c(G)$, we have the less information loss. Thus, clustering specific information loss measures should evaluate the divergence between both sets of clusters $c(G)$ and $c(\widetilde{G})$.

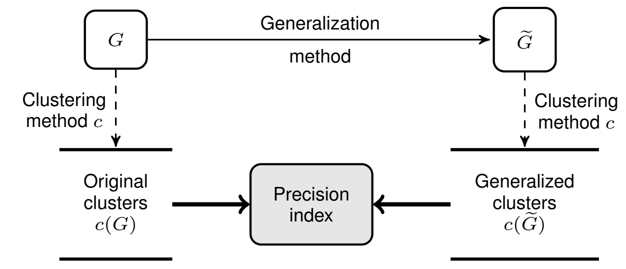


Fig. 3: Framework for evaluating the clustering-specific information loss measure.

Each super-vertex contains information regarding vertices that have been grouped into it. Thus, we are able to compute the original vertex distribution on the clusters of the generalized graph. Ideally, the results should be the same. That is, the same number of sets (i. e. clusters) with the same original vertices in each set. In this case, we can say that the generalization process has not affected the clustering process. When the sets do not match, we should be able to compute a measure of divergence. For this purpose, we use the *precision index* [8]. Assuming that we know the true communities of a graph, the precision index can be directly used to evaluate the similarity between two cluster assignments. Given a graph of n nodes and q true communities, we assign to nodes the same labels $l_{tc}(\cdot)$ as the community they belong to. In our case, the true communities are the ones assigned on the original dataset (i. e. $c(G)$), since we want to obtain communities as close as the ones we would get on non-generalized data. Assuming that the generalized graph has been divided into clusters (i. e. $c(\widetilde{G})$), then for every cluster, we examine all the vertices within each super-vertex and assign to them as predicted label $l_{pc}(\cdot)$ the most frequent true label in that cluster (basically the mode). Then, the precision index can be defined as follows:

$$precision(G, \widetilde{G}) = \frac{1}{n} \sum_{i=1}^{n} \mathbb{1}_{l_{tc}(v_i) = l_{pc}(v_i)}, \tag{4}$$

where $\mathbb{1}$ is the indicator function such that $\mathbb{1}_{x=y}$ equals 1 if $x = y$ and 0 otherwise. Note that the precision index is a value in the range $[0, 1]$, which takes the value 0 when there is no overlap between the sets and the value 1 when the overlap between the sets is complete.

We have used four graph clustering algorithms to evaluate the community structure. They are the following ones:

- *Girvan-Newman* (or *GN*) [21] is an important community detection algorithm in graphs. It is a hierarchical divisive algorithm, in which edges are iteratively removed based on the value of their betweenness centrality.
- *Multilevel* [6] is a multi-step technique based on a local optimization of Newman-Girvan modularity in the neighborhood of each node. After a partition is identified in this way, communities are replaced by super-nodes, yielding a smaller weighted network. The procedure is then iterated, until modularity does not increase any further.

TABLE I: Generalization results using our four grouping methods: *Manhattan* and *2-path* similarity metrics, and *Multilevel* and *Fastgreedy* graph clustering algorithms. For each dataset and method, we compare the results obtained on n, m, $\overline{dist}$, d, h, T and precision index using Multilevel (*ML*), Infomap (*IM*), Fastgreedy (*FG*) and Girvan-Newman (*GN*) clustering algorithms.

Network	**Method**	n	m	$\overline{dist}$	d	h	T	*ML*	*IM*	*FG*	*GN*
		1,000	4,969	3.263	5	3.083	0.010	-	-	-	-
	Manhattan	672	4,828	2.672	5	2.514	0.051	0.147	0.031	0.243	0.296
ER-1000	2-Path	612	4,782	2.617	5	2.451	0.070	0.150	0.040	0.215	0.311
	Multilevel	135	2,772	1.741	4	1.549	0.489	0.394	0.036	0.229	0.189
	Fastgreedy	119	2,766	1.630	3	1.443	0.531	0.147	0.030	0.544	0.182
		1,000	4,985	2.481	4	2.362	0.032	-	-	-	-
	Manhattan	483	4,383	2.144	4	2.047	0.108	0.157	1.000	0.185	0.433
BA-1000	2-Path	436	2,690	1.991	3	1.957	0.070	0.132	1.000	0.176	0.321
	Multilevel	106	1,728	1.689	3	1.526	0.457	0.618	1.000	0.184	0.374
	Fastgreedy	104	1,682	1.685	2	1.522	0.451	0.176	1.000	0.477	0.374
		34	78	4.588	5	2.032	0.25	-	-	-	-
	Manhattan	25	62	2.023	3	1.769	0.335	0.705	0.500	0.676	0.676
Karate	2-Path	23	40	2.122	3	1.878	0.242	0.794	0.500	0.617	0.705
	Multilevel	10	14	1.977	4	1.636	0.375	0.823	0.647	0.911	0.617
	Fastgreedy	9	16	1.694	3	1.430	0.589	0.794	0.500	0.941	0.352
		1,222	16,714	2.737	8	2.519	0.225	-	-	-	-
	Manhattan	866	13,229	2.408	5	2.248	0.245	0.830	0.833	0.853	0.646
Polblogs	2-Path	1,048	9,086	2.575	7	2.410	0.148	0.950	0.959	0.985	0.823
	Multilevel	171	3,071	1.944	6	1.737	0.532	0.993	0.517	0.967	0.767
	Fastgreedy	169	3,062	1.944	6	1.733	0.536	0.976	0.520	0.973	0.740
		1,133	5,451	3.606	8	3.334	0.166	-	-	-	-
	Manhattan	745	5,274	2.886	6	2.683	0.149	0.420	0.463	0.533	0.352
URV email	2-Path	944	4,444	3.334	7	3.091	0.135	0.586	0.682	0.555	0.517
	Multilevel	160	1,710	2.179	6	1.931	0.386	0.781	0.134	0.601	0.290
	Fastgreedy	157	1,763	2.187	6	1.922	0.420	0.468	0.147	0.862	0.217

- *Infomap* [22] optimizes the map equation, which exploits the information-theoretic duality between the problem of compressing data and the problem of detecting significant structures in the graph.
- *Fastgreedy* [12] is a hierarchical agglomeration algorithm for detecting community structure. Starting from a set of isolated nodes, the edges of the original graph are iteratively added to produce the largest possible increase of the modularity at each step.

B. Results

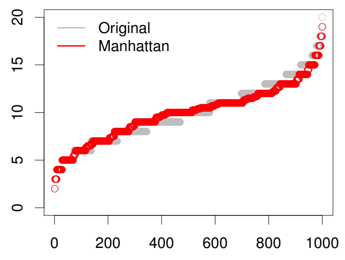
In this section, we present the results of our generalization approach in terms of data utility and information loss. We considered both generic information loss measures (detailed in Section V-A2) and also information loss regarding clustering-specific graph-mining tasks (described in Section V-A3).

Results are given in Table I. Each cell indicates the value for the corresponding measure and method. The first row of each dataset points out the values obtained on the original network, previous to the generalization process and with no alteration. Although deviation is undesirable, it is inevitable due to the graph generalization process.

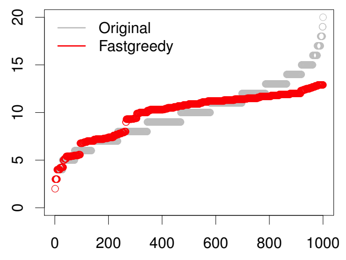
The first two tested networks are the synthetic ones. As we have commented previously, ER-1000 has been created using the Erdös-Rényi model. Its degree distribution does not follow a power law. Figure 4a presents the degree distribution of the original graph (grey) and the reconstructed version of the generalized graph using the Manhattan similarity (red). As we can see, most of the vertices in the original network have degree values between 7 and 13, while only few have degree values lower than 7 or higher than 13.

The reconstructed version of the degree distribution uses only information from the generalized graph. As we described, two labels inside each super-vertex inform about the number of vertices and the number of intra-edges, and the super-edges contains a label indicating the number of edges between the vertices inside each endpoint of the super-edge. Thus, we can compute the average degree of the vertices inside each super-vertex and reconstruct the degree distribution. Due to the fact that the degree distribution, and its directly related properties, are important for some analysis or graph-mining tasks, we have considered interesting to analyze the reconstructed degree distribution in our experiments.

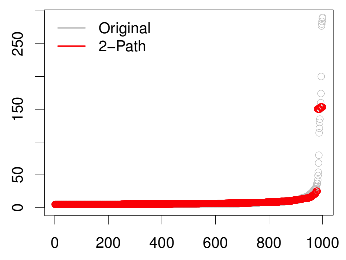
The original ER-1000 network has 1,000 vertices, while the generalized ones contain about 600 super-vertices when Manhattan and 2-path similarity methods are used with a *contraction percentage* $p = 0.5$, while the number of super-vertices fall to 135 and 119 when the clustering algorithms are utilized to create the partition groups. Obviously, it has an important impact on the reconstructed degree distribution as we can see in Figure 4b. Comparing this figure to the previous one, we can clearly point out that the generalized graph using Manhattan similarity better reconstructs the degree distribution. However, we must take into account that the generalized graph using Fastgreedy contains fewer super-



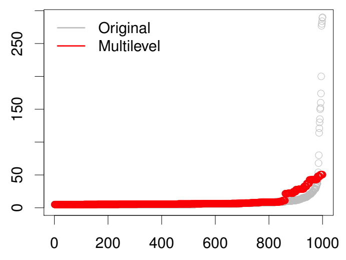
(a) Manhattan (ER-1000)



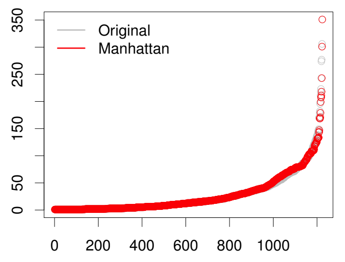
(b) Fastgreedy (ER-1000)



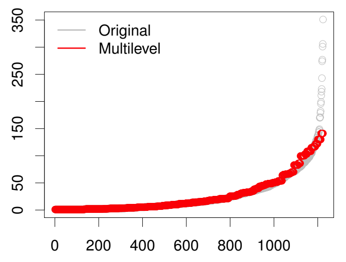
(c) 2-Path (BA-1000)



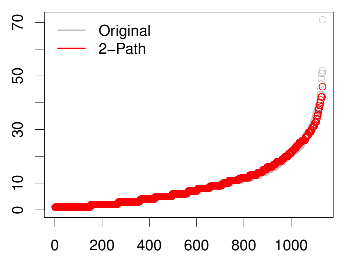
(d) Multilevel (BA-1000)



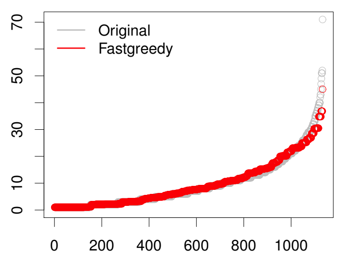
(e) Manhattan (Polblogs)



(f) Multilevel (Polblogs)



(g) 2-Path (URV email)



(h) Fastgreedy (URV email)

Fig. 4: Comparison between the original and the reconstructed degree distributions after generalization process. Horizontal axis indicates the vertex set and the vertical one corresponds to the degree of each vertex.

vertices, and it affects the reconstruction process. Comparing the generic information loss measures in Table I, we can state that the information loss on generalized graphs by Manhattan and 2-path similarity measures are quite similar, while it is also similar on generalized graphs by clustering algorithms though their values are not as close to the original ones.

Regarding the clustering-specific information loss, we used the precision index, as previously described, and its results are presented on the four last columns in Table I. Regarding the ER-1000 network, our approach achieves moderate and low precision score values.

The second synthetic network, BA-1000, has been constructed by applying scale-free model and its degree distribution follows a power law. Figure 4c and 4d point out clearly a large number of vertices with small degree value and few vertices with very high degree value. It is important to underline the scale difference between this figure and the previous one. Like in the previous dataset, the generalized graph using Manhattan similarity gets a better-reconstructed degree distribution. Regarding the generic information loss measures, generalized graph by Manhattan similarity carries out the best results, i. e. the closest ones to the original values. Nevertheless, results on our real graph mining tasks are alike across our four generalization methods. It is interesting to point out that precision values on Infomap are equal to 1 for all tested methods, indicating a perfect matching between original and generalized clusters. Since this network presents a few important hubs and all other vertices attached to them, the Infomap clustering algorithm creates only one partition, which contains all vertices. The behavior is the same on generalized graphs and consequently, the precision index matching is complete.

The first tested real network, and also the smallest one, is Zachary's Karate Club. Like in the previous cases, the generalized graphs obtained using Multilevel and Fastgreedy algorithms are more compressed than the ones obtained using Manhattan and 2-path methods. Nonetheless, the generic information loss measures are closely similar for all these methods. Precision index values are much better than the previous analysis, reaching several values close to 0.8 or 0.9. All methods scores values between 0.7 and 0.82 on precision index using Multilevel clustering algorithm and between 0.62 and 0.94 on precision index using Fastgreedy algorithm. Infomap and Girvan-Newman algorithms obtain lower values, but still between 0.5 and 0.7 in almost all methods.

Polblogs is our second tested real network. Figures 4e and 4f show the degree distribution. As can be seen, the Manhattan similarity method is able to reconstruct the hubs of the original network, due to the fact that the compression ratio is lower than the generalized graph produced by the Multilevel algorithm. The best values on average distance, harmonic mean of the shortest distance and transitivity are also achieved by the Manhattan generalized graph. In reference to the clustering-specific information loss, we get the best general values of all tested networks. Precision index marks are between 0.8 and 0.99 in almost all experiments, using all our methods. It is interesting to underline the results of 2-path similarity method, which carries out values in range 0.82-0.98. However, the number of super-vertices is large, which means the compression is low. Alternatively, the compression is much larger using the Multilevel and Fastgreedy methods, and precision index marks are still high – between 0.74 and 0.99 in almost all cases.

Lastly, URV email is the third real network tested in our experiments. Regarding the degree distribution, the absence of hubs with very high degree is relevant. In this case, as we can see in Figures 4g and 4h, the generalized graph using Fastgreedy algorithm obtains a reconstructed degree distribution close to the original one. Even though it is true that the reconstructed degree distribution obtained by 2-path similarity method is better, the former uses only 157 super-vertices while the later utilizes 944 super-vertices. Probably due to this, the 2-path method achieves the closest results on average distance, diameter and harmonic mean of the shortest distance. In reference to the clustering-specific information loss, the results are moderate. 2-path method ranges from 0.52 to 0.68, while Manhattan method gets values on range 0.35-0.53. The generalized graph using Multilevel algorithm scores 0.78 when the same clustering algorithm is used to compute the precision index. It is obvious that the matching will be higher if we use the same clustering algorithm to create vertex's partitions and also to compute the precision index. Similar behavior occurs when using Fastgreedy generalized graph.

VI. Conclusions

In this paper, we have presented an approach to generalize or summarize a graph while preserving its communities. This approach relies on two intuitions: first, the k-shells in the graph decomposition are related to community structures, thus preserving the shells on generalization process would improve the clustering-specific graph-mining tasks; and second, a similarity measure have to be used to create partitions of vertices inside each k-shell. We have introduced four methods to compute the similarity among vertices in a graph or a shell. An empirical evaluation of these methods have been conducted on several synthetic and real networks, comparing information loss based on different graph properties and also on clustering-specific information loss. We have demonstrated that our methods are able to generalize a graph while preserving the most important features and maintaining data utility on clustering-specific graph mining tasks. As we have seen throughout our experimental framework, our four methods to define the vertex's partitions keep data utility on community detection tasks, but the underlying structure of the network is critical, and it has to be considered to find the best similarity measure.

Many interesting directions for future research have been uncovered by this work. Firstly, a deeper analysis on how the original graph's structure affects the generalization process must be conducted. Secondly, it would be thought-provoking to apply different similarity measures in each k-shell, according to the specific structure of the shell. Lastly, other information loss measures based on real graph mining processes can be considered such as information flow.

References

[1] L. A. Adamic, and N. Glance, "The political blogosphere and the 2004 U.S. election" in LinkKDD '05. ACM, 2005, pp. 36–43.

[2] L. Backstrom, C. Dwork, and J. Kleinberg, "Wherefore art thou r3579x?: anonymized social networks, hidden patterns, and structural steganography" in WWW '07. ACM, 2007, pp. 181–190.

[3] A.-L. Barabási, and R. Albert, "Emergence of Scaling in Random Networks". Science, vol. 286, no. 5439, pp. 509–512, 1999.

[4] V. Batagelj, and M. Zaveršnik, "Fast algorithms for determining (generalized) core groups in social networks". Advances in Data Analysis and Classification, vol. 5, no. 2, pp. 129–145, 2011.

[5] S. Bhagat, G. Cormode, B. Krishnamurthy, and D. Srivastava, "Class-based graph anonymization for social network data" in VLDB '09, vol. 2, no. 1, pp. 766–777, 2009.

[6] V. D. Blondel, J.-L. Guillaume, R. Lambiotte, and E. Lefebvre, "Fast unfolding of communities in large networks". Journal of Statistical Mechanics: Theory and Experiment, vol. 2008, no. 10, 2008.

[7] B. Bollobás, "Extremal Graph Theory". Academic Press, London. 1978.

[8] B.J. Cai, H.Y. Wang, H.R. Zheng, and H. Wang, "Evaluation repeated random walks in community detection of social networks" in ICMLC '10. IEEE, 2010, pp. 1849–1854.

[9] A. Campan, and T. M. Truta, "A Clustering Approach for Data and Structural Anonymity in Social Networks" in PinKDD '08. ACM, 2008, pp. 1–10.

[10] S. Carmi, S. Havlin, S. Kirkpatrick, Y. Shavitt, and E. Shir, E., "A model of Internet topology using k-shell decomposition". PNAS, vol. 104, no. 27, pp. 11150–11154, 2007.

[11] J. Casas-Roma, J. Herrera-Joancomartí, and V. Torra, "Anonymizing graphs: measuring quality for clustering". Knowledge and Information Systems, pp. 1–22, 2014.

[12] A. Clauset, M. E. J. Newman, and C. Moore, "Finding community structure in very large networks". Physical Review E, vol. 70, no. 6, pp. 1–6, 2004.

[13] G. Cormode, D. Srivastava, T. Yu, and Q. Zhang, "Anonymizing bipartite graph data using safe groupings". The VLDB Journal, vol. 19, no. 1, pp. 115–139, 2010.

[14] P. Erdös, and A. Rényi, "On Random Graphs I". Publicationes Mathematicae, vol. 6, pp. 290–297, 1959.

[15] R. Ford, T. M. Truta, and A. Campan, "p-Sensitive k-Anonymity for Social Networks" in DMIN '09. CSREA Press, 2009, pp. 403–409.

[16] R. Guimerà, L. Danon, A. Díaz-Guilera, F. Giralt, and A. Arenas, "Self-similar community structure in a network of human interactions". Physical Review E, vol. 68:065103, pp. 1–4, 2003.

[17] M. Hay, G. Miklau, D. Jensen, D. Towsley, and P. Weis, "Resisting structural re-identification in anonymized social networks" in VLDB '08, vol. 1, no. 1, pp. 102–114, 2008.

[18] X. He, J. Vaidya, B. Shafiq, N. Adam, and V. Atluri, "Preserving Privacy in Social Networks: A Structure-Aware Approach" in WI-IAT '09. IEEE, 2009, pp. 647–654.

[19] F. D. Malliaros, and M. Vazirgiannis, "To Stay or Not to Stay: Modeling Engagement Dynamics in Social Graphs", in CIKM '13. ACM, 2013, pp. 469–478.

[20] M. E. Nergiz, and C. Clifton, "Thoughts on k-anonymization". Data & Knowledge Engineering, vol. 63, no. 3, pp. 622–645, 2007.

[21] M. E. J. Newman, and M. Girvan, "Finding and evaluating community structure in networks". Physical Review E, vol. 69, no. 2, pp. 1–16, 2003.

[22] M. Rosvall, and C. T. Bergstrom, "Maps of random walks on complex networks reveal community structure". PNAS, vol. 105, no. 4, pp. 1118–1123, 2008.

[23] S. B. Seidman, "Network structure and minimum degree". Social Networks, vol. 5, no. 3, pp. 269–287, 1983.

[24] V. K. Sihag, "A clustering approach for structural k-anonymity in social networks using genetic algorithm" in CUBE '12. ACM, 2012, pp. 701–706.

[25] L. Singh, and C. Schramm, "Identifying Similar Neighborhood Structures in Private Social Networks" in ICDM '10. IEEE, 2010, pp. 507–516).

[26] K. Stokes, and V. Torra, "On some clustering approaches for graphs" in FUZZ-IEEE '11. IEEE, 2011, pp. 409–415.

[27] K. Stokes, and V. Torra, "Reidentification and k-anonymity: a model for disclosure risk in graphs". Soft Computing, vol. 16, no. 10, pp. 1657–1670, 2012.

[28] L. Sweeney, "k-anonymity: a model for protecting privacy". International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems, vol. 10, no. 5, pp. 557–570, 2002.

[29] W. W. Zachary, "An information flow model for conflict and fission in small groups". Journal of Anthropological Research, vol. 33, no. 4, pp. 452-473, 1977.

[30] E. Zheleva, and L. Getoor, "Preserving the Privacy of Sensitive Relationships in Graph Data" in PinKDD '07. Springer-Verlag, 2007, pp. 153–171.