Towards Identity Anonymization on Graphs

Kun Liu
IBM Almaden Research Center
San Jose, CA 95120, USA
kun@us.ibm.com

Evimaria Terzi
IBM Almaden Research Center
San Jose, CA 95120, USA
eterzi@us.ibm.com

ABSTRACT

The proliferation of network data in various application domains has raised privacy concerns for the individuals involved. Recent studies show that simply removing the identities of the nodes before publishing the graph/social network data does not guarantee privacy. The structure of the graph itself, and in its basic form the degree of the nodes, can be revealing the identities of individuals. To address this issue, we study a specific graph-anonymization problem. We call a graph k-degree anonymous if for every node v, there exist at least $k-1$ other nodes in the graph with the same degree as v. This definition of anonymity prevents the re-identification of individuals by adversaries with *a priori* knowledge of the degree of certain nodes. We formally define the graph-anonymization problem that, given a graph G, asks for the k-degree anonymous graph that stems from G with the minimum number of graph-modification operations. We devise simple and efficient algorithms for solving this problem. Our algorithms are based on principles related to the realizability of degree sequences. We apply our methods to a large spectrum of synthetic and real datasets and demonstrate their efficiency and practical utility.

Categories and Subject Descriptors: G.2.2 [**DISCRETE MATHEMATICS**]: Graph Theory; H.2.8 [**DATABASE MANAGEMENT**]: Database Applications—*Data mining*

General Terms: Algorithms, Experimentation, Theory

Keywords: Anonymity, Degree Sequence, Dynamic Programming

1. INTRODUCTION

Social networks, online communities, peer-to-peer file sharing and telecommunication systems can be modelled as complex graphs. These graphs are of significant importance in various application domains such as marketing, psychology, epidemiology and homeland security. The management and analysis of these graphs is a recurring theme with increasing interest in the database, data mining and theory communities. Past and ongoing research in this direction has revealed interesting properties of the data and presented efficient ways of maintaining, querying and updating them. However, with the exception of some recent work [2, 10, 19, 14, 18], the privacy concerns associated with graph-data analysis and management have been largely ignored.

In their recent work, Backstrom *et. al.* [2] point out that the simple technique of anonymizing graphs by removing the identities of the nodes before publishing the actual graph does not always guarantee privacy. It is shown in [2] that there exist adversaries that can infer the identity of the nodes by solving a set of restricted graph isomorphism problems. However, the problem of designing techniques that could protect individuals' privacy has not been addressed in [2].

Hay *et. al.* [10] further observe that the structural similarity of nodes' neighborhood in the graph determines the extent to which an individual in the network can be distinguished. This structural information is closely related to the degrees of the nodes and their neighbors. Along this direction, the authors propose an anonymity model for social networks – a graph satisfies *k-candidate anonymity* if for every structure query over the graph, there exist at least k nodes that match the query. The structure queries check the existence of neighbors of a node or the structure of the subgraph in the vicinity of a node. However, the authors of [10] mostly focus on providing a set of anonymity definitions and studying their properties, and not on designing algorithms that guarantee the construction of a graph that satisfies their anonymity requirements.

Motivated by Backstrom *et al.* and Hay *et al.*'s work, we want to answer the following question that is still unanswered: *how to minimally modify the graph to protect the identity of each individual involved?* Being able to answer this kind of questions is important to understand the privacy issues on graph/social networks, the computational complexity of the related problems, and other challenges and opportunities in this emerging field (see *e.g.*, Jon Kleinberg, *Challenges in Social Network Data: Processes, Privacy and Paradoxes*, ACM SIGKDD 2007 Invited Talk).

We note that in a social network, nodes correspond to individuals or other social entities, and edges correspond to social relationships between them. The privacy breaches in social network data can be grouped to three categories: 1) *identity disclosure*: the identity of the individual who is associated with the node is revealed; 2) *link disclosure*: sensitive relationships between two individuals are disclosed; and 3)

SIGMOD'08, June 9–12, 2008, Vancouver, BC, Canada.
Copyright 2008 ACM 978-1-60558-102-6/08/06 ...$5.00.

content disclosure: the privacy of the data associated with each node is breached, *e.g.*, the email message sent and/or received by the individuals in a email communication graph. We believe that a perfect privacy-protection system should consider all of these issues. However, protecting against each of the above breaches may require different techniques. For example, for content disclosure, standard privacy-preserving data mining techniques [1], such as data perturbation and k-anonymization can help. For link disclosure, the various techniques studied by the link-mining community [8, 19] can be useful.

In this paper, we focus on identity disclosure. We propose a systematic framework for identity anonymization on graphs. We use some of the definitions proposed in [10], and we solve the algorithmic problems that arise from these definitions.

1.1 The Problem

In order to prevent the identity disclosure of individuals, we propose a new graph-anonymization framework. More specifically, we address the following problem: given a graph G and an integer k, modify G via a set of edge-addition (or deletion) operations in order to construct a new k-degree anonymous graph $\widehat{G}$, in which every node v has the same degree with at least $k-1$ other nodes. Of course, one could transform G to the complete graph, in which all nodes would be identical. Although such an anonymization would preserve privacy of individual nodes, it would make the anonymized graph useless for any study. For that reason we impose the additional requirement that the minimum number of such edge-modifications is made. In this way, we want to preserve the utility of the original graph, while at the same time satisfy the degree-anonymity constraint.

In this paper, we assume that the graph is simple, *i.e.*, the graph is undirected, unweighted, containing no self-loops or multiple edges. We focus our presentation on the problem of edge additions. The case of edge deletions is symmetric and thus can be handled analogously; it is sufficient to consider the complement of the input graph. In the end of this paper, we discuss how to extend the proposed framework to allow simultaneous edge addition and deletion operations when modifying the input graph.

1.2 Related Work

Since the introduction of the concept of anonymity in databases [15], there has been increasing interest in the database community in studying the complexity of the problem and proposing algorithms for anonymizing data records under different anonymization models [4, 12, 13]. Though lots of attention has been given to the anonymization of tabular data, the privacy issues of graphs/social networks and the notion of anonymization of graphs have only been recently touched.

Backstrom *et. al.* [2] show that simply removing the identifiers of the nodes does not always guarantee privacy. Adversaries can infer the identity of the nodes by solving a set of restricted isomorphism problems based on the uniqueness of small random subgraphs embedded in an network.

Hay *et. al.* [10] observe that the structural similarity of the nodes in the graph determines the extent to which an individual in the network can be distinguished from others. Based on the notion of k-anonymity [15], they propose a k-candidate anonymity model.

Pei and Zhou in [14] consider yet another definition of graph anonymity – a graph is k-anonymous if for every node there exist at least $k-1$ other nodes that share isomorphic neighborhoods; in this case the neighborhood of a node is defined by its immediate neighbors and the connections between them. This definition of anonymity in graphs is different from ours. In a sense it is a more strict one. Given the difference in the definition, the corresponding algorithmic problems arising in [14] are also different from the problems we consider in this paper.

Zheleva and Getoor [19] consider the problem of protecting sensitive relationships among the individuals in the anonymized social network. This is closely related to the link-prediction problem that has been widely studied in the link-mining community [8]. In [19] simple edge-deletion and node-merging algorithms are proposed to reduce the risk of sensitive link disclosure. This work is different from ours in that we are primarily interested in identity protection; once the identity of each individual is disguised, there is no need to hide the sensitive relationships any more. Also the set of combinatorial problems that we need to solve in our framework are very different from the set of problems discussed in [19].

Sensitive link and relationship protection is also discussed by Ying and Wu [18]. They study how anonymization algorithms that are based on randomly adding and removing edges change certain graph properties. More specifically, they focus on the change caused in the eigenvalues (spectrum) of the network. The authors additionally explore, how the randomized network can be exploited by the adversary to gain knowledge about the existence of certain links. The focus of our paper as well as the set of techniques we develop are orthogonal to those discussed in [18].

Frikken and Golle [7] study the problem of assembling pieces of graphs owned by different parties privately. They propose a set of cryptographic protocols that allow a group of authorities to jointly reconstruct a graph without revealing the identity of the nodes. The graph thus constructed is isomorphic to a perturbed version of the original graph. The perturbation consists of addition and or deletions of nodes and or edges. Unlike their work, we try to anonymize a single graph while keeping the data utility in mind. Moreover, the methods in [7] involve cryptographic protocols that are computationally demanding, while our approach employs simple and efficient combinatorial algorithms.

1.3 Roadmap

The rest of the paper is organized as follows. In Section 2 we define the graph-anonymization problem and in Section 3 we provide a high-level overview of our approach. Sections 4 – 7 consider the anonymization problem where only edge additions (or only deletions) are allowed. In Section 8 we show an extension of our framework that allows simultaneous edge additions and deletions, and we conclude the paper in Section 9.

2. PROBLEM DEFINITION

Let $G(V,E)$ be a simple graph; V is a set of nodes and E the set of edges in G. We use $\mathbf{d}_G$ to denote the *degree sequence* of G. That is, $\mathbf{d}_G$ is a vector of size $n=|V|$ such that $\mathbf{d}_G(i)$ is the degree of the i-th node of G. Throughout the paper, we use $\mathbf{d}(i)$, $\mathbf{d}(v_i)$ and $\mathbf{d}_G(i)$ interchangeably to denote the degree of node $v_i \in V$. When the graph is clear from the context we drop the subscript in notation and use $\mathbf{d}(i)$ instead. Without loss of generality, we also assume that entries in $\mathbf{d}$ are ordered in decreasing order of the degrees they correspond to, that is, $\mathbf{d}(1) \geq \mathbf{d}(2) \geq \ldots \geq \mathbf{d}(n)$. Additionally, for $i < j$ we use $\mathbf{d}[i, j]$ to denote the subsequence of $\mathbf{d}$ that contains elements $i, i+1, \ldots, j-1, j$.

Before defining the notion of a *k-degree anonymous graph*, we first define the notion of a k-anonymous vector of integers.

Definition 1. *A vector of integers $\mathbf{v}$ is k-anonymous, if every distinct value in $\mathbf{v}$ appears at least k times.*

For example, vector $\mathbf{v} = [5, 5, 3, 3, 2, 2, 2]$ is 2-anonymous.

Definition 2. *A graph $G(V, E)$ is k-degree anonymous if the degree sequence of G, $\mathbf{d}_G$, is k-anonymous.*

Alternatively, Definition 2 states that for every node $v \in V$ there exist at least $k-1$ other nodes that have the same degree as v. This property prevents the re-identification of individuals by adversaries with *a priori* knowledge of the degree of certain nodes. This echoes the observation made by Hay *et. al.* [10].

Figure 1 shows two examples of degree-anonymous graphs. In the graph on the left, all three nodes have the same degree and thus the graph is 3-degree anonymous. Similarly, the graph on the right is 2-degree anonymous since there are two nodes with degree 1 and four nodes with degree 2.

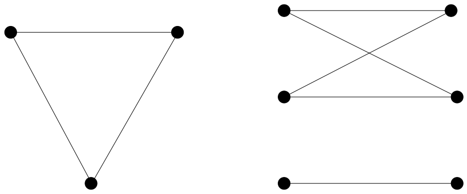


Figure 1: Examples of a 3-degree anonymous graph (left) and a 2-degree anonymous graph (right).

Degree anonymity has the following monotonicity property.

Proposition 1. *If a graph $G(V, E)$ is k_1-degree anonymous, then it is also k_2-degree anonymous, for every $k_2 \leq k_1$.*

We use the definitions above to define the Graph Anonymization problem. The input to the problem is a simple graph $G(V, E)$ and an integer k. The requirement is to use a set of graph-modification operations on G in order to construct a k-degree anonymous graph $\widehat{G}(\widehat{V}, \widehat{E})$ that is structurally similar to G. We require that the output graph is over the same set of nodes as the original graph, that is, $\widehat{V} = V$. Moreover, we restrict the graph-modification operations to edge additions, that is, graph $\widehat{G}$ is constructed from G by adding a (minimal) set of edges. We call the cost of anonymizing G by constructing $\widehat{G}$ the graph-anonymization cost Ga and we compute it by $\textsc{Ga}(\widehat{G}, G) = |\widehat{E}| - |E|$.

Formally, we define the Graph Anonymization as follows.

Problem 1 (Graph Anonymization). *Given a graph $G(V, E)$ and an integer k, find a k-degree anonymous graph $\widehat{G}(V, \widehat{E})$ with $\widehat{E} \cap E = E$ such that $\textsc{Ga}(\widehat{G}, G)$ is minimized.*

Note that the Graph Anonymization problem always has a *feasible* solution. In the worst case, all edges not present in the input graph can be added. In this way, the graph becomes complete and all nodes have the same degree; thus, any degree-anonymity requirement is satisfied (due to Proposition 1).

However, in the formulation of Problem 1 we want to find the k-degree anonymous graph that incurs the minimum graph-anonymization cost. That is, we want to add the *minimum* number of edges to the original graph to obtain a k-degree anonymous version of it. The least number of edges constraint tries to capture the requirement of structural similarity between the input and output graphs. If

$$L_1\left(\widehat{\mathbf{d}} - \mathbf{d}\right) = \sum_i \left|\widehat{\mathbf{d}}(i) - \mathbf{d}(i)\right|,$$

then minimizing the number of additional edges can be translated into minimizing the L_1 distance of the degree sequences of G and $\widehat{G}$. This is due to the fact that

$$\textsc{Ga}(\widehat{G}, G) = \left|\widehat{E}\right| - |E| \quad = \quad \frac{1}{2} L_1\left(\widehat{\mathbf{d}} - \mathbf{d}\right). \tag{1}$$

It is obvious that Problem 1 can be modified so that it allows only for edge deletions, instead of additions. It can be easily shown that solving the latter variant is equivalent to solving Problem 1 on the complement of the input graph. Recall that the complement of a graph $G(V, E)$ has the same nodes as G and has as edges all edges that are not in G. Therefore, all our results carry over to the edge-deletion case as well. The generalized problem where we allow for simultaneous additions and deletions of edges so that the output graph is k-degree anonymous is another natural variant. We discuss this problem in Section 8 and we show that our algorithmic framework need only be minimally modified to solve the latter problem. For now, let us focus on Problem 1.

In general, requiring that $\widehat{G}(V, \widehat{E})$ is a supergraph of the input graph $G(V, E)$ is a rather strict constraint. We will show that we can naturally relax this requirement to the one where $\widehat{E} \cap E \approx E$ rather than $\widehat{E} \cap E = E$. We call this problem the Relaxed Graph Anonymization problem and we develop a set of algorithms for this relaxed version. In our experimental evaluation we show that the degree-anonymous graphs we obtain in this case are still very similar to the original input graphs.

3. OVERVIEW OF THE APPROACH

We propose a two-step approach for the Graph Anonymization problem and its relaxed version. For an input graph $G(V, E)$ with degree sequence $\mathbf{d}$ and an integer k, we proceed as follows:

1. First, starting from $\mathbf{d}$, we construct a new degree sequence $\widehat{\mathbf{d}}$ that is k-anonymous and such that the *degree-anonymization* cost

   $$\textsc{Da}(\widehat{\mathbf{d}}, \mathbf{d}) = L_1(\widehat{\mathbf{d}} - \mathbf{d}),$$

   is minimized.

2. Given the new degree sequence $\widehat{\mathbf{d}}$, we then construct a graph $\widehat{G}(V, \widehat{E})$ such that $\mathbf{d}_{\widehat{G}} = \widehat{\mathbf{d}}$ and $\widehat{E} \cap E = E$ (or $\widehat{E} \cap E \approx E$ in the relaxed version).

Note that step 1 requires $L_1(\widehat{\mathbf{d}}-\mathbf{d})$ to be minimized, which in fact translates into the requirement of the minimum number of edge additions due to Equation (1). Step 2 tries to construct a graph with degree sequence $\widehat{\mathbf{d}}$, which is a supergraph (or has large overlap in its set of edges) with the original graph. If $\widehat{\mathbf{d}}$ is the optimal solution to the problem in Step 1 and Step 2 outputs a graph with degree sequence $\widehat{\mathbf{d}}$, then the output of this two-step process is the optimal solution to the Graph Anonymization problem.

Although in reality obtaining the optimal solution is not that easy, we show how to solve the Graph Anonymization and its relaxed version by performing Steps 1 and 2 as described above. These two steps give rise to two problems, which we formally define and solve in subsequent sections. Performing step 1 translates into solving the Degree Anonymization defined as follows.

Problem 2 (Degree Anonymization). *Given* $\mathbf{d}$*, the degree sequence of graph* $G(V,E)$*, and an integer* k *construct a* k*-anonymous sequence* $\widehat{\mathbf{d}}$ *such that* $L_1(\widehat{\mathbf{d}}-\mathbf{d})$ *is minimized.*

Similarly, performing step 2 translates into solving the Graph Construction problem that we define below.

Problem 3 (Graph Construction). *Given a graph* $G(V,E)$ *and a* k*-anonymous degree sequence* $\widehat{\mathbf{d}}$*, construct graph* $\widehat{G}(V,\widehat{E})$ *such that* $\widehat{\mathbf{d}} = \mathbf{d}_{\widehat{G}}$ *and* $\widehat{E}\cap E = E$ *(or* $\widehat{E}\cap E \approx E$ *in the relaxed version).*

In the next sections we develop algorithms for solving Problems 2 and 3. There are cases where we are unable to find the optimal k-degree anonymous graph $\widehat{G}^*$. In these cases, we are content to finding a k-degree anonymous graph $\widehat{G}$ that has cost $\text{GA}(\widehat{G},G) \geq \text{GA}(\widehat{G}^*,G)$ but as close to $\text{GA}(\widehat{G}^*,G)$ as possible.

4. DEGREE ANONYMIZATION

In this section we give algorithms for solving the Degree Anonymization problem. Given the degree sequence $\mathbf{d}$ of the original input graph $G(V,E)$, the algorithms output a k-anonymous degree sequence $\widehat{\mathbf{d}}$ such that the degree-anonymization cost $\text{DA}(\mathbf{d}) = L_1(\widehat{\mathbf{d}}-\mathbf{d})$ is minimized.

We first give a dynamic-programming algorithm (DP) that solves the Degree Anonymization problem optimally in time $O(n^2)$. Then, we show how to modify it to achieve linear-time complexity. For completeness, we also give a fast greedy algorithm that runs in time $O(nk)$ and, as shown in the experimental section, gives high-quality results in practice.

In Problem 1 we have restricted our attention to edge-addition operations. Thus, the degrees of the nodes can only increase in the Degree Anonymization problem. That is, if $\mathbf{d}$ is the original sequence and $\widehat{\mathbf{d}}$ is the k-anonymous degree sequence, then for every $1 \leq i \leq n$ we have that $\widehat{\mathbf{d}}(i) \geq \mathbf{d}(i)$. This allows us to make the following observation.

Observation 1. *Consider a degree sequence* $\mathbf{d}$*, with* $\mathbf{d}(1) \geq \ldots \geq \mathbf{d}(n)$*, and let* $\widehat{\mathbf{d}}$ *be the optimal solution to the* Degree Anonymization *problem with input* $\mathbf{d}$*. If* $\widehat{\mathbf{d}}(i) = \widehat{\mathbf{d}}(j)$*, with* $i < j$*, then* $\widehat{\mathbf{d}}(i) = \widehat{\mathbf{d}}(i+1) = \ldots = \widehat{\mathbf{d}}(j-1) = \widehat{\mathbf{d}}(j)$*.*

Given a (sorted) input degree sequence $\mathbf{d}$, let $\text{DA}(\mathbf{d}[1,i])$ be the degree-anonymization cost of subsequence $\mathbf{d}[1,i]$. Additionally, let $I(\mathbf{d}[i,j])$ be the degree anonymization cost when all nodes $i, i+1, \ldots, j$ are put in the same anonymized group. Alternatively, this is the cost of assigning to all nodes $\{i,\ldots,j\}$ the same degree, which by construction will be the highest degree, in this case $\mathbf{d}(i)$, or

$$I(\mathbf{d}[i,j]) = \sum_{\ell=i}^{j} (\mathbf{d}(i) - \mathbf{d}(\ell)).$$

Using Observation 1 we can construct a set of dynamic-programming equations that solve the Graph Anonymization problem. That is,
for $i < 2k$,

$$\text{DA}(\mathbf{d}[1,i]) = I(\mathbf{d}[1,i]). \quad (2)$$

For $i \geq 2k$,

$$\text{DA}(\mathbf{d}[1,i]) = \min \quad (3)$$
$$\Big\{ \min_{k \leq t \leq i-k} \{\text{DA}(\mathbf{d}[1,t]) + I(\mathbf{d}[t+1,i])\}, I(\mathbf{d}[1,i]) \Big\}.$$

When $i < 2k$, it is impossible to construct two different anonymized groups each of size k. As a result, the optimal degree anonymization of nodes $1,\ldots,i$ consists of a single group in which all nodes are assigned the same degree equal to $\mathbf{d}(1)$.

Equation (3) handles the case where $i \geq 2k$. In this case, the degree-anonymization cost for the subsequence $\mathbf{d}[1,i]$ consists of optimal degree-anonymization cost of the subsequence $\mathbf{d}[1,t]$, plus the anonymization cost incurred by putting all nodes $t+1,...,i$ in the same group (provided that this group is of size k or larger). The range of variable t as defined in Equation (3) is restricted so that all groups examined, including the first and last ones, are of size at least k.

Running time of the DP algorithm: For an input degree sequence of size n, the running time of the DP algorithm that implements Recursions (2) and (3) is $O(n^2)$; First, the values of $I(\mathbf{d}[i,j])$ for all $i < j$ can be computed in an $O(n^2)$ preprocessing step. Then, for every i the algorithm goes through at most $n-2k+1$ different values of t for evaluating the Recursion (3). Since there are $O(n)$ different values of i, the total running time is $O(n^2)$.

We now show how to improve the running time of the DP algorithm from $O(n^2)$ to $O(nk)$. The core idea for this speedup lies in the simple observation that *no anonymous group should be of size larger than* $2k-1$*. If any group is larger than or equal to* $2k$*, it can be broken into two subgroups with equal or lower overall degree-anonymization cost.* The proof of this observation is rather simple and is thus omitted. Using this observation, the preprocessing step that computes the values of $I(\mathbf{d}[i,j])$, does not have to consider all the combinations of (i,j) pairs, but for every i consider only j's such that $k \leq j-i+1 \leq 2k-1$. Thus, the running time for this step drops to $O(nk)$.

Similarly, for every i, we do not have to consider all t's in the range $k \leq t \leq i-k$ as in Recursion (3), but only t's in the range $\max\{k, i-2k+1\} \leq t \leq i-k$. Therefore, Recursion (3) can be rewritten as follows

$$\text{DA}\left(\mathbf{d}[1,i]\right) = \min_{\max\{k,i-2k+1\}\leq t\leq i-k} \left\{\text{DA}\left(\mathbf{d}[1,t]\right) + I\left(\mathbf{d}[t+1,i]\right)\right\}. \tag{4}$$

For this range of values of t we guarantee that the first group has size at least k, and the last one has size between k and $2k-1$. Therefore, for every i the algorithm goes through at most k different values of t for evaluating the new recursion. Since there are $O(n)$ different values of i, the overall running time of the DP algorithm is $O(nk)$. Therefore, we have the following.

THEOREM 1. *Problem 2 can be solved in polynomial time using the* DP *algorithm described above.*

In fact, in the case where we consider edge additions or deletions only and we do not consider simultaneous edge additions and deletions, the running time of the DP algorithm can be further improved to $O(n)$. That is, the running time can become linear in n but independent of k. This is due to the fact that the value of $\text{DA}\left(\mathbf{d}[1,i']\right)$ given in Equation (4) is decreasing in t for i and i' sufficiently larger than i. This means that for every i not all integers t in the interval $[\max\{k,i-2k+1\},i-k]$ are candidate for boundary points between groups. In fact, we only need to keep a limited number of such points and their corresponding degree-anonymization costs calculated as in Equation (4). With a careful bookkeeping we can get rid of the factor k in the running time of the DP algorithm. We refer to the details of this speedup to the extended version of the paper.

For completeness, we also give a Greedy linear-time alternative algorithm for the DEGREE ANONYMIZATION problem. Although this algorithm is not guaranteed to find the optimal anonymization of the input sequence, our experiments show that it performs extremely well in practice, achieving anonymizations with costs very close to the optimal.

The Greedy algorithm first forms a group consisting of the first k highest-degree nodes and assigns to all of them degree $\mathbf{d}(1)$. Then it checks whether it should merge the $(k+1)$-th node into the previously formed group or start a new group at position $(k+1)$. For taking this decision the algorithm computes the following two costs:

$$C_{\text{merge}} = \big(\mathbf{d}(1) - \mathbf{d}(k+1)\big) + I\left(\mathbf{d}[k+2,2k+1]\right),$$

and

$$C_{\text{new}} = I\left(\mathbf{d}[k+1,2k]\right).$$

If C_{merge} is greater than C_{new}, a new group starts with the $(k+1)$-th node and the algorithm proceeds recursively for the sequence $\mathbf{d}[k+1,n]$. Otherwise, the $(k+1)$-th node is merged to the previous group and the $(k+2)$-th node is considered for merging or as a starting point of a new group. The algorithm terminates after considering all n nodes.

Running time of the Greedy algorithm: For degree sequences of size n, the running time of the Greedy algorithm is $O(nk)$; for every node i, Greedy looks ahead at $O(k)$ other nodes in order to make the decision to merge the node with the previous group or to start a new group. Since there are n nodes, the total running time is $O(nk)$.

5. GRAPH CONSTRUCTION

In this section we present algorithms for solving the GRAPH CONSTRUCTION problem. Given the original graph $G(V,E)$ and the desired k-anonymous degree sequence $\widehat{\mathbf{d}}$ output by the DP (or Greedy) algorithm, we construct a k-degree anonymous graph $\widehat{G}(V,\widehat{E})$ with $\widehat{E}\cap E = E$ and degree sequence $\mathbf{d}_{\widehat{G}}$ with $\mathbf{d}_{\widehat{G}} = \widehat{\mathbf{d}}$.

5.1 Basics on Realizability of Degree Sequences

Before giving the actual algorithms for the GRAPH CONSTRUCTION problem, we first present some known facts about the realizability of degree sequences for simple graphs. Later on, we extend some of these results in our own problem setting.

DEFINITION 3. *A degree sequence* $\mathbf{d}$*, with* $\mathbf{d}(1)\geq,\ldots,\geq \mathbf{d}(n)$ *is called* realizable *if and only if there exists a simple graph whose nodes have precisely this sequence of degrees.*

Erdös and Gallai [6] have stated the following *necessary* and *sufficient* condition for a degree sequence to be realizable.

LEMMA 1. *([6]) A degree sequence* $\mathbf{d}$ *with* $\mathbf{d}(1)\geq\ldots\geq \mathbf{d}(n)$ *and* $\sum_i \mathbf{d}(i)$ *even, is realizable if and only if for every* $1\leq \ell\leq n-1$ *it holds that*

$$\sum_{i=1}^{\ell}\mathbf{d}(i) \quad\leq\quad \ell(\ell-1) + \sum_{i=\ell+1}^{n}\min\{\ell,\mathbf{d}(i)\} \tag{5}$$

Informally, Lemma 1 states that for each subset of the ℓ highest-degree nodes, the degrees of these nodes can be "absorbed" within the nodes and the outside degrees. The proof of Lemma 1 is inductive ([9]) and it provides a natural construction algorithm, which we call ConstructGraph (see Algorithm 1 for the pseudocode).

The ConstructGraph algorithm takes as input the desired degree sequence $\mathbf{d}$ and outputs a graph with exactly this degree sequence, if such graph exists. Otherwise it outputs a "No" if such graph does not exist. The algorithm is iterative and in each step it maintains the residual degrees of vertices. In each iteration it picks an arbitrary node v and adds edges from v to $\mathbf{d}(v)$ nodes of *highest* residual degree, where $\mathbf{d}(v)$ is the residual degree of v. The residual degrees of these $\mathbf{d}(v)$ nodes are decreased by one. If the algorithm terminates and outputs a graph, then this graph has the desired degree sequence. If at some point the algorithm cannot make the required number of connections for a specific node, then it outputs "No" meaning that the input degree sequence is not realizable.

Note that the ConstructGraph algorithm is an *oracle* for the realizability of a given degree sequence; if the algorithm outputs "No", then this means that there does not exist a simple graph with the desired degree sequence.

Running time of the ConstructGraph algorithm: If n is the number of nodes in the graph and $d_{\max} = \max_i \mathbf{d}(i)$, then the running time of the ConstructGraph algorithm is $O(nd_{\max})$. This running time can be achieved by keeping an array A of size $d_{\max}$ such that $A[\mathbf{d}(i)]$ keeps a hash table of all the nodes of degree $\mathbf{d}(i)$. Updates to this array (degree changes and node deletions) can be done in constant time. For every node i at most $d_{\max}$ constant-time operations are required. Since there are n nodes the running time

Algorithm 1 The `ConstructGraph` algorithm.

```
    Input: A degree sequence $\mathbf{d}$ of length $n$.
    Output: A graph $G(V,E)$ with nodes having degree
    sequence $\mathbf{d}$ or "No" if the input sequence is not realizable.
 1: $V \leftarrow \{1,\ldots,n\}$, $E \leftarrow \emptyset$
 2: if $\sum_i \mathbf{d}(i)$ is odd then
 3:     Halt and return "No"
 4: while 1 do
 5:     if there exists $\mathbf{d}(i)$ such that $\mathbf{d}(i) < 0$ then
 6:         Halt and return "No"
 7:     if the sequence $\mathbf{d}$ are all zeros then
 8:         Halt and return $G(V,E)$
 9:     Pick a random node $v$ with $\mathbf{d}(v) > 0$
10:     Set $\mathbf{d}(v) = 0$
11:     $V_{\mathbf{d}(v)} \leftarrow$ the $\mathbf{d}(v)$-highest entries in $\mathbf{d}$ (other than $v$)
12:     for each node $w \in V_{\mathbf{d}(v)}$ do
13:         $E \leftarrow E \cup (v,w)$
14:         $\mathbf{d}(w) \leftarrow \mathbf{d}(w) - 1$
```

of the algorithm is $O(nd_{\max})$. In the worst case, $d_{\max}$ can be of order $O(n)$, and in this case the running time of the `ConstructGraph` algorithm is quadratic. In practice, $d_{\max}$ is much less than n, which makes the algorithm very efficient in practical settings.

Note that the random node in Step 9 of Algorithm 1 can be replaced by either the current highest-degree node or the current lowest-degree node. When we start with higher degree nodes, we get topologies that have very dense cores, while when start with lower degree nodes, we get topologies with very sparse cores. A random pick is a balance between the two extremes. The running time is not affected by this choice, due to the data structure A.

5.2 Realizability of Degree Sequences with Constraints

Notice that Lemma 1 is not directly applicable to the Graph Construction problem. This is because not only do we need to construct a graph $\widehat{G}$ with a given degree sequence $\widehat{\mathbf{d}}$, but we also require that $E \subseteq \widehat{E}$. We capture these two requirements in the following definition of *realizability of $\widehat{\mathbf{d}}$ subject to graph G.*

Definition 4. *Given input graph $G(V,E)$, we say that degree sequence $\widehat{\mathbf{d}}$ is realizable subject to G, if and only if there exists a simple graph $\widehat{G}(V,\widehat{E})$ whose nodes have precisely the degrees suggested by $\widehat{\mathbf{d}}$ and $E \subseteq \widehat{E}$.*

Given the above definition we have the following alternation of Lemma 1.

Lemma 2. *Consider degree sequence $\widehat{\mathbf{d}}$ and graph $G(V,E)$ with degree sequence $\mathbf{d}$. Let vector $\mathbf{a} = \widehat{\mathbf{d}} - \mathbf{d}$ such that $\sum_i \mathbf{a}(i)$ is even. If $\widehat{\mathbf{d}}$ is realizable subject to graph G then*

$$\sum_{i\in V_\ell} \mathbf{a}(i) \quad \le \quad \sum_{i\in V_\ell} \Big(\ell - 1 - \mathbf{d}^\ell(i)\Big) \qquad + \sum_{i\in V-V_\ell} \min\{\ell - \mathbf{d}^\ell(i), \mathbf{a}(i)\}, \qquad (6)$$

where $\mathbf{d}^\ell(i)$ is the degree of node i in the input graph G when counting only edges in G that connect node i to one of the nodes in V_ℓ. Here V_ℓ is an ordered set of ℓ nodes with the ℓ largest $\mathbf{a}(i)$ values, sorted in decreasing order. In other words, for every pair of nodes (u,v) where $u \in V_\ell$ and $v \in V \setminus V_\ell$, it holds that $\mathbf{a}(u) \ge \mathbf{a}(v)$ and $|V_\ell| = \ell$.

Although the proof of the lemma is omitted due to space constraints, one can see the similarity between Inequalities (5) and (6); if G is a graph with no edges between its nodes, then $\mathbf{a}$ is the same as $\widehat{\mathbf{d}}$, $\mathbf{d}^\ell(i)$ is zero, and the two inequalities become identical.

Lemma 2 states that Inequality (6) is just a *necessary* condition for realizability subject to the input graph G. Thus, if Inequality (6) does not hold, we can conclude that for input graph $G(V,E)$, there does not exist a graph $\widehat{G}(V,\widehat{E})$ with degree sequence $\widehat{\mathbf{d}}$ such that $E \subseteq \widehat{E}$.

Although Lemma 2 gives only a necessary condition for realizability subject to an input graph G, we still want to devise an algorithm for constructing a degree-anonymous graph $\widehat{G}$, a supergraph of G, if such a graph exists. We call this algorithm the `Supergraph`, which is an extension of the `ConstructGraph` algorithm (We omit the pseudocode of `Supergraph` due to space limits).

The inputs to the `Supergraph` are the original graph G and the desired k-anonymous degree distribution $\widehat{\mathbf{d}}$. The algorithm operates on the sequence of *additional degrees* $\mathbf{a} = \widehat{\mathbf{d}} - \mathbf{d}_G$ in a manner similar to the one the `ConstructGraph` algorithm operates on the degrees $\mathbf{d}$. However, since $\widehat{G}$ is drawn on top of the original graph G, we have the additional constraint that edges already in G cannot be drawn again.

The `Supergraph` first checks whether Inequality (6) is satisfied and returns "No" if it does not. Otherwise it proceeds iteratively and in each step it maintains the residual additional degrees $\mathbf{a}$ of the vertices. In each iteration it picks an arbitrary vertex v and adds edges from v to $\mathbf{a}(v)$ vertices of *highest* residual additional degree, ignoring nodes v' that are already connected to v in G. For every new edge (v,v'), $\mathbf{a}(v')$ is decreased by 1. If the algorithm terminates and outputs a graph, then this graph has degree sequence $\widehat{\mathbf{d}}$ and is a supergraph of the original graph. If the algorithm does not terminate, then it outputs "Unknown", meaning that there might exist a graph, but the algorithm is unable to find it. Though `Supergraph` is similar to `ConstructGraph`, it *is not* an oracle. That is, if the algorithm does not return a graph $\widehat{G}$ supergraph of G, it does not necessarily mean that such a graph does not exist.

For degree sequences of length n and $a_{\max} = \max_i \mathbf{a}(i)$ the running time of the `Supergraph` algorithm is $O(na_{\max})$, using the same data-structures as those described in Section 5.1.

5.3 The `Probing` scheme

If the `Supergraph` algorithm returns graph $\widehat{G}$, then not only do we guarantee that this graph is the k-degree anonymous but also that the least number of edge additions has been made.

If `Supergraph` returns "No" or "Unknown", we are content in tolerating some more edge-additions in order to get a degree-anonymous graph. For that we introduce the `Probing` scheme that forces the `Supergraph` algorithm to output the desired k-degree anonymous graph with a little extra cost. This scheme is in fact a randomized iterative process that tries to slightly change the degree sequence $\widehat{\mathbf{d}}$.

Algorithm 2 The `Probing` scheme.

```
Input: Input graph $G(V,E)$ with degree distribution $\mathbf{d}$ and integer $k$.
Output: Graph $\widehat{G}(V,\widehat{E})$ with $k$-anonymous degree sequence $\widehat{\mathbf{d}}$, such that $E \subseteq \widehat{E}$.
$\widehat{\mathbf{d}}$ = DP($\mathbf{d}$)    /* or Greedy($\mathbf{d}$) */
$\left(\text{realizable}, \widehat{G}\right)$ = Supergraph($\widehat{\mathbf{d}}$)
while realizable = "No" or "Unknown" do
    $\mathbf{d} = \mathbf{d}$ + random_noise
    $\widehat{\mathbf{d}}$ = DP($\mathbf{d}$)    /*or Greedy($\mathbf{d}$) */
    $\left(\text{realizable}, \widehat{G}\right)$ = Supergraph($\widehat{\mathbf{d}}$)
Return $\widehat{G}$
```

The pseudocode of the `Probing` scheme is shown in Algorithm 2. For input graph $G(V,E)$ and integer k, the `Probing` scheme first constructs the k-anonymous sequence $\widehat{\mathbf{d}}$ by invoking the `DP` (or `Greedy`) algorithm. If the subsequent call to the `Supergraph` algorithm returns a graph $\widehat{G}$, then `Probing` outputs this graph and halts. If `Supergraph` returns "No" or "Unknown" , then `Probing` slightly increases some of the entries in $\mathbf{d}$ via the addition of uniform noise - the specifics of the noise-addition strategy is further discussed in the next paragraph. The new noisy version of $\mathbf{d}$ is then fed as input to the `DP` (or `Greedy`) algorithm again. A new version of the $\widehat{\mathbf{d}}$ is thus constructed and input to the `Supergraph` algorithm to be checked. The process of noise addition and checking is repeated until a graph is output by `Supergraph`. Note that this process will always terminate because in the worst case, the noisy version of $\mathbf{d}$ will contain all entries equal to $n-1$, and there exists a complete graph that satisfies this sequence and is k-degree anonymous with $E \subseteq \widehat{E}$.

Since the `Probing` procedure will always terminate, the key question is how many times the **while** loop is executed. This depends, to a large extent, on the noise addition strategy. In our implementation, we examine the nodes in increasing order of their degrees, and slightly increase the degree of a single node in each iteration. This strategy is suggested by the degree sequences of the input graphs. In most of these graphs there is a small number of nodes with very high degrees. However, rarely any two of these high-degree nodes share exactly the same degree. In fact, we often observe big differences among them. On the contrary, in most graphs there is a large number of nodes with the same small degrees (close to 1). Given such a graph, the `DP` (or `Greedy`) algorithm will be forced to increase the degrees of some of the large-degree nodes a lot, while leaving the degrees of small-degree nodes untouched. In the anonymized sequence thus constructed, a small number of high-degree nodes will need a large number of nodes to connect their newly added edges. However, since the degrees of small-degree nodes does not changed in the anonymized sequence, the demand of edge end-points imposed by the high-degree nodes cannot be facilitated. Therefore, by slightly increasing the degrees of small-degree nodes in $\mathbf{d}$ we force the `DP` (or `Greedy`) algorithm to assign them higher degrees in the anonymized sequence $\widehat{\mathbf{d}}$. In that way, there are more additional free edges end-points to connect with the anonymized high-degree nodes.

From our experiments on a large spectrum of synthetic and real-world data, we observe that, in most cases, the extra edge-additions incurred by the `Probing` procedure are negligible. That is, the degree sequences produced by the `DP` (or `Greedy`) are almost realizable, and more importantly, realizable with respect to the input graph G. Therefore, the `Probing` is rarely invoked, and even if it is invoked, only a very small number of repetitions are needed. We further discuss this in the experimental section of the paper.

6. RELAXED GRAPH CONSTRUCTION

The `Supergraph` algorithm presented in the previous section extends the input graph $G(V,E)$ by adding additional edges. It guarantees that the output graph $\widehat{G}(V,\widehat{E})$ be k-degree anonymous and $E \subseteq \widehat{E}$. However, the requirement that $E \subseteq \widehat{E}$ may be too strict to satisfy. In many cases, we are content with a degree-anonymous graph where $\widehat{E} \cap E \approx E$, which means that most of the edges of the original graph appear in the degree-anonymous graph as well, but not necessarily all of them. We call this version of the problem the Relaxed Graph Construction problem.

6.1 The `Greedy_Swap` algorithm

Let $\widehat{\mathbf{d}}$ be a k-anonymous degree sequence output by `DP` (or `Greedy`) algorithm. Let us additionally assume for now, that $\widehat{\mathbf{d}}$ is realizable so that the `ConstructGraph` algorithm with input $\widehat{\mathbf{d}}$, outputs a simple graph $\widehat{G}_0(V,\widehat{E}_0)$ with degree sequence exactly $\widehat{\mathbf{d}}$. Although $\widehat{G}_0$ is k-degree anonymous, its structure may be quite different from the original graph $G(V,E)$. The `Greedy_Swap` algorithm is a greedy heuristic that given $\widehat{G}_0$ and G, it transforms $\widehat{G}_0$ into $\widehat{G}(V,\widehat{E})$ with degree sequence $\mathbf{d}_{\widehat{G}} = \widehat{\mathbf{d}} = \mathbf{d}_{\widehat{G}_0}$ and $E \cap \widehat{E} \approx E$.

At every step i, the graph $\widehat{G}_{i-1}(V,\widehat{E}_{i-1})$ is transformed into the graph $\widehat{G}_i(V,E_i)$ such that $\widehat{\mathbf{d}}_{\widehat{G}_0} = \widehat{\mathbf{d}}_{\widehat{G}_{i-1}} = \widehat{\mathbf{d}}_{\widehat{G}_i} = \widehat{\mathbf{d}}$ and $|\widehat{E}_i \cap E| > |\widehat{E}_{i-1} \cap E|$. The transformation is made using *valid swap* operations defined as follows:

Definition 5. *Consider a graph $\widehat{G}_i(V,\widehat{E}_i)$. A* valid swap *operation is defined by four vertices i,j,k and l of $\widehat{G}_i(V,\widehat{E}_i)$ such that $(i,k) \in \widehat{E}_i$ and $(j,l) \in \widehat{E}_i$ and $(i,j) \notin \widehat{E}_i$ and $(k,l) \notin \widehat{E}_i$, or, $(i,l) \notin \widehat{E}_i$ and $(j,k) \notin \widehat{E}_i$. A valid swap operation transforms $\widehat{G}_i$ to $\widehat{G}_{i+1}$ by updating the edges as follows*

$$\widehat{E}_{i+1} \leftarrow \widehat{E}_i \setminus \{(i,k),(j,l)\} \cup \{(i,j),(k,l)\}, \quad \textit{or}$$
$$\widehat{E}_{i+1} \leftarrow \widehat{E}_i \setminus \{(i,k),(j,l)\} \cup \{(i,l),(j,k)\}.$$

A visual illustration of the swap operation is shown in Figure 2. It is clear that performing valid swaps on a graph leaves the degree sequence of the graph intact. The pseudocode for the `Greedy_Swap` algorithm is given in Algorithm 3. At each iteration of the algorithm, the swappable pair of edges e_1 and e_2 is picked to be swapped to edges e_1' and e_2'. The selection among the possible valid swaps is made so that the pair with the maximum (c) increase in the edge intersection is picked. The `Greedy_Swap` algorithm halts when there are no more valid swaps that can increase the size of the edge intersection.

Algorithm 4 gives the pseudocode of the whole process of solving the Relaxed Graph Construction problem when

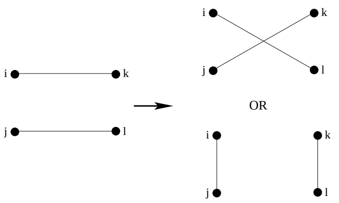


Figure 2: The swap transformation.

Algorithm 3 The `Greedy_Swap` algorithm.

```
    Input: An initial graph $\widehat{G}_0(V, \widehat{E}_0)$ and the input graph $G(V, E)$.
    Output: Graph $\widehat{G}(V, \widehat{E})$ with the same degree sequence as $\widehat{G}_0$, such that $\widehat{E} \cap E \approx E$.
$\widehat{G}(V, \widehat{E}) \leftarrow \widehat{G}_0(V, \widehat{E}_0)$
$(c, (e_1, e_2, e'_1, e'_2)) =$ Find_Max_Swap($\widehat{G}$)
while $c > 0$ do
    $\widehat{E} = \widehat{E} \setminus \{e_1, e_2\} \cup \{e'_1, e'_2\}$
    $(c, (e_1, e_2, e'_1, e'_2)) =$ Find_Max_Swap
return $\widehat{G}$
```

the degree sequence $\widehat{\mathbf{d}}$ is realizable. The first step involves a call to the `ConstructGraph` algorithm, which we have described in Section 5.1, Algorithm 1. The `ConstructGraph` algorithm will return a graph $\widehat{G}_0$ with degree distribution $\widehat{\mathbf{d}}$. The `Greedy_Swap` algorithm is then invoked with input the constructed graph $\widehat{G}_0$. The final output of the process is a k-degree anonymous graph that has degree sequence $\widehat{\mathbf{d}}$ and large overlap in its set of edges with the original graph.

A naive implementation of the algorithm would require time $O(I|\widehat{E}_0|^2)$, where I is the number of iterations of the greedy step and $|\widehat{E}_0|$ the number of edges in the input graph graph. Given that $|\widehat{E}_0| = O(n^2)$, the running time of the `Greedy_Swap` algorithm could be $O(n^4)$, which is daunting for large graphs. However, we employ a simple sampling procedure that considerably improves the running time. Instead of doing the greedy search over the set of all possible edges, we uniformly at random pick a subset of size $O(log|\widehat{E}_0|) = O(\log n)$ of the edges and run the algorithm on those. This reduces the running time of the greedy algorithm to $O(I \log^2 n)$, which makes it efficient even for very large graphs. As we show in our experimental evaluation, the `Greedy_Swap` algorithm performs very well in practice, even in cases where it starts with graph $\widehat{G}_0$ that shares small number of edges with G.

The `Probing` Scheme for `Greedy_Swap`: As in the case of the `Supergraph` algorithm, it is possible that the `ConstructGraph` algorithm outputs a "No" or "Unknown". In this case we invoke a `Probing` procedure identical to the one we have described in Section 5.3.

6.2 The `Priority` algorithm

We additionally show a simple modification of the `ConstructGraph` algorithm that allows the construction of degree anonymous graphs with similar high edge intersection with the original graph directly, without using `Greedy_Swap`. We call this algorithm the `Priority` algorithm because during the graph-construction phase, it gives priority to already existing edges in the input graph $G(V, E)$. The intersections we obtain using the `Priority` algorithm are comparable, if not better, to the intersections we obtain using the `Greedy_Swap` algorithm. However, the `Priority` algorithm is less computationally demanding than the naive implementation of the `Greedy_Swap` procedure.

Algorithm 4 An overall algorithm for solving the Relaxed Graph Construction problem; the realizable case.

```
    Input: A realizable degree sequence $\widehat{\mathbf{d}}$ of length $n$.
    Output: A graph $\widehat{G}(V, E')$ with degree sequence $\widehat{\mathbf{d}}$ and $E \cap E' \approx E$.
$\widehat{G}_0 =$ ConstructGraph($\widehat{\mathbf{d}}$)
$\widehat{G} =$ Greedy_Swap($\widehat{G}_0$)
```

The `Priority` algorithm is similar to the `ConstructGraph`. Recall that the `ConstructGraph` algorithm at every step picks a node v with residual degree $\widehat{\mathbf{d}}(v)$ and connects it to $\widehat{\mathbf{d}}(v)$ nodes with the highest residual degree. `Priority` works in a similar manner with the only difference that it makes two passes over the sorted degree sequence $\widehat{\mathbf{d}}$ of the remaining nodes. In the first pass, it considers only nodes v' such that $\widehat{\mathbf{d}}(v') > 0$ and edge $(v, v') \in E$. If there are less that $\widehat{\mathbf{d}}(v)$ such nodes it makes a second pass considering nodes v' such that $\widehat{\mathbf{d}}(v') > 0$ and edge $(v, v') \notin E$. In that way `Priority` tries to connect node v to as many of his neighbors in the input graph G. The graphs thus constructed share lots of edges with the input graph. In terms of running time, the `Priority` algorithm is the same as `ConstructGraph`.

In the case where `Priority` fails to construct a graph by reaching a dead-end in the edge-allocation process, the `Probing` scheme is employed; and random noise addition is enforced until the `Priority` algorithm outputs a valid graph.

7. EXPERIMENTS

In this section we evaluate the performance of the proposed graph-anonymization algorithms.

7.1 Datasets

We use both synthetic and real-world datasets. For the experiments with synthetic datasets, we generate *random*, *small-world* and *scale-free* graphs.

Random graphs: Random graphs are graphs with nodes randomly connected to each other with probability p. Given the number of nodes n and the parameter p, a random graph is generated by creating an edge between each pair of nodes u and v with probability p. We use $\mathcal{G}_R$ to denote the family of graphs generated by this data-generation model and G_R to denote a member of the family.

Small-world graphs: A small-world graph is a type of graph in which most nodes are not neighbors of one another, but most nodes can be reached from every other by a small number of hops. This kind of graphs have large *clustering coefficient* (CC) that is significantly higher than expected by random chance, and small *average path length* (APL) that is close to that of an equivalent random graph. The average path length is defined as the average length of the shortest path between all pairs of reachable nodes. The clustering coefficient is defined as the average fraction of pairs of neighbors of a node that are also connected to each other. These two indices, along with the degree distribution, are considered as standard measures in graph-analysis studies. We generate small-world graphs using the model proposed in [16]. We denote by $\mathcal{G}_W$ the family of graphs generated by this model and G_W the members of this family. The data-generation process is controlled by a parameter α that determines the extent to which the graph exhibits community structure. Values of α in the range $[5, 7]$ generate small-world graphs. We have additionally conducted experiments with small-world graphs generated using the alternative model proposed in [17]. However, since the results we obtained are very similar to the results obtained by using graphs in $\mathcal{G}_W$, we do not report them here due to space limitations.

Scale-free graphs: The scale-free graphs correspond to graphs with power-law degree distribution. In a power-law graph the probability that a node has degree d is proportional to $d^{-\gamma}$. The power-law distribution is determined by the exponent γ. The value of γ may vary, taking values between 2 and 3 for most real networks. We use the model proposed by Barabási and Albert [3] to generate scale-free graphs. The graph-generation process proceeds by inserting nodes sequentially. Each new node is initially connected to ℓ already existing nodes with probability proportional to their degree. We use $\mathcal{G}_{BS}$ to denote the family of graphs generated by this model and G_{BS} to denote members of the family.

The structures of the graphs in $\mathcal{G}_W$ and $\mathcal{G}_{BS}$ are different: graphs in $\mathcal{G}_W$ do not exhibit power-law degree distributions while graphs in $\mathcal{G}_{BS}$ have small clustering coefficient.

For the real-world data, we use the **prefuse**, the **enron**, the **powergrid** and the **co-authors** graphs.

Prefuse graph: This graph is used as an example of small social network in the *Prefuse project*. The graph is available at the project's web page: `http://prefuse.org/` and it consists of 129 nodes.

Enron graph: The Enron email graph (available at `http://www.cs.cmu.edu/enron/`) is derived from a corpus of emails sent to and from managers at Enron Corporation. This data was originally made public by the Federal Energy Regulatory Commission. The dataset contains 151 users. An edge between two users is added if they have corresponded at least five times.

Powergrid graph: In this graph, the nodes represent generators, transformers and substations in a powergrid network; the edges represent high-voltage transmission lines between them. The dataset is available at `http://www.cs.helsinki.fi/u/tsaparas/MACN2006/`.

Co-authors graph: The co-authors dataset consists of 7955 authors of papers in database and theory conferences and it is available at the collection of Computer Science Bibliographies at `http://liinwww.ira.uka.de/bibliography/`. The co-authors graph is constructed by creating undirected edges between authors that have co-authored paper.

Table 1 summarizes the properties of the graphs we used for our experiments. All the graphs are simple, unweighted and undirected.

7.2 Evaluating Degree Anonymization algorithms

The goal of our first experiment is to compare the qualitative performance of the `Greedy` and `DP` algorithms in solving the Degree Anonymization problem. We report the results in terms of the *performance ratio* R which is the ratio of the cost of the solution obtained by the `Greedy` algorithm to the optimal cost obtained by the `DP` algorithm. That is,

	#Nodes	#Edges	APL	CC
$\mathcal{G}_W$ ($\alpha = 6$)	1000	5000	9.15	0.77
$\mathcal{G}_R$	1000	5000	3.27	0.01
$\mathcal{G}_{BS}$ ($\gamma = 3$)	1000	2995	3.57	0.02
prefuse	129	161	3.17	0.44
enron	151	502	3.32	0.46
powergrid	4941	6594	9.12	0.10
co-authors	7955	10055	6.00	0.64

Table 1: Structural properties of the graphs used for the experiments.

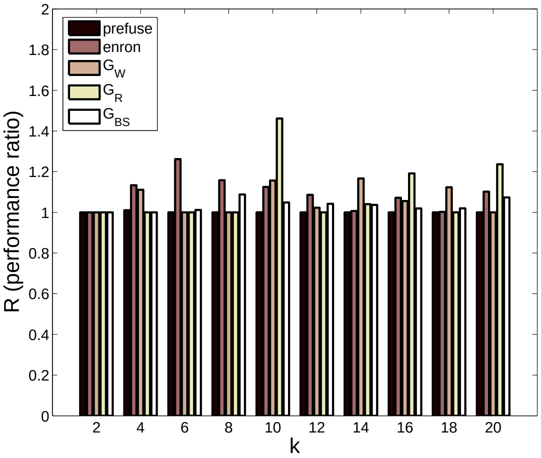


Figure 3: Performance ratio (R) of the `Greedy` algorithm with respect to the optimal `DP` for solving the Degree Anonymization problem; values of $k \in \{2, 4, 6, \ldots, 20\}$. The ratio is computed for (a)real datasets enron and prefuse and (b) synthetic instances of graphs $G_W \in \mathcal{G}_W$, $G_{BS} \in \mathcal{G}_{BS}$ and $G_R \in \mathcal{G}_R$.

$R = \frac{L_1(\widehat{\mathbf{d}}_{\text{greedy}} - \mathbf{d})}{L_1(\widehat{\mathbf{d}}_{\text{dp}} - \mathbf{d})}$, where $\mathbf{d}$ is the input degree sequence; $\widehat{\mathbf{d}}_{\text{greedy}}$ and $\widehat{\mathbf{d}}_{\text{dp}}$ are the k-anonymous degree sequences output by the `Greedy` and the `DP` algorithms respectively. Values of R close to 1 imply that the two algorithms achieve exactly the same cost, in which case `Greedy` performs optimally. The closer R is to 1, the better the performance of the `Greedy` algorithm.

Figure 3 shows the values of R obtained for different values of $k = \{2, 4, 6, \ldots, 20\}$. We observe that in most cases, R is very close to 1, while the largest (worst) value is no more than 1.5. That is, the `Greedy` algorithm is mostly equivalent to the `DP`.

7.3 Evaluating Graph Construction algorithms

In this section we evaluate the performance of `Supergraph`, `Priority` and `Greedy_Swap` algorithms. The algorithms for Graph Construction and its relaxed version are evaluated together mostly because our expectations from these algorithms are the same; we want them to output a graph that is k-degree anonymous and it is as similar as possible to the original input graph. The k-degree anonymity is guaranteed by construction. For evaluating the structural similarity between the input and output graphs we use a set of evaluation measures that we list below. We report our results for different synthetic and real-world graphs.

Anonymization cost $L_1(\mathbf{d}_A - \mathbf{d})$: This is the L_1 norm of the vector of differences between the k-anonymous degree sequence obtained using algorithm $Algo$ with $Algo \in \{$ `Supergraph`, `Priority`, `Greedy_Swap`$\}$ and the degree sequence of the original graph. The smaller the value of $L_1(\mathbf{d}_A - \mathbf{d})$ the better the qualitative performance of the algorithm. Figures 4(a), 5(a), 6(a) and 7(a) summarize the anonymization cost of the different algorithms as a function of $k = \{5, 10, 15, 20, 25, 50, 100\}$ for synthetic datasets $G_W \in \mathcal{G}_W$ with $\alpha = 6$, $G_{BS} \in \mathcal{G}_{BS}$, and **powergrid** and **co-authors** data. In the plots we also report the *Baseline* cost, which refers to the L_1 difference between the degree sequence of the original graph and the degree sequence obtained as a solution to the Degree Anonymization problem. The Baseline cost is a lower bound of the difference $L_1(\mathbf{d}_A - \mathbf{d})$ for $A \in \{$ `Supergraph`, `Priority`, `Greedy_Swap`$\}$. From the plots we can observe that in most of the datasets the final anonymization cost is very close to the *Baseline* cost, for all three algorithms. This observation implies that in the majority of the cases, the degree sequences that are solutions to the Degree Anonymization problem are also realizable and, more importantly, realizable with respect to the input graph G. Therefore, the `Probing` scheme seems to be rarely invoked, or even if it is invoked, only a small number of repetitions are required.

We note that only in the case of $\mathcal{G}_{BS}$ graphs, $L_1(\mathbf{d}_{\texttt{Supergraph}} - \mathbf{d})$ cost is relatively high for all values of k (see Figure 5(a)). This is due to two reasons: 1) The degrees of graphs in $\mathcal{G}_{BS}$ have a power-law distribution. This causes large differences among the degrees of high-degree nodes, meaning that the degrees of high-degree nodes have to be changed significantly in order to meet the degree-anonymous requirement. 2) The `Supergraph` algorithm constructs the degree-anonymous graph by extending the input graph, and it is the only of our proposed algorithms that tries to comply with all the edge constraints imposed by the input graph. Therefore, it can potentially add more noise than other algorithms that build the graph from scratch.

Clustering Coefficient (CC): We additionally compare the clustering coefficients of the anonymized graphs with the clustering coefficients of the original graphs. Figures 4(b), 5(b), 6(b) and 7(b) summarize our findings. In all plots, there is a constant line appearing, this corresponds to the value of the clustering coefficient of the *original* graph, which is unaffected by the value of k. Note that all the plots show that the values of the clustering coefficient, though different in the degree-anonymous graphs, they never deviate too much from their original values; the largest difference in the CC values from the original values is observed for the **co-author** dataset, where the difference is 0.24 for the degree-anonymous graph produced by the `Greedy_Swap` algorithm when $k = 100$. But even in this case, the other two algorithms output graphs with CC almost equal to that of the original graph.

Note that there is no clear trend on how the CC changes when the graph becomes degree anonymous. Both increments and decrements are observed, however the changes are generally negligible.

Average Path Length (APL): Finally in Figures 4(c), 5(c), 6(c) and 7(c) we report the values of the average path length of the degree-anonymous graphs, anonymized by our three algorithms, for $G_W \in \mathcal{G}_W$, $G_{BS} \in \mathcal{G}_{BS}$, **powergrid** and **co-authors** data and values of $k = 5, 10, 15, 20, 25, 50, 100$. The APL of the original graph is also reported in all plots. As expected, the anonymization process decreases the average path length of the output graph since new connections are added.

	`Supergraph`	`Priority`	`Greedy_Swap`
$\mathcal{G}_W$ ($\alpha = 6$)	1	0.99	0.99 (0.01)
$\mathcal{G}_R$	1	0.99	0.99 (0.01)
$\mathcal{G}_{BS}$ ($\gamma = 3$)	1	0.92	0.93 (0.04)
prefuse	1	0.87	0.83 (0.36)
enron	1	0.95	0.95 (0.16)
powergrid	1	0.99	0.97 (0.01)
co-authors	1	0.99	0.91 (0.01)

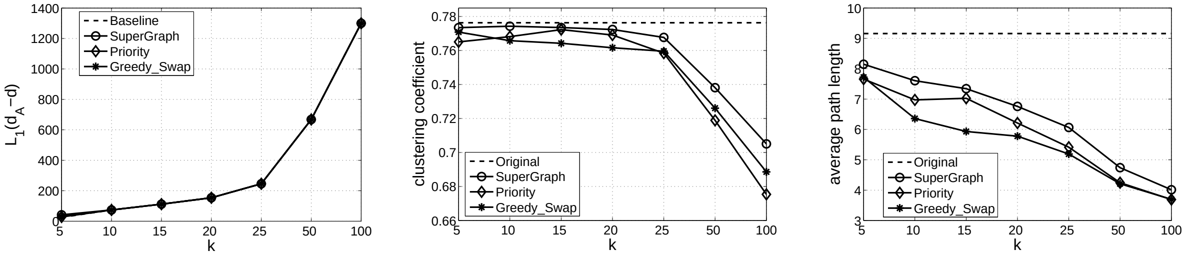
Table 2: Mean of the edge-intersection (EI) values obtained for different algorithms and different datasets. The means for prefuse and enron are computed over a set of different values of $k = 2, 4, 6, \ldots, 20$. The means for other data sets are over $k = 5, 10, 15, 20, 25, 50, 100$. Values in the parenthesis are the EI when `Greedy_Swap` starts.

Very similar results have been obtained for other datasets generated using the random-graph model as well as the **enron** and **prefuse** datasets. However, we omit the corresponding plots due to space constraints.

Edge Intersection (EI): With the term edge intersection we refer to the percentage of edges in the degree-anonymous graphs that are also in the original graph. That is, given original graph $G(V, E)$ and degree-anonymous graph $\widehat{G}(V, \widehat{E})$ we define the edge intersection to be $\text{EI}(E, \widehat{E}) = \frac{|\widehat{E} \cap E|}{|E|}$. This measure is used for the evaluation of the `Priority` and `Greedy_Swap` algorithms that solve the relaxed version of the Graph Construction problem. The value of edge intersection for the `Supergraph` algorithm is by construction always 1. Table 2 summarizes the values of EI for different datasets and different algorithms. The reported values are averages of the edge-intersection value over different values of k. The variance of the observations is always very small; less than 10^{-4}. It is easy to observe that both the `Greedy_Swap` and `Priority` algorithms achieve very high-values of EI, succeeding in constructing graphs that are "almost" supergraphs of the original graph. We also list in the parenthesis the EI values of the graphs used as starting points for the `Greedy_Swap` algorithm. It can be seen that the `Greedy_Swap` algorithm performs extremely well even though it usually starts with graphs with low EI values.

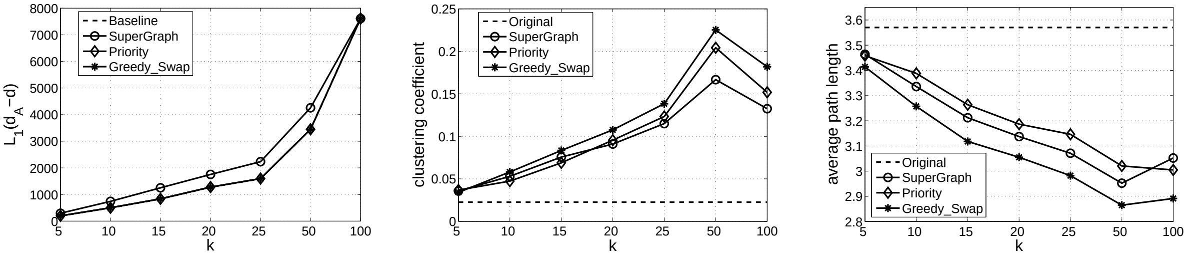
7.3.1 Exploring the Whole Spectrum of $\mathcal{G}_W$ Graphs

Previous experiments demonstrate that the clustering coefficient and average path length are reasonably preserved after anonymization. In Figure 8 we demonstrate this fact even further by showing the values of CC and APL for graphs in $\mathcal{G}_W$ for the different values of the parameter α that guides the data-generation process. More specifically, the two plots in Figure 8 provide evidence that for the whole spectrum of the values of α, the values of CC in the anonymized graph match almost perfectly the original graph. Similar results for APL can be observed when $\alpha \geq 5$. One could claim that these are also the most interesting values of α since they correspond to graphs that either have the small-world property ($\alpha \in [5, 7]$) or are random graphs ($\alpha > 10$). Thus, for most of the interesting cases, the



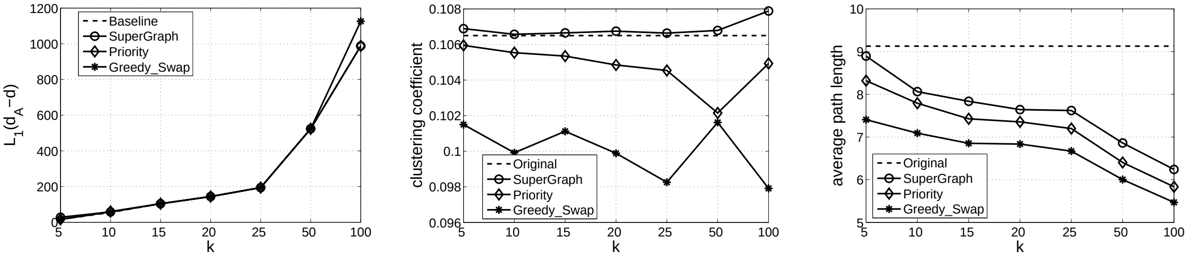
(a) $L_1(\mathbf{d}_A - \mathbf{d})$ as a function of k (b) CC as a function of k (c) APL as a function of k

Figure 4: Synthetic datasets: small-world graphs $G_W \in \mathcal{G}_W$, with $\alpha = 6$.



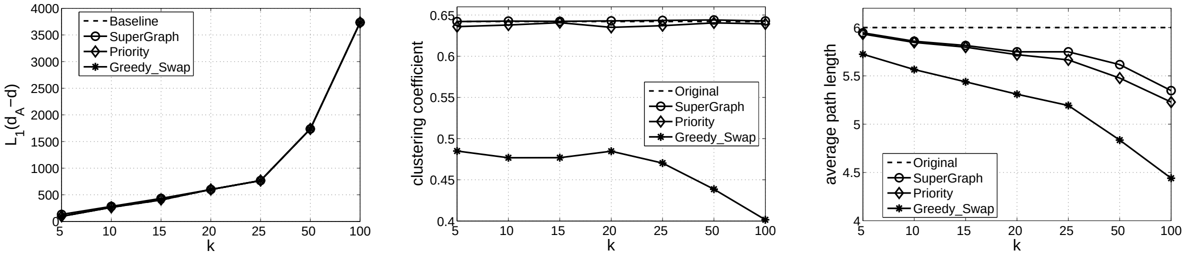
(a) $L_1(\mathbf{d}_A - \mathbf{d})$ as a function of k (b) CC as a function of k (c) APL as a function of k

Figure 5: Synthetic datasets: scale-free graphs $G_{BS} \in \mathcal{G}_{BS}$.



(a) $L_1(\mathbf{d}_A - \mathbf{d})$ as a function of k (b) CC as a function of k (c) APL as a function of k

Figure 6: Real datasets datasets: powergrid data.



(a) $L_1(\mathbf{d}_A - \mathbf{d})$ as a function of k (b) CC as a function of k (c) APL as a function of k

Figure 7: Real datasets datasets: co-authors data.

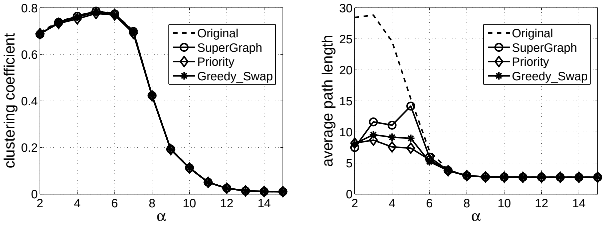


Figure 8: Clustering coefficient and average path length for the whole spectrum of $\mathcal{G}_W$ graphs ($k = 10$, #Nodes = 1000).

anonymization process does not spoil the properties of the input graph, and studying of the anonymized versions of the networks is expected to give valid results.

For small values of α ($\alpha < 5$), the input graph tends to have a overwhelmingly large number of small, isolated, and densely connected components. Most of the nodes can only be reached by nodes outside their community via delegates of the connected components, which we call the "hub". In other words, most shortest paths between nodes are through the hubs. Since these hubs usually have high degrees, the anonymization process tries to anonymize them by adding edges from the them to nodes in other connected components. In that way, the degree-anonymous graph obtained have much smaller values of APL when compared to the input graph.

The results shown in Figure 8 correspond to $k = 10$, however the trends are similar for other values of k as well.

7.3.2 Exploring the Scale-free Graphs

Previous work on the analysis of complex networks has shown that many of the real-world graphs are scale free, *i.e.*, their node degrees follow a power-law distribution. In this section we demonstrate that our anonymization framework does not destroy the power-law property of the original graph if k is not too big. That is, if the input graph has a power-law degree distribution, so does the degree-anonymous version of it.

In Table 3, we report the values of the estimated exponent (γ) of the power-law distribution of the original **co-authors** data and its degree-anonymous counterpart. The new γ values exhibit high degree of similarity to the original one for $k < 15$. For large values of k, a great amount of the nodes in the anonymized graph will have the same degree, and the power-law distribution will change. We claim that this is a natural result for any degree-anonymization algorithm.

	γ		
	`Supergraph`	`Priority`	`Greedy_Swap`
original	**2.07**	**2.07**	**2.07**
$k = 10$	2.26	2.26	2.26
$k = 15$	2.13	2.13	2.13
$k = 20$	1.97	1.97	1.97
$k = 25$	1.83	1.83	1.83
$k = 50$	1.57	1.57	1.57
$k = 100$	1.22	1.22	1.22

Table 3: Real dataset: co-authors graph. Value of the exponent (γ) of the power-law distribution of the original and the k-degree anonymous graph obtained using `Supergraph`, `Priority` and `Greedy_Swap` algorithms, for $k = 10, 15, 20, 25, 50, 100$.

8. EXTENSIONS: SIMULTANEOUS EDGE ADDITIONS AND DELETIONS

So far we have restricted ourselves to edge-additions (or deletions) only. In this section, we show how to extend our framework to allow simultaneous edge additions and deletions. Note that our intention is not to provide a comprehensive algorithmic solution for this extended version of the problem. Our goal is just to show that the framework we developed in the previous sections can be used to solve this generalized problem.

Similar to what we have discussed before, given an input graph $G(V, E)$ with degree sequence $\mathbf{d}$, we proceed as follows:

1. First produce a k-degree anonymous sequence $\widehat{\mathbf{d}}$ from $\mathbf{d}$, such that $L_1(\widehat{\mathbf{d}} - \mathbf{d})$ is minimized.
2. Then construct graph $\widehat{G}(V, \widehat{E})$ with degree sequence $\widehat{\mathbf{d}}$ such that $E \cap \widehat{E}$ is as large as possible.

Step 1 is different from before since the degrees of the nodes in $\widehat{\mathbf{d}}$ can either increase or decrease when compared to their original values in $\mathbf{d}$. Despite this complication, it is easy to show that a dynamic-programming algorithm similar to the one developed in Section 4 can be used to find such a $\widehat{\mathbf{d}}$ that minimizes $L_1(\widehat{\mathbf{d}} - \mathbf{d})$.

The only difference is in the evaluation of $I(\mathbf{d}[i, j])$ that corresponds to the L_1 cost of putting all nodes $i, i+1, \ldots, j$ in the same anonymized group. Note that the indices correspond to the ordering of the nodes in decreasing order of their degree in $\mathbf{d}$. In this case,

$$I(\mathbf{d}[i, j]) = \sum_{\ell=i}^{j} |d^* - \mathbf{d}(\ell)|, \tag{7}$$

where d^* is the degree d such that

$$d^* = \arg\min_d \sum_{\ell=i}^{j} |d - \mathbf{d}(\ell)|.$$

From [11] we know that d^* is the median of the values $\{\mathbf{d}(i), \ldots \mathbf{d}(j)\}$, and therefore given i and j, computing the cost $I(\mathbf{d}[i, j])$ can be done optimally in linear time (see [5] for details).

As before, the dynamic-programming algorithm requires the evaluation of Recursion (4), where $I((\mathbf{d}[i, j]))$ is computed as shown in Equation (7). If the values of $I(\mathbf{d}[i, j])$ are precomputed, then the computation of the recursion takes time $O(nk)$.

Note that again, $I(\mathbf{d}[i, j])$ need not be computed for all pairs of indices i, j; for every i it is enough to consider the j's for which $k \leq j-i+1 \leq 2k-1$. Therefore, the preprocessing step also takes $O(nk)$. Thus, step 1 requires total $O(nk)$ time to be solved optimally using dynamic programming.

As in Section 4, a greedy (non-optimal) algorithm can be used for solving step 1. This greedy algorithm is very similar to the one described in Section 4, an its detailed description is omitted. We only note here, that its running time is also $O(nk)$. At every greedy step the cost of adding a new point to the last group needs to be evaluated. The addition of such a new point makes the group size be either even or odd number. However, it is easy to show that in the first case the median of the points shifts one position to the right, but the only update to the cost is due to the lastly added point. In the second case, the median remains the same and again the only update in the cost is due to the lastly added point. Therefore, the computation of the merging cost requires constant time. For reasons we have discussed in the previous paragraph the cost of starting a new segment is also done in constant time (given a preprocessing step). Therefore, the total running time of the greedy algorithm is also $O(nk)$.

For Step 2 we consider the `Greedy_Swap` algorithm of Section 6.1. Recall that `Greedy_Swap` constructs an initial graph $\widehat{G}_0(V, \widehat{E}_0)$ from a given degree sequence $\widehat{\mathbf{d}}$. Then it transforms $\widehat{G}_0$ into $\widehat{G}(V, \widehat{E})$ with degree sequence $\mathbf{d}_{\widehat{G}} = \widehat{\mathbf{d}} = \mathbf{d}_{\widehat{G}_0}$ and $\widehat{E} \cap E \approx E$. This algorithm implicitly allows for both edge-additions and edge-deletions. Thus, we adopt this algorithm for solving Step 2. For simplicity, we call the combination of the new dynamic programming and `Greedy_Swap` the `Simultaneous_Swap` algorithm.

We performed the same set of experiments as what we have done in the previous section. Due to space constraints, we cannot report all the results. We only compare the quality of the graphs produced by the old `Greedy_Swap` and the new `Simultaneous_Swap`. Tables 4 and 5 briefly summarize these results. From Table 4 we can observe that, for all values of $k = 10, 15, 20, 25, 50, 100$, the exponent (γ) of the power-law distribution of the k-degree anonymous graph obtained by `Simultaneous_Swap` is much closer to that from the original graph than the one obtained by the `Greedy_Swap` algorithm. This tells us that the new algorithm better preserves the power-law properties of the **co-authors** graph. In Table 5, it can be seen that the L_1 cost incurred by the new algorithm is only about 1/3 of the cost by the old algorithm for all the four datasets and all values of k. In the meantime, the clustering coefficient (CC) and average path length (APL) of the new anonymized graph are much closer to the original ones. These results show that, by allowing simultaneous edge additions and deletions, we can produce a k-degree anonymous graph with much lower cost and much higher quality.

	γ	
	`Greedy_Swap`	`Simultaneous_Swap`
original	**2.07**	**2.07**
$k = 10$	2.26	2.45
$k = 15$	2.13	2.33
$k = 20$	1.97	2.28
$k = 25$	1.83	2.25
$k = 50$	1.57	2.05
$k = 100$	1.22	1.92

Table 4: Real dataset: co-authors graph. Value of the exponent (γ) of the power-law distribution of the original and the k-degree anonymous graph obtained using `Greedy_Swap` and `Simultaneous_Swap` algorithms, for $k = 10, 15, 20, 25, 50, 100$.

	$\widetilde{L_1}$	$\widetilde{\mathrm{CC}}$	$\widetilde{\mathrm{APL}}$
$\mathcal{G}_W$ $(\alpha = 6)$	0.22 (0.0049)	0.62 (0.0247)	0.43 (0.0162)
$\mathcal{G}_{BS}$ $(\gamma = 3)$	0.28 (0.0216)	0.09 (0.0008)	0.34 (0.0318)
powergrid	0.36 (0.0755)	0.69 (0.0744)	0.74 (0.0242)
co-authors	0.27 (0.0047)	0.94 (0.0063)	0.34 (0.0351)

Table 5: Graph-quality improvement obtained by the `Simultaneous_Swap` algorithm in terms of the L_1 cost, clustering coefficient and average path length. $\widetilde{L_1}$ is defined as the average of $\frac{L_1(\mathbf{d}_{\texttt{Simultaneous\_Swap}} - \mathbf{d})}{L_1(\mathbf{d}_{\texttt{Greedy\_Swap}} - \mathbf{d})}$ over $k = 5, 10, 15, 20, 25, 50$. $\widetilde{\mathbf{CC}}$ is defined as the average of $\frac{|\mathbf{CC}_{\texttt{Simultaneous\_Swap}} - \mathbf{CC}_{\texttt{original}}|}{|\mathbf{CC}_{\texttt{Greedy\_Swap}} - \mathbf{CC}_{\texttt{original}}|}$ over k. $\widetilde{\mathbf{APL}}$ is defined as the average of $\frac{|\mathbf{APL}_{\texttt{Simultaneous\_Swap}} - \mathbf{APL}_{\texttt{original}}|}{|\mathbf{APL}_{\texttt{Greedy\_Swap}} - \mathbf{APL}_{\texttt{original}}|}$ over k. The closer these values are to 0, the better the performance of the `Simultaneous_Swap`. Values in the parenthesis are the variance.

9. CONCLUSIONS

The degree of a node in a graph, among other structural characteristics, can to a large extent distinguish the node from other nodes. In this paper, we studied a specific graph-anonymity notion that prevents the re-identification of individuals by an attacker with certain prior knowledge of the degrees. We formally defined the Graph Anonymization problem that, given an input graph asks for the minimum number of edge additions (or deletions) that allow the transformation of the input to a degree-anonymous graph; *i.e.*, a graph in which every node shares the same degree with $k - 1$ other nodes. We decomposed this problem into two subproblems and proposed simple and efficient algorithms for solving them. We applied our algorithms to a set of synthetic and real-world graph data and demonstrated the utility of the degree-anonymous graphs as well as the efficiency of our methods. Finally, we extended our algorithms to allow simultaneous edge additions and deletions.

Before concluding this paper, we would like to note that, compared with existing data anonymization and perturbation techniques for tabular data, dealing with graphs is a much more challenging task. In tabular data, each tuple can be viewed as an independent sample from some distribution. However, in a graph, all the nodes and edges are correlated; a single change of an edge and/or a node can spread across the whole network. Moreover, in graphs it is difficult to model the capability of an attacker. Any topological structure of the graph can be potentially used to derive private information. Finally, it is difficult to measure the utility of a graph. We are not aware of any effective metrics to quantify the information loss incurred by the changes of its nodes and edges.

In this paper, we tried to address some of these issues using simple and intuitive notions. Lots of additional work needs to be done in order to develop theoretically and practically sound privacy models for graphs.

Acknowledgments

We would like to thank Ken Clarkson for pointing out a faster $O(n)$ time dynamic-programming algorithm for evaluating Recursion (4).

10. REFERENCES

[1] Aggarwal, C. C., and Yu, P. S. *Privacy-Preserving Data Mining: Models and Algorithms*, vol. 34 of *Advances in Database Systems*. Springer, 2008.

[2] Backstrom, L., Dwork, C., and Kleinberg, J. M. Wherefore art thou R3579X?: Anonymized social networks, hidden patterns, and structural steganography. In *Proceedings of the 16th International Conference on World Wide Web (WWW'07)* (Alberta, Canada, May 2007), pp. 181–190.

[3] Barabási, A.-L., and Albert, R. Emergence of scaling in random networks. *Science 286*, 5439 (October 1999), 509–512.

[4] Bayardo, R. J., and Agrawal, R. Data privacy through optimal k-anonymization. In *Proceedings of the 21st International Conference on Data Engineering (ICDE'05)* (Tokyo, Japan, April 2005), pp. 217–228.

[5] Cormen, T., Leiserson, C., and Rivest, R. *Introduction to Algorithms.* MIT Press, 1990.

[6] Erdös, P., and Gallai, T. Graphs with prescribed degrees of vertices. *Mat. Lapok* (1960).

[7] Frikken, K. B., and Golle, P. Private social network analysis: How to assemble pieces of a graph privately. In *Proceedings of the 5th ACM Workshop on Privacy in Electronic Society (WPES'06)* (Alexandria, VA, 2006), pp. 89–98.

[8] Getoor, L., and Diehl, C. P. Link mining: a survey. *ACM SIGKDD Explorations Newsletter 7*, 2 (2005), 3–12.

[9] Hakimi, S. L. On realizability of a set of integers as degrees of the vertices of a linear graph. *Journal of the Society for Industrial and Applied Mathematics 10*, 3 (1962), 496–506.

[10] Hay, M., Miklau, G., Jensen, D., Weis, P., and Srivastava, S. Anonymizing social networks. Technical report, University of Massachusetts Amherst, 2007.

[11] Lee, Y.-S. Graphical demonstration of an optimality property of the median. *The American Statistician 49*, 4 (November 1995), 369–372.

[12] Machanavajjhala, A., Gehrke, J., Kifer, D., and Venkitasubramaniam, M. l-diversity: Privacy beyond k-anonymity. In *Proceedings of the 22nd International Conference on Data Engineering (ICDE'06)* (Atlanta, GA, April 2006), p. 24.

[13] Meyerson, A., and Williams, R. On the complexity of optimal k-anonymity. In *Proceedings of the Twenty-third ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems (PODS'04)* (Paris, France, 2004), pp. 223–228.

[14] Pei, J., and Zhou, B. Preserving privacy in social networks against neighborhood attacks. In *Proceedings of the 24th International Conference on Data Engineering (ICDE'08)* (Cancun, Mexico, April 2008).

[15] Samarati, P., and Sweeney, L. Generalizing data to provide anonymity when disclosing information. In *Proceedings of the Seventeenth ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems (PODS'98)* (Seattle, WA, 1998), p. 188.

[16] Watts, D. J. Networks, dynamics, and the small-world phenomenon. *American Journal of Sociology 105*, 2 (September 1999), 493–527.

[17] Watts, D. J., and Strogatz, S. H. Collective dynamics of small-world networks. *Nature 393*, 6684 (June 1998), 409–410.

[18] Ying, X., and Wu, X. Randomizing social networks: a spectrum preserving approach. In *Proceedings of SIAM International Conference on Data Mining (SDM'08)* (Atlanta, GA, April 2008).

[19] Zheleva, E., and Getoor, L. Preserving the privacy of sensitive relationships in graph data. In *Proceedings of the International Workshop on Privacy, Security, and Trust in KDD (PinKDD'07)* (San Jose, CA, August 2007).