Expert Systems With Applications 153 (2020) 113454

Contents lists available at ScienceDirect

Expert Systems With Applications

journal homepage: www.elsevier.com/locate/eswa

GRAM: An efficient (k, l) graph anonymization method

R. Mortazavi*, S.H. Erfani

Computer Engineering Department, School of Engineering, Damghan University, Damghan, Iran

ARTICLE INFO

Article history:
Received 4 April 2019
Revised 27 January 2020
Accepted 12 April 2020
Available online 17 April 2020

Keywords:
Data privacy
Graph anonymity
Data publishing
(k, l) Anonymization
Social networks

ABSTRACT

There are plenty of applications that use graphs for representing the association between different entities. Many research communities are interested in publishing such graphs to study the knowledge contained in them while the privacy of involved individuals is a challenging issue. A successful computational privacy model to address the problem of data privacy in microdata publishing is k-anonymity. A relaxed version of it for graphs is called (k, l)-anonymity: identifying at most l neighbors of a vertex v in the graph, an attacker cannot limit v in a group of less than k vertices. This paper introduces the GRAM: an efficient (k, l) GRaph Anonymization Method based on edge addition. At first, the GRAM adds enough edges to the graph to meet the privacy requirement. Then, the algorithm removes some redundant added edges to minimize changes in the original graph. The necessary and sufficient conditions for removing an added edge while maintaining the privacy requirement are proposed in the paper. This makes it possible to realize the model efficiently. An extensive set of experiments shows that the GRAM usually achieves a good trade-off between data utility and privacy, while it is more efficient than similar techniques.

1. Introduction

Nowadays, large amounts of personal data are being collected on numerous types of networks. Social networks with a significant number of addicted users (Leong, Hew, Ooi, Lee, & Hew, 2019) contain huge amounts of information about their users (Langari, Sardar, Mousavi, & Radfar, 2020). For example, Facebook claims over 1.4 billion monthly and 900 million daily active users (Abawajy, Ninggal, & Herawan, 2016). Additionally, there is an increasing number of requests by research communities to access the data on the association of users to groups in social networks for different purposes such as source detection of rumor in social networks (Shelke & Attar, 2019), identification of important members in online social networks (Rios, Aguilera, Nuñez-Gonzalez, & Graña, 2019), user modeling and usage profiling (Ying, Chiu, Venkatramanan, & Zhang, 2018), inferring social tie between users in the location-based social networks (LBSNs) (Njoo, Hsu, & Peng, 2018), and perception of social phenomena (Coletto et al., 2017). These issues make the users worry about their privacy due to the existence of sensitive information about them within the data. For instance, participants of social network services most often expressed their privacy concerns and the sharing of personal information online left them scared and feeling vulnerable (Jung, Walden, Johnson, & Sundar, 2017). Similarly, in health information dissemination, the shift by health care providers towards more dynamic methods of information sharing can be the cause of threat to the privacy of social media platforms (Sharma & Kaur, 2017).

The major challenge in sharing such graph data is to preserve the privacy of involved entities while the implicit knowledge embedded in them is maintained.

The privacy concerns of involved individuals in network data structures have motivated many research problems in recent years. Consequently, various definitions and graph modification algorithms are proposed to address the problem of privacy-preserving graph publishing. These methods use different techniques such as edge and vertex modification (addition and/or deletion), producing uncertain graphs, and clustering-based approaches (Casas-Roma, Herrera-Joancomartí, & Torra, 2017b). In the light of successful methods for traditional microdata anonymization such as Mortazavi and Jalili (2016), Machanavajjhala, Kifer, Gehrke, and Venkitasubramaniam (2007) and Soria-Comas, Domingo-Ferrer, Sánchez, and Martínez (2014), various techniques based on k-anonymity (Sweeney, 2001) are among the well-addressed graph privacy models (Feder, Nabar, & Terzi, 2008; Liu & Terzi, 2008) in which alternative definitions of k-anonymity for graphs have been introduced. In a k-anonymous dataset, each record is indistinguishable from at least $k-1$ other records in the dataset. However, it is shown that if k is large in relation to the order $|V|$ of the graph,

* Corresponding author.
E-mail addresses: r_mortazavi@du.ac.ir (R. Mortazavi), sh.erfani@du.ac.ir (S.H. Erfani).

https://doi.org/10.1016/j.eswa.2020.113454
0957-4174/

the usefulness of the k-anonymous graph is limited (Stokes, 2013; Stokes & Torra, 2012).

Additionally, in a social relationship graph, some private information leakages may occur indirectly based on the friendship relationship. In an important study, Bagrow, Liu, and Mitchell (2019) shown that approximately 95% of the potential predictive accuracy attainable for an individual is available within the social ties of an individual only, without requiring the individual's data. Therefore, a relaxed version of the k-anonymity model is defined which is called (k, l)-anonymity (Feder et al., 2008; Stokes & Torra, 2012). Simply, a graph is (k, l)-anonymous if no vertex can be limited within a group of less than k vertices based on identifying at most l of its neighbors in the graph.

Unfortunately, realizing the (k, l)-anonymity model for practical usages is difficult. Previous studies failed to address the privacy of individuals in theory or practice. Feder et al. (2008) defined the model for the first time. They developed a problem based on it and showed that it is NP-hard in general. However, the definition is shown later that suffers from some real-world problems and may lead to information leakages. Furthermore, their proposal is only a theoretic solution with no experimental evaluations. Therefore, the performance of the method is under question in practice. Stokes and Torra (2012) revisited the definition and suggested an idea based on probabilistic vertex removing and cloning to achieve (k, l)-anonymity, but no algorithmic details and evaluation results are reported. They also confirmed that the procedure may result in an empty graph and cannot always protect real individuals. In another study, Mortazavi and Erfani (2018) reported experimental outcomes based on 0–1 integer programming to address the model requirement, but their method is constrained to $l = 1$ and very small graphs which limits it for practical usages. Therefore, it is important to devise a utility-aware and efficient method to produce a (k, l)-anonymous version of a real-world graph.

1.1. Our contributions

The aim of this paper is to devise an effective method for graph anonymization that satisfies (k, l)-anonymity. There are numerous attacks in social network data publishing (Yang, Zhu, Zhou, & Xiang, 2016). This paper focuses on realizing an effective method against the attacks that try to disclose the identity of entities in a graph based on the identity of their neighbors which is addressed in the (k, l)-anonymity model. The proposed method utilizes edge addition to satisfy the privacy requirement while the produced graph is also useful in terms of common graph-theoretic measures. On the other hand, the method is efficient and can be applied to graphs with thousands of vertices and edges. The main contributions of the current study can be summarized as follows:

- A detailed method for realizing (k, l)-anonymity model based on edge addition is developed. The method considers the cases of $l = 1$ and $l > 1$ separately for efficiency issues.
- The proposed method first adds enough edges to the neighbors of each vertex such that they *all* share at least k common neighbors. This step constructs an augmented graph. Considering the privacy requirement, some of these added edges are removed in a utility-aware manner to decrease the amount of change in the graph.
- The conditions for removing an added edge from the augmented (k, l)-anonymous graph without violating the privacy requirement are defined and proved.
- A comprehensive set of experiments is conducted to evaluate the pertinence of the proposed method. The outcomes are compared with the results of similar existing techniques over some real-world graphs to validate the performance of the proposed method. Additionally, reproducible results of the method establish a baseline for comparisons in future studies.

1.2. Organization of this paper

The remaining of the paper is organized as follows: Section 2 establishes the theory used in the study. Section 3 reviews some related works. Section 4 provides the details of the GRAM. Section 5 evaluates the proposed method on some graphs. Finally, Section 6 concludes the paper.

2. Theory

This section defines the problem of (k, l)-anonymization of a graph. Section 2.1 introduces some notations used in the study. Section 2.2 describes the problem in details.

2.1. Notation

Let $G = (V, E)$ be a simple connected graph where $V = \{v_1, v_2, \ldots, v_n\}$ is the vertex set. Additionally, $E \subseteq V \times V$ denotes the set of m undirected edges in which $e_{i,j} = e_{j,i} = \{v_i, v_j\}$ shows a link between v_i and v_j both in V. The degree of the vertex $v \in V$ in G is referred to by $deg_G(v)$ and $D_G = \max_{v \in V} deg_G(v)$. Let $V' \subset V$, then $N_G(V', dist)$ represents all neighbors at distance $dist$ of at least a vertex in V', i.e., $N_G(V', dist) = \{u \in V : \exists v' \in V', d_G(u, v') = dist\}$ where $d_G(u, v)$ denotes the distance between vertices u and v in G. Assume $N_G(V') = N_G(V', 1)$. More specifically, $\forall v \in V$, the set of all immediate neighbors of v is shown by $N_G(\{v\})$ or simply $N_G(v)$. For a vertex set $V' \subset V$, let $Com_G(V')$ defines the set of common neighbors of all vertices of V' in G, i.e., $Com_G(V') = \{u \in V : \forall v' \in V', \{u, v'\} \in E\}$.

2.2. The (k, l)-anonymity model

There may be different definitions of a (k, l)-anonymous graph (see Section 3 for more details). This study considers the more recent and intuitive one that is introduced in Stokes and Torra (2012).

Definition 1 (Definition 21 of Stokes and Torra (2012))**.** $G = (V, E)$ is (k, l)-anonymous if for any vertex $v \in V$ and for any subset $S \subseteq N_G(v)$ of cardinality $|S| \leq l$, there are at least k distinct vertices $\{v_i\}_{i=1}^{k}$ such that $S \subseteq N_G(v_i)$ for $i \in [1, k]$.

Intuitively, $G = (V, E)$ is a (k, l)-anonymous graph if for any subset S of at most l neighbors of a vertex v, there exists a set V' of at least k vertices that share S as their common neighbors.

In this study, we consider an attacker model in which only real neighbors of an individual may be known by the attacker[1] Based on Definition 1, an attacker who re-identifies at most l real neighbors of an individual represented as v in the anonymized published graph G', could not limit v in a group of less than k vertices in it. In other words, the chance to re-identify a vertex is less than $1/k$ based on knowing at most l real neighbors of it in the anonymized graph. According to the definition and for the special case of $l = 1$ and $k \geq 2$, if an adversary identifies a neighbor v_j of a vertex v_i in a (k, l)-anonymous graph, she would not be able to re-identify v_i using v_j with probability more than $1/k$. The attacker may use different background knowledge such as the degree of the neighbor vertex v_j (Casas-Roma et al., 2017b) to identify it in G and then tries to *cascade* her knowledge to re-identify other vertices.

Example 1. Assume the toy graph G in Fig. 1a shows the relationship between some individuals. For $k = 2$ and $l = 1$, the degree of

[1] This is referred to by strong (k, l)-anonymization process in Feder et al. (2008).

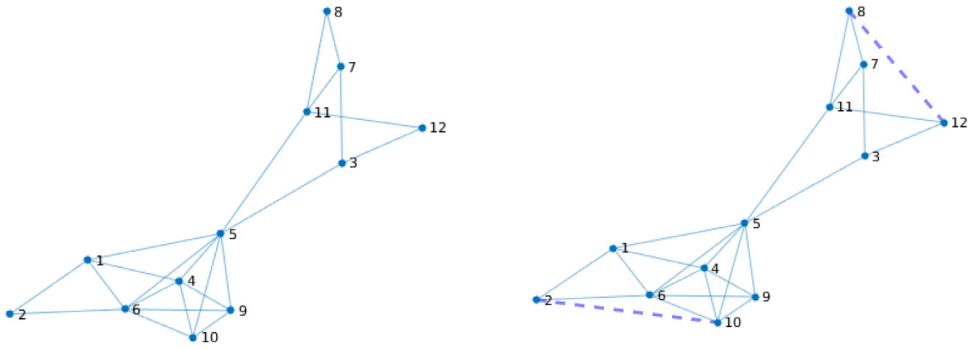

Fig. 1. A small graph and its protected version used in Example 1.

all vertices (i.e., neighbors) is greater than or equal to 2, so G is (2,1)-anonymous. However, the degree of some neighbors of v_{11} is less than 3. Therefore, G is not (3,1)-anonymous. However, we can add some *fake* links to G (shown by blue dashed lines in Fig. 1b) to make it (3,1)-anonymous. Similarly, for $k = 2$ and $l = 2$, an attacker who re-identifies v_7 and v_{11} in G, can distinguish v_8 as their only common friend with full confidence, i.e., G is not (2,2)-anonymous.

3. Related works

In this section, some (graph) anonymization methods are reviewed. These techniques usually include random perturbation approaches that modify edges and/or vertices of the graph to add some sort of noise to the graph, and model-driven techniques that modify edge and/or vertices of the graph (usually within an optimization process) to satisfy a privacy model requirement. A recent survey on graph anonymization methods is provided in Casas-Roma, Herrera-Joancomartí, and Torra (2017a).

Hay, Miklau, Jensen, Weis, and Srivastava (2007) designed an algorithm for randomly removing and then adding fake edges. Their method is efficient and simple, but hubs of the original graph are not well-protected and can be re-identified (Casas-Roma et al., 2017a).

Further, in the light of k-anonymity, some anonymization methods are devised for graphs. For instance, Liu and Terzi (2008) introduced k-degree anonymity in which all vertices meet k-anonymity for their degree. The authors devised a mathematical programming approach to construct a k-degree anonymous graph.

In another study, Langari et al. (2020) combined the fuzzy clustering and firefly optimization algorithm to achieve k-anonymity or its extensions in social networks. However, the method did not address any graph specific attacks. Additionally, the behavior of the proposed technique for large datasets is questionable since it utilizes evolutionary optimization algorithms.

Feder et al. (2008) defined (k, l)-anonymity as a relaxed version of k-anonymity. Based on their definition, a graph $G = (V, E)$ is called (k, l)-anonymous if for each vertex $v \in V$, there exists a set of vertices $U \subseteq V$ not containing v such that $|U| \geq k$ and for each $u \in U$, the two vertices u and v share at least l neighbors. In order to preserve the utility of the produced graph in terms of the number of added edges, the authors considered matching only between problematic (called deficient) vertices in the original graph for adding edges until the privacy constraint is satisfied. However, it is shown later that the definition suffers from some unforeseen disclosure risks and redefined the model.

Stokes and Torra (2012) revisited the initial definition of (k, l)-anonymity. The authors suggested randomly removing or cloning the problematic vertices until the graph becomes (k, l)-anonymous. However, as stated by the authors, these fake vertices make the algorithm to lose its status as an anonymization method. Additionally, an adversary who knows the anonymization algorithm can re-identify the victim with probability greater than $1/k$, since the clones of a vertex have completely the same neighbors of it and can be identified easily.

Mortazavi and Erfani (2018) implemented a limited version of (k, l)-anonymity for $l = 1$ using Integer Linear Programming (ILP). However, their approach is not suitable for a real-world application since it is based on solving a complex 0–1 optimization problem which has an exponential time and memory complexity.

3.1. Our contributions against existing techniques

Although many graph anonymization methods have been introduced in the literature, most of them are different from the proposed technique in this study. In brief, none of the past methods have realized and/or evaluated a general (k, l)-anonymization method.

The approaches that inject some sort of noise and change vertices or edges randomly are scalable and easy to implement, but they usually cannot offer a good trade-off between information loss and disclosure risk. Many of the previous studies try to apply traditional microdata models such as k-anonymity or t-closeness directly to graph structures. However, this usually results in a considerable information loss in comparison with the procedures devised specifically for graph anonymization (Stokes & Torra, 2012). Among the studies that realize a graph privacy model, some of them utilize evolutionary optimization approaches (Langari et al., 2020). However, it is not clear how the method scales when the dataset becomes large. Additionally, some graph specific anonymity models such as k-degree assume an optimistic attack model in which an attacker uses the degree of a vertex to re-identify it in a protected graph. Although very popular or alone individuals in a social network can be re-identified in this approach, the model does not consider how an attacker may use such information to re-identify other vertices. However, this is the main focus of the privacy model addressed in the (k, l)-anonymity in which the neighboring information is mainly used for re-identification.

More specific to the privacy model of this study, i.e., (k, l)-anonymity, the study by Feder et al. (2008) is almost a theoretic study with some ideas only for special cases while no experimental results are reported. The approach proposed by Stokes and Torra (2012) based on removing problematic vertices may lead to an empty protected graph. Additionally, their trick to mitigate the problem by randomly cloning a problematic vertex introduces some fake entities. An adversary can easily spot these fake vertices in the protected graph since they have completely the same

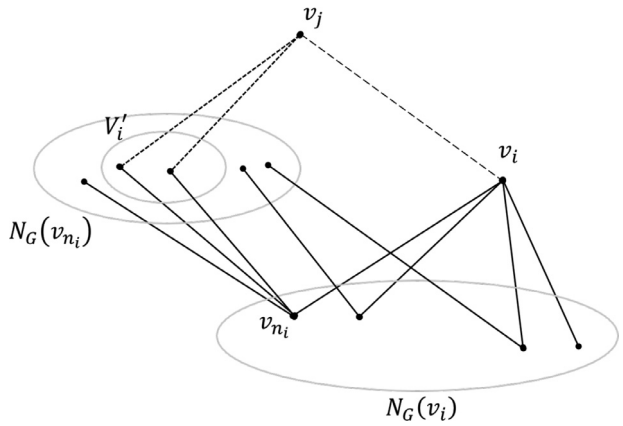

Fig. 2. The process of removing an edge $\{v_i, v_j\}$ (dashed line). Solid lines are edges in G and dotted ones are edges in G_2 in Proposition 2.

neighbors of the original vertex. Again, no implementation details or experimental results are reported by the authors. The study by Mortazavi and Erfani (2018) is almost the first work that reports experimental results, but only for the limited case of $l = 1$.

In contrast, the steps of the proposed method in the current study to realize (k, l)-anonymity are described in detail. The reproducible results confirm that it can achieve a good trade-off between information loss and disclosure risk. It is also general and efficient and can be applied to graphs with thousands of vertices.

4. The proposed method

In this paper, we propose the GRAM: an efficient (k, l)-anonymization method of a simple connected graph $G = (V, E)$. Section 4.1 provides some facts that are used within the GRAM. Section 4.2 describes the steps of the GRAM and Section 4.3 analyzes its complexity.

4.1. Preliminaries

This section provides some facts about a (k, l)-anonymous graph.

Proposition 1. *Let $G = (V, E)$ be a simple connected graph. If G is $(k, 1)$-anonymous then $\forall v \in V$, $deg_G(v) \geq k$.*

Proof. The graph is connected, so each vertex v has at least a neighbor. The degree of all of the neighbors must be $\geq k$ by Definition 1 which is satisfied in the assumption. □

Proposition 2. *Let $G_2 = (V, E_2)$ be a (k, l)-anonymized version of a simple connected graph $G = (V, E)$ where $E \subseteq E_2$. Additionally, assume an added edge $e_{i,j} \in E_2 \backslash E$ is removed from G_2 that yields the graph $G' = (V, E')$, $E' = E_2 \backslash e_{i,j}$. The following statements are true in G':*

1. *G' is $(k-1, l)$-anonymous.*
2. *Let $v_{n_i} \in N_G(v_i)$ and define $V'_{n_i} = N_G(v_{n_i}) \cap N_{G_2}(v_j)$. Similarly, V'_{n_j} can be defined. The graph G' is (k, l)-anonymous if and only if for all v_{n_i} (and v_{n_j}) any subset of at most l members of $V'_{n_i}(V'_{n_j})$ including $v_{n_i}(v_{n_j})$ has at least k common neighbors in G'.*

Proof. Fig. 2 illustrates the proof.

1. By removing $e_{i,j}$, any neighbor $v_{n_i} \in N_G(v_i) \cup N_G(v_j)$ in G' may lose its privacy since one of its neighbors, i.e., $v_i(v_j)$ has lost its neighbor $v_j(v_i)$. Let S denotes a subset of at most l vertices from $N_G(v_{n_i})$. If S contains neither v_i nor v_j, $Com_{G'}(S) = Com_{G_2}(S)$ since none of the neighbors of vertices in S are changed in G'. However, if S contains v_i or v_j, $|Com_{G'}(S)| \geq |Com_{G_2}(S)| - 1$[2], therefore, G' becomes $(k-1, l)$-anonymous.
2. In the forward direction, all vertices in $V'_{n_i}(V'_{n_j})$ are real neighbors of $v_{n_i}(v_{n_j})$. It is assumed that G' is (k, l)-anonymous. Therefore, any subset S of at most l neighbors of the vertex v_{n_i} in G, i.e., $S \subseteq N_G(v_{n_i})$, $|S| \leq l$ has at least k common neighbors in G', which is true by definition. In the reverse direction, as stated in the first part of the proposition, only a subset S of at most l members of a vertex in the neighborhood of v_i (or v_j) may threat $v_{n_i}(v_{n_j})$ such that (i) $v_i \in S(v_j \in S)$ and (ii) they are in $N_G(v_{n_i}) \cap N_{G_2}(v_j)$ $(N_G(v_{n_j}) \cap N_{G_2}(v_i))$ since only these vertices have lost one of their common neighbors in G', i.e., the $v_j(v_i)$. It is assumed that all of such subsets have at least k common neighbors so G' remains (k, l)-anonymous. □

Algorithm 1: The pseudo-code of the GRAM.

Input: $G = (V, E)$: original graph, k, l: privacy parameters
Output: $G' = (V, E')$: protected graph
1: $n = |V|$
2: $[G_2, costs] \leftarrow addEdges(G, k, l, n)$ ▷ Algorithm 2
3: $G' \leftarrow removeEdges(G, G_2, costs, k, l)$ ▷ Algorithm 3
4: **return** G'

4.2. The algorithm

Algorithm 1 shows the main body of the proposed procedure. The function accepts a simple connected[3] graph $G = (V, E)$, and the privacy parameters k, l as inputs. It constructs a protected graph G' which satisfies (k, l)-anonymity requirement. The number of vertices is stored in n in Line 1. In Line 2, enough edges are added to the graph to ensure that the augmented graph G_2 satisfies (k, l)-anonymity requirement. This is done using the *addEdges* function which is shown in Algorithm 2 . However, not all of the added edges are necessary for the privacy requirement. In Line 3, some added edges are removed from G_2 provided that the privacy condition is maintained using the *removeEdges* function (Algorithm 3). The result is saved in G' which is returned in Line 4.

The pseudo-code of the *addEdges* function is shown in Algorithm 2. It accepts a simple graph $G = (V, E)$, the privacy parameters k, l, and the graph order n. It constructs a (k, l)-anonymous augmented graph $G_2 = (V, E_2)$ where some edges are added to G, i.e., $E \subseteq E_2$. When a new edge is added to a graph to protect some of its vertices, the underlying graph structure changes or equivalently, the addition has a *cost*. It is desired to decrease the amount of change to preserve the original graph structure as much as possible, so the minimum number of added edges is favorable. Additionally, some added edges may contribute to protecting more vertices and some of them may be redundant, i.e., they can be removed without violating the privacy requirement. Therefore, it is required to save the contribution of each added edge to prioritize the removal of more costly redundant ones (Algorithm 3). The cost of each added edge as its negative contribution in the *addEdges* function is stored in the *costs* matrix[4] The more vertices protected by $e_{i,j}$, the less the value of the $costs(i, j)$ will be at the end of the algorithm. More precisely, for an added edge $e_{i,j}$ in Algorithm 2, the entry $costs(i, j)$ equals n minus the number of times the edge is used to protect the neighbors of v_i and v_j in Algorithm 2. First,

[2] Note that if v_{n_i} is in both $N_G(v_i)$ and $N_G(v_j)$, the number of common neighbors of S does not decrease since the graph is simple and has no self-loop.

[3] If G is not connected, the algorithm may be applied to each component, separately.

[4] The *costs* entries of real edges play no role in the algorithm and will be discarded in Algorithm 3.

Algorithm 2: The pseudo-code of *addEdges* function.

Input: $G = (V, E)$: graph,k, l: privacy parameters, n: graph order
Output: $G_2 = (V, E_2)$: augmented graph

1: $G_2 \leftarrow G$
2: $costs(i, j) \leftarrow n, \forall i, j \in \{1, 2, \cdots, n\}$
3: $flag \leftarrow 1$
4: $dist \leftarrow 1$
5: **if** $l == 1$ **then** ▷ the simple case
6: **while** $flag == 1$ **and** $dist < n - 1$ **do**
7: $flag \leftarrow 0$
8: **for** $i = 1 : n$ **do**
9: **if** $deg_{G_2}(v_i) < k$ **then**
10: Connect v_i to $N_G(v_i, dist + 1)$ in G_2
11: Decrease the *costs* of the involved edges
12: $flag \leftarrow 1$
13: $dist + +$
14: **else**
15: **while** $flag == 1$ **and** $dist < n$ **do**
16: $flag \leftarrow 0$
17: **for** $i = 1 : n$ **do**
18: **if** $|Com_{G_2}(N_G(v_i))| < k$ **then**
19: Connect all vertices $v_j \in N_G(v_i)$ to some vertices in $V' = N_G(v_i, dist)$ in G_2 with minimum number of new edges
20: Decrease the *costs* of the involved edges
21: $flag \leftarrow 1$
22: $dist + +$
23: **return** G_2

Algorithm 3: The pseudo-code of the *removeEdges* function.

Input: $G = (V, E)$: original graph, $G_2 = (V, E_2)$: augmented graph, *costs*: costs matrix, k, l: privacy parameters
Output: $G' = (V, E')$: protected graph

1: $G' = (V, E') \leftarrow G_2$
2: **foreach** $e_{i,j} \in E_2 \backslash E$ *in decreasing order of their associated costs* **do**
3: **if** $l == 1$ **then**
4: **if** $deg_{G'}(v_i) > k$ **and** $deg_{G'}(v_j) > k$ **then**
5: $E' \leftarrow E' \backslash e_{i,j}$ ▷ Proposition 1
6: **else**
7: $E' \leftarrow E_2 \backslash e_{i,j}$
8: **if** $isSafe(G, G_2, G', v_i, v_j, k, l)$ **then** ▷ Algorithm 4
9: $E_2 \leftarrow E'$ ▷ Proposition 2
10: **else**
11: $E' \leftarrow E' \cup e_{i,j}$ ▷ $e_{i,j}$ cannot be removed
12: **return** G'

G_2 is initialized by G in Line 1. Three variables *costs*, *flag*, and *dist* are set in Lines 2–4. Based on the value of l, two different cases are considered for adding new edges[5]

For the simple case of $l = 1$ (Line 5) can be handled efficiently since the degree of all vertices must be $\geq k$ (Proposition 1) or else they are connected to their close real neighbors in G_2. The privacy requirement for each vertex is checked within a loop (Lines 6–13). The loop continues while *flag* is set and $dist < n - 1$ since the diameter of the graph is always less than n. Within the loop body, in the first step *flag* is reset (Line 7), and then some computations are accomplished for each vertex v_i, $\forall i \in \{1, 2, \cdots, n\}$ in Lines 8–12. If v_i does not have enough neighbors in G_2 (Line 9), it is linked to its real neighbors at distance $dist + 1$ (Line 10), the *costs* entries of involved edges are decreased by one (Line 11), and *flag* is set (Line 12). Line 13 increments *dist* by one to consider further neighbors if required.

For the case of $l > 1$, the algorithm connects *all* real neighbors of each vertex to some other vertices in the graph such that the requirement of the (k, l)-anonymity is satisfied. The privacy requirement for each vertex v_i is checked in a nested loop (Lines 17–21). If *all* neighbors of v_i in G i.e., $N_G(v_i)$, do not share at least k common neighbors in G_2 (Line 18), all vertices in $N_G(v_i)$ are linked to a subset of V' the union of real neighbors of v_i at distance *dist* (Line 19)[6] Note that these links are in G_2 and some of them may be existed due to previous iterations of the algorithm. If no such link exists, it is added to E_2. It is desired to minimize the number of new edges to preserve the utility of the original graph as much as possible. Therefore, after connecting a vertex in V' to all vertices in $N_G(v_i)$, if the condition (Line 18) is satisfied, remaining vertices in V' are discarded. All the existed or newly added links in the step are called *involved edges* in Algorithm 2. The *costs* values of involved edges are decreased by one in Line 20 since their contribution to graph anonymization increases. Correspondingly, *flag* is set in Line 21. The last step within the loop increments *dist* by one (Line 22). Finally, Line 23 returns the augmented graph G_2.

Algorithm 3 shows the pseudo-code of the *removeEdges* function (Line 3 of Algorithm 1). The function gets the original and augmented graphs $G = (V, E)$ and $G_2 = (V, E_2)$, respectively along with the privacy parameters, k and l. A protected graph $G' = (V, E')$ is constructed by removing unnecessary *fake* links from G_2 as the output of the function. The algorithm removes some edges that are added by Algorithm 2, in a greedy manner. The edges are processed based on the decreasing order of their associated *costs* to prioritize the removal of less contributed added edges. A fake edge is removed if the privacy requirement is not violated. After initializing G' by G_2 in Line 1, for each added edge $e_{i,j}$, the algorithm checks whether the edge can be removed or not. In the case of $l = 1$, the edge can be removed provided that the degree of both edge endpoints is greater than k in G' (Proposition 1) which is checked in Line 4. The case of $l > 1$ becomes more complex. First, $e_{i,j}$ is removed from E' temporarily and the removal *safety* is verified by *isSafe* function in Line 8 (Proposition 2). The edge is also removed from E_2 if *isSafe* returns TRUE or else it is added to E' in Line 11. Finally, G' is returned in Line 12.

Given the original graph $G = (V, E)$, the augmented graph $G_2 = (V, E_2)$, the modified graph $G' = (V, E')$, a candidate edge for removal $\{v_i, v_j\}$, and k, $l > 1$, the function *isSafe* of Algorithm 4 checks whether it was safe to remove the edge from E' or not. It is designed based on Proposition 2. If the removal does not break the privacy requirement, the algorithm returns TRUE or else it returns FALSE. For each neighbor v_{n_i} of vertex v_i in the original graph G, the safety of the removal is checked in Lines 1–4. For any v_{n_i}, all of its neighbors in G which are also connected to v_j in G_2 are stored in V'_i (Fig. 2). It is checked in Line 3 whether there exists a subset of at most $\min(l, |V'_i|)$ vertices in V'_i that includes v_i and has less than k common neighbors or not. If there is such a subset, it is not *safe* to remove the edge and the function returns FALSE in Line 4. A similar computation is accomplished in Lines 5–8 for all neighbors v_{n_j} of v_j. The function returns TRUE in Line 9 if

[5] This algorithm uses l only for performance issues.

[6] For the case of $dist = 1$, no self-loop is added to the simple graph G_2.

Algorithm 4: The pseudo-code of the *isSafe* function.

Input: $G = (V, E)$: original graph, $G_2 = (V, E_2)$: augmented graph, $G' = (V, E')$: modified graph, $\{v_i, v_j\}$: candidate edge to be removed, k, l: privacy parameters
Output: $safe$: safety flag

foreach $v_{n_i} \in N_G(v_i)$ **do**
  $V_i' \leftarrow N_G(v_{n_i}) \cap N_{G_2}(v_j)$
  if there is a subset with at most $\min(l, |V_i'|)$ of V_i' including v_i with less than k common neighbors **then**
    return FALSE
foreach $v_{n_j} \in N_G(v_j)$ **do**
  $V_j' \leftarrow N_G(v_{n_j}) \cap N_{G_2}(v_i)$
  if there is a subset with at most $\min(l, |V_j'|)$ of V_j' including v_j with less than k common neighbors **then**
    return FALSE
return TRUE

none of the previous conditions are met which means it is safe to remove the edge $\{v_i, v_j\}$ from E_2.

4.3. The algorithm complexity

This study implements the GRAM using an adjacency matrix data structure in which a boolean $n \times n$ matrix represents the graph. Therefore, the space complexity of the GRAM is $O(n^2)$. The main body of the GRAM in Algorithm 1 consists of two main procedures: (1) Adding edges in Algorithm 2, in which both of the two cases have $O(n^2)$ iterations. Using pre-computed pairwise distances between all vertices[7], for the case of $l = 1$ each iteration requires at most $O(n)$ operations, while each iteration in the case of $l > 1$ is accomplished in $O(n.D_G^2)$. Therefore, *addEdges* is in $O(n^3.D_G^2)$. However, it is notable that the function constructs a full graph by adding $O(n^2 - m)$ edges in the worst case. (2) Algorithm 3 deletes some added edges. It involves sorting *costs* that takes $O((n^2 - m)\log(n^2 - m))$ computations and calling *isSafe* function at most $(n^2 - m)$ times. The *isSafe* function is accomplished in $O(n.l.D_G^{l+1})$ since $v_i(v_j)$ has $O(D_G)$ neighbors v_{n_i} (v_{n_j}) and V_i' (V_j') is defined for each one. The function requires considering all subsets of at most l members of the V_i' and V_j' each one with at most D_G members which takes $O(D_G^l)$ operations. For each subset S with $O(l)$ vertices, it is required to compute the $Com_{G'}(S)$ that requires $O(n.l)$ computations. Therefore, the whole process of the GRAM requires $O((n^3.D_G^2) + (n^2 - m)\log(n^2 - m) + (n^2 - m).n.l.D_G^{l+1})$ operations. When $m < < n^2$ and for small values of D_G and l, the runtime reduces to $O(n^3)$ in the worst case. Despite the theoretical upper bound of the runtime complexity, as the experiments in Section 5 confirm, the GRAM is efficient and seems to be useful in practice.

5. Experimental evaluations

In this section, different experiments are conducted to evaluate the efficacy of the proposed method[8] In all experiments, a PC with Core 2 Duo 3.0 GHz CPU, 16 GB of main memory and Windows 10 operating system is used. The algorithm is implemented in MATLAB 2017b. Section 5.1 introduces datasets and measures that are used to evaluate anonymization procedures in the study. Section 5.2 reports information loss of the anonymized graphs based on different utility measures. Section 5.3 shows the runtime of the GRAM. Section 5.4 compares the proposed method with two implementations of (k, l)-anonymity reported in Mortazavi and Erfani (2018) and Stokes and Torra (2012). Finally, Section 5.5 discusses theoretical and practical implications of the proposed approach.

5.1. Datasets and measures

The proposed method is applied to some graph datasets available online to see its performance in different situations. Some of these datasets are also addressed in previous studies (Clarkson, Liu, & Terzi, 2010; Rousseau, Casas-Roma, & Vazirgiannis, 2018). The structural properties of the graphs are given in Table 1. For each graph, the order n, size m, and density of the graph along with the maximum, minimum, average, and mode of vertex degrees, the Average Path Length *APL*, the Average Clustering Coefficient *ACC*, and the Average Betweenness Centrality *ABC* are reported. The *APL* is a concept in the network topology that is defined as the average number of steps along the shortest paths for all possible pairs of network vertices. It is a measure of the efficiency of information or mass transport on a network[9]. The *ACC* is the average of local clustering coefficients of graph vertices. The measure for a vertex quantifies how close its neighbors are to being a clique and determines whether a graph is a small-world network (Watts & Strogatz, 1998). Moreover, *ABC* is the average of betweenness centrality of all vertices. The measure for each vertex captures the number of shortest paths in the graph that passes through the vertex.[10]

In this study, the amount of change in the above measures after and before applying the anonymization procedure is considered as information loss. For instance, $\Delta APL = APL_2 - APL_1$ quantifies the amount of change in *APL* of the original graph. Obviously, $APL_2 \leq APL_1$ when some edges are added to the graph. The same indicators are also computed for both *ACC* and *ABC* values.

5.2. Information loss of anonymized graphs

This section reports $\Delta m = m_2 - m_1$, ΔAPL, ΔACC and ΔABC values of the GRAM in Tables 2, 3, 4, and 5, respectively. In these tables, information loss measures increase slightly in all cases when k or l increases. This is rational since more edges have to be added to the graph to meet the privacy requirement. For example, for jazz in Table 3, $|\Delta APL|$ increases from 0.0294 to 0.0849 when the privacy parameter changes from $k = 3$ to $k = 10$ in $l = 1$. Similarly, for URVemail and $k = 5$, $|\Delta ACC|$ in Table 4 is increased from 0.1564 to 0.3060 when l is changed from 1 to 3. These tables confirm that as the graphs become more sparse, the error measures increase significantly. This is logical since more changes have to be made to satisfy the privacy requirement.

Additionally, the experiments show that the utility of the anonymized graphs is more sensitive to l than k. For instance, $|\Delta APL|$ of URVemail is increased from 0.0759 to 0.3216 for $l = 1$ when k is changed from 3 to 10, while for $k = 3$, the value comes up to 1.2177 for $l = 3$. The same trend can be seen for other datasets and measures. This means that to defend against an attacker with more information about the neighborhood of vertices, much more changes in the structure of the graph are required that comes at the expense of more information loss.

[7] Pairwise distances can be computed simply, for example by repeated Breath First Traversal for each vertex, which totally requires $O(n(n + m))$ operations.

[8] Illustrative examples of the GRAM execution are provided in the supplementary material.

[9] https://en.wikipedia.org/wiki/Average_path_length.

[10] https://en.wikipedia.org/wiki/Betweenness_centrality.

Table 1
Structural properties of some real-world graphs.

graph	n	m	*density*	degree				*APL*	*ACC*	*ABC*
				max	*min*	*avg*	*mode*			
jazz	198	2742	0.1406	100	1	27.70	23	2.24	0.62	121.65
karate	34	78	0.1390	17	1	4.59	2	2.41	0.57	23.24
URVemail	1133	5451	0.0085	71	1	9.62	1	3.61	0.22	1475.01
USpowergrid	4941	6594	0.0005	19	1	2.67	2	18.99	0.08	44433.29

Table 2
The number of added edges Δm by the proposed method.

k	l	jazz	karate	URVemail	USpowergrid
3	1	13	10	333	2956
	2	917	74	11,236	13,177
	3	1480	107	17,445	14,625
4	1	23	21	582	5082
	2	1249	99	14,432	17,334
	3	1993	136	21,103	18,704
5	1	35	35	871	7359
	2	1512	135	17,160	21,343
	3	2285	164	24,697	22,682
10	1	117	108	2627	19,869
	2	2910	237	27,899	39,320
	3	3960	258	37,174	42,180

Table 3
$|\Delta APL|$ error indicator of the proposed method.

k	l	jazz	karate	URVemail	USpowergrid
3	1	0.0294	0.0214	0.0759	0.7183
	2	0.3081	0.6399	1.0602	9.1675
	3	0.3708	0.7130	1.2177	9.6444
4	1	0.0374	0.0891	0.1096	1.7026
	2	0.3463	0.7041	1.1822	10.0882
	3	0.4082	0.7718	1.3114	10.3416
5	1	0.0354	0.1800	0.1408	2.9163
	2	0.3842	0.7522	1.2647	10.4880
	3	0.4450	0.8253	1.3777	10.8656
10	1	0.0849	0.5276	0.3216	7.0290
	2	0.4862	0.9697	1.4676	12.0127
	3	0.5538	1.0071	1.5387	12.1981

Table 4
$|\Delta ACC|$ error indicator of the proposed method.

k	l	jazz	karate	URVemail	USpowergrid
3	1	0.0350	0.1309	0.1575	0.4270
	2	0.0926	0.2411	0.2699	0.5186
	3	0.1146	0.2538	0.2894	0.4394
4	1	0.0401	0.1861	0.1557	0.4276
	2	0.0995	0.2609	0.2752	0.5439
	3	0.1258	0.2662	0.2963	0.4842
5	1	0.0409	0.2396	0.1564	0.4373
	2	0.1063	0.2572	0.2821	0.5600
	3	0.1316	0.2647	0.3060	0.5192
10	1	0.0594	0.3005	0.1589	0.4754
	2	0.1263	0.2941	0.3082	0.5743
	3	0.1473	0.3097	0.3491	0.5753

Table 5
$|\Delta ABC|$ error indicator of the proposed method.

k	l	jazz	karate	URVemail	USpowergrid
3	1	13.16	3.29	155.93	6674.77
	2	67.90	19.51	853.51	25603.02
	3	76.67	20.01	868.41	26267.91
4	1	10.64	6.67	227.23	15397.26
	2	73.88	20.32	923.70	27492.58
	3	86.56	19.06	933.10	27648.01
5	1	17.28	8.60	287.99	21805.44
	2	79.90	19.76	943.04	28092.86
	3	86.32	17.98	940.70	29285.86
10	1	27.61	23.66	647.51	36864.66
	2	83.43	18.74	1004.61	30955.63
	3	78.89	19.21	1007.46	31258.26

Table 6
The runtime of the GRAM in seconds for different datasets and privacy parameters k and l.

k	l	jazz	karate	URVemail	USpowergrid
3	1	1	1	1	5
	2	4	1	13	21
	3	27	1	40	22
4	1	1	1	1	6
	2	2	1	13	24
	3	6	1	47	25
5	1	1	1	1	7
	2	2	1	14	27
	3	7	1	53	29
10	1	1	1	1	9
	2	2	1	16	38
	3	7	1	82	41

5.3. The runtime of the proposed method

This section reports the execution time of the proposed method for different graphs. Table 6 shows the whole execution time (excluding input and output operations) for $k = \{3, 4, 5, 10\}$ and $l = \{1, 2, 3\}$. The results support the theoretical complexities, i.e., the runtime increases for larger graphs and greater values of k and l. For example, the runtime of the GRAM is always about 1 second for karate while it takes about 5 to 41 seconds to finish in the case of the USpowergrid graph. Additionally, the procedure runtime is more sensitive to l than k since it is exponential in l. However, the results confirm that the proposed method is efficient and practical for graphs with thousands of vertices and edges.

5.4. Comparison with previous studies

In this section, a comparison between the GRAM and the procedure introduced in Feder et al. (2008) is conducted. Feder et al. suggested minimizing the number of added edges to minimize information loss. They proved the problem is NP-hard in general. Mortazavi and Erfani implemented their idea based on an Integer Linear Programming approach (thus called ILP) for $l = 1$ (model 1 of Mortazavi & Erfani (2018)). Table 7 shows the results of the ILP based on Δm, $|\Delta APL|$, $|\Delta ACC|$, $|\Delta ABC|$, and the runtime in seconds. Comparing these outcomes with the values reported in Tables 2–5 confirm that fewer edges are added in the ILP at the expense of much more required time. For instance, in the case of $k = 3$, and $l = 1$, USpowergrid is anonymized by the ILP in which 2054 edges are added in 516 seconds, while the GRAM adds 2956 edges in just 5 seconds. In almost all cases, the difference between the measures is decreased when k is increased. Regarding the properties of the original graphs (Table 1), the relative improvement of the ILP is negligible in most cases. For example, the dataset of jazz is anonymized for $k = 3$ with $|\Delta APL_{ILP}| = 0.0047$ and $|\Delta APL_{GRAM}| = 0.0294$ which means about $0.0247/2.24 * 100\% = 1\%$ of improve-

Table 7
The performance measures of the idea of Feder et al. (2008) implemented by ILP for $l = 1$ (Mortazavi & Erfani, 2018).

Error	k	jazz	karate	URVemail	USpowergrid
Δm	3	7	7	209	2054
	4	12	16	389	4025
	5	19	28	602	6197
	10	83	100	2116	18,144
$\lvert\Delta APL\rvert$	3	0.0047	0.0539	0.0470	10.4078
	4	0.0166	0.1912	0.0922	12.1901
	5	0.0250	0.2982	0.1331	13.1569
	10	0.0691	0.6386	0.4592	16.9679
$\lvert\Delta ACC\rvert$	3	0.0118	0.1528	0.0218	0.0252
	4	0.0194	0.2189	0.0394	0.0427
	5	0.0293	0.3043	0.0532	0.0532
	10	0.0564	0.3429	0.1089	0.3584
$\lvert\Delta ABC\rvert$	3	8.6026	6.9185	150.9991	42739.0534
	4	12.2778	12.9127	258.1235	43817.3446
	5	12.5005	15.7988	369.9953	44204.7361
	10	28.6292	22.9675	1042.6328	46825.0865
runtime (sec)	3	3	1	13	516
	4	2	2	14	1177
	5	2	2	17	3632
	10	2	2	23	4224

ment. The same argument holds for other measures, but *ACC* improvements are considerable in the ILP, especially for URVemail and USpowergrid graphs. In brief, the GRAM achieves a better trade-off between information loss and performance for the same level of privacy. Additionally, the GRAM is not limited to $l = 1$ which is an important limitation of the ILP approaches in practice.

Another similar study in the literature is reported by Stokes and Torra (2012) in which problematic vertices are randomly removed (along with their connected edges) or cloned (with completely the same neighbors of the original vertex) to generate a (k, l)-anonymous graph. However, they did not provide implementation details and the results in their study. In the following, an implementation of their idea in a probabilistic approach (so it is called the PROB) based on the following assumptions is proposed: (1) the probability to delete or clone a problematic vertex is the same, (2) a vertex can be removed if its removal does not violate the privacy requirement for remaining vertices, (3) if a problematic vertex cannot be removed, it is cloned, (4) all vertices without edges are removed from the constructed graph, and (5) all vertices are randomly visited to meet the privacy requirement. Table 8 shows the results of the PROB for jazz and URVemail in terms of the change in the number of vertices $\Delta n = n_2 - n_1$, and the previous measures, i.e., Δm, ΔAPL, ΔACC, ΔABC, and the runtime in seconds.

The results in Table 8 confirm that in most cases the delete was unsuccessful and most of the problematic vertices are cloned. However, as stated by the authors, this decreases the privacy of individuals in the anonymized graph, since these cloned vertices do not match a real-world entity. Additionally, the PROB approach adds much more new edges than the GRAM and ILP which means a considerable information loss. The ΔACC of the PROB is usually better than the GRAM. This is probably due to the way the PROB changes the graph, i.e., it is more vertex oriented than edge oriented and it does not change the neighborhood of a vertex selectively. However, GRAM is usually the winner in terms of other measures. More precisely, the GRAM produces better results in terms of ΔAPL, ΔABC, and runtime in 24, 19, and 19 out of 24 experiments, respectively.

5.5. Methodological contribution

The paper has some considerable research contributions to privacy-aware graph publishing systems. The most important contribution is the use of a two-phase approach of adding and then removing some added edges for graph anonymization. By using this technique, it is shown empirically that a graph can be anonymized efficiently such that the (k, l)-anonymity requirement is satisfied. Even though methods for graph anonymization based on edge addition (Feder et al., 2008; Mortazavi & Erfani, 2018) and deletion (Bonchi, Gionis, & Tassa, 2014) have been utilized in graph anonymization, the application of them in a two-staged approach is very limited. Secondly, the idea of considering the union of vertex neighbors at different distances to satisfy (k, l)-anonymity has provided a significant methodological contribution to the literature of graph anonymization. Using this idea as a starting point, future studies on graph anonymization procedures may use the current two-stage approach as their research methodology guideline. Practically, the findings of this research will provide new insight to practitioners to address graph anonymity, especially in social network analysis. This will allow social media managers to value the privacy of their clients while providing useful information for researchers in a timely manner. The GRAM enables high ranking officers to devise flexible policies and strategies for data publishing.

6. Conclusions and future work

This paper develops the GRAM, a procedure for (k, l) graph anonymization. The adversary is assumed to use the neighborhood of a victim vertex to re-identify it. An anonymization algorithm based on edge addition is suggested. The utility measures related to graph theory are used to evaluate the GRAM. A set of comprehensive comparisons with similar studies is conducted. Given a specified privacy level, the results show that the GRAM is promising in terms of both information loss and performance. Evaluating the proposed method based on other utility measures such as

Table 8
The results of the probabilistic approach (PROB) introduced in Stokes and Torra (2012).

k	l	Δn		Δm		ΔAPL		ΔACC		ΔABC		runtime (sec)	
		jazz	URVemail	jazz	URVemail	jazz	URVemail	jazz	URVemail	jazz	URVemail	jazz	URVemail
3	1	1	205	158	4740	0.1120	−0.8841	−0.0103	0.0208	6.3182	−114.8036	1	3
	2	161	1222	9008	33,315	−4.4490	−10.3021	0.0408	0.0205	−67.0226	−1192.6556	2	39
	3	228	1300	13,423	34,833	−7.4435	−11.3947	0.0475	0.0096	−110.4920	−1309.8785	188	301
4	1	7	438	234	11,338	−0.0698	−2.6144	−0.0001	0.0386	−1.9903	−356.6960	1	7
	2	266	1895	17,128	62,535	−9.1488	−19.2463	0.0521	0.0180	−126.8733	−1908.2498	4	61
	3	344	2023	24,788	65,595	−13.1799	−21.4942	0.0442	0.0103	−165.5855	−2094.4817	340	689
5	1	14	735	478	21,721	−0.2304	−5.1947	0.0135	0.0433	−6.0168	−664.0667	1	6
	2	357	2580	26,641	100,159	−14.1956	−30.6330	0.0547	0.0183	−180.2727	−2639.2773	6	85
	3	469	2751	39,699	105,150	−21.0268	−34.2789	0.0469	0.0108	−228.5680	−2885.2119	573	1399
10	1	94	2129	4517	114,166	−2.4935	−21.5012	0.0709	0.0439	−51.8485	−1887.0678	1	16
	2	846	6091	108,902	419,490	−54.0546	−125.6105	0.0556	0.0184	−411.4157	−6425.9441	34	429
	3	1118	6331	168,095	429,555	−87.7606	−135.2821	0.0461	0.0117	−557.6057	−6772.4960	6616	17,142

classification accuracy (Chamikara, Bertok, Liu, Camtepe, & Khalil, 2018), is left for future work. Moreover, it is interesting to use microaggregation methods (Mortazavi & Jalili, 2015; 2016) or a better heuristic to add and remove edges to improve the utility of the anonymization method. The problem of privacy leakage seems to be more critical for people addicted to social media since they probably share more information with others to fulfill their needs for socialization. However, the problem is out of the scope of this research and has to be investigated with more details in a study similar to Leong et al. (2019).

Declaration of Competing Interest

None.

Credit authorship contribution statement

R. Mortazavi: Conceptualization, Methodology, Formal analysis, Writing - original draft, Writing - review & editing. **S.H. Erfani:** Conceptualization, Data curation, Writing - review & editing.

Acknowledgment

This research is partially supported by ITRC (Iran Telecommunication Research Center) under contract No. 12200/500. The authors would like to thank Klara Stokes for her insightful comments that help improve the paper. Additionally, the authors would like to acknowledge the excellent comments and suggestions made by anonymous reviewers of the manuscript.

Supplementary material

Supplementary material associated with this article can be found, in the online version, at doi:10.1016/j.eswa.2020.113454.

References

Abawajy, J. H., Ninggal, M. I. H., & Herawan, T. (2016). Privacy preserving social network data publication. *IEEE Communications Surveys & Tutorials, 18*(3), 1974–1997.

Bagrow, J. P., Liu, X., & Mitchell, L. (2019). Information flow reveals prediction limits in online social activity. *Nature Human Behaviour, 3*(2), 122–128.

Bonchi, F., Gionis, A., & Tassa, T. (2014). Identity obfuscation in graphs through the information theoretic lens. *Information Sciences, 275*, 232–256.

Casas-Roma, J., Herrera-Joancomartí, J., & Torra, V. (2017a). A survey of graph-modification techniques for privacy-preserving on networks. *Artificial Intelligence Review, 47*(3), 341–366.

Casas-Roma, J., Herrera-Joancomartí, J., & Torra, V. (2017b). k-Degree anonymity and edge selection: Improving data utility in large networks. *Knowledge and Information Systems, 50*(2), 447–474.

Chamikara, M. A. P., Bertok, P., Liu, D., Camtepe, S., & Khalil, I. (2018). Efficient data perturbation for privacy preserving and accurate data stream mining. *Pervasive and Mobile Computing, 48*, 1–19.

Clarkson, K. L., Liu, K., & Terzi, E. (2010). Toward identity anonymization in social networks. *Link Mining: Models, Algorithms, and Applications*, 359–385.

Coletto, M., Esuli, A., Lucchese, C., Muntean, C. I., Nardini, F. M., Perego, R., & Renso, C. (2017). Perception of social phenomena through the multidimensional analysis of online social networks. *Online Social Networks and Media, 1*, 14–32.

Feder, T., Nabar, S. U., & Terzi, E. (2008). Anonymizing graphs. arXiv:0810.5578.

Hay, M., Miklau, G., Jensen, D., Weis, P., & Srivastava, S. (2007). *Anonymizing social networks*.

Jung, E. H., Walden, J., Johnson, A. C., & Sundar, S. S. (2017). Social networking in the aging context: Why older adults use or avoid Facebook. *Telematics and Informatics, 34*(7), 1071–1080.

Langari, R. K., Sardar, S., Mousavi, S. A. A., & Radfar, R. (2020). Combined fuzzy clustering and firefly algorithm for privacy preserving in social networks. *Expert Systems with Applications, 141*, 112968. doi:10.1016/j.eswa.2019.112968.

Leong, L.-Y., Hew, T.-S., Ooi, K.-B., Lee, V.-H., & Hew, J.-J. (2019). A hybrid sem-neural network analysis of social media addiction. *Expert Systems with Applications, 133*, 296–316.

Liu, K., & Terzi, E. (2008). Towards identity anonymization on graphs. In *Proceedings of the 2008 ACM SIGMOD international conference on management of data* (pp. 93–106). ACM.

Machanavajjhala, A., Kifer, D., Gehrke, J., & Venkitasubramaniam, M. (2007). L-diversity: Privacy beyond *k*-anonymity. *ACM Transaction on Knowledge Discovery from Data (TKDD), 10*(1), 1217299–1217302.

Mortazavi, R., & Erfani, S. H. (2018). An effective method for utility preserving social network graph anonymization based on mathematical modeling. *International Journal of Engineering, 31*(10), 1624–1632.

Mortazavi, R., & Jalili, S. (2015). Preference-based anonymization of numerical datasets by multi-objective microaggregation. *Information Fusion, 25*(0), 85–104.

Mortazavi, R., & Jalili, S. (2016). Enhancing aggregation phase of microaggregation methods for interval disclosure risk minimization. *Data Mining and Knowledge Discovery, 30*(3), 605–639.

Njoo, G. S., Hsu, K.-W., & Peng, W.-C. (2018). Distinguishing friends from strangers in location-based social networks using co-location. *Pervasive and Mobile Computing, 50*, 114–123.

Rios, S. A., Aguilera, F., Nuñez-Gonzalez, J. D., & Graña, M. (2019). Semantically enhanced network analysis for influencer identification in online social networks. *Neurocomputing, 326-327*, 71–81.

Rousseau, F., Casas-Roma, J., & Vazirgiannis, M. (2018). Community-preserving anonymization of graphs. *Knowledge and Information Systems, 54*(2), 315–343.

Sharma, P., & Kaur, P. D. (2017). Effectiveness of web-based social sensing in health information dissemination-A review. *Telematics and Informatics, 34*(1), 194–219.

Shelke, S., & Attar, V. (2019). Source detection of rumor in social network - A review. *Online Social Networks and Media, 9*, 30–42.

Soria-Comas, J., Domingo-Ferrer, J., Sánchez, D., & Martínez, S. (2014). Enhancing data utility in differential privacy via microaggregation-based *k*-anonymity. *The VLDB Journal - The International Journal on Very Large Data Bases, 23*(5), 771–794.

Stokes, K. (2013). Graph k-anonymity through k-means and as modular decomposition. In *Nordic conference on secure IT systems* (pp. 263–278). Springer.

Stokes, K., & Torra, V. (2012). Reidentification and k-anonymity: A model for disclosure risk in graphs. *Soft Computing, 16*(10), 1657–1670.

Sweeney, L. (2001). *Computational disclosure control: A primer on data privacy protection*. Massachusetts Institute of Technology Ph.D. thesis..

Watts, D. J., & Strogatz, S. H. (1998). Collective dynamics of 'small-world' networks. *Nature, 393*(6684), 440.

Yang, M., Zhu, T., Zhou, W., & Xiang, Y. (2016). Attacks and countermeasures in social network data publishing. *ZTE Communications, 2*, 1.

Ying, Q. F., Chiu, D. M., Venkatramanan, S., & Zhang, X. (2018). User modeling and usage profiling based on temporal posting behavior in OSNs. *Online Social Networks and Media, 8*, 32–41.