Fast Identity Anonymization on Graphs

Xuesong Lu, Yi Song, and Stéphane Bressan

School of Computing, National University of Singapore
{xuesong,songyi,steph}@nus.edu.sg

Abstract. Liu and Terzi proposed the notion of k-degree anonymity to address the problem of identity anonymization in graphs. A graph is k-degree anonymous if and only if each of its vertices has the same degree as that of, at least, k-1 other vertices. The anonymization problem is to transform a non-k-degree anonymous graph into a k-degree anonymous graph by adding or deleting a minimum number of edges.

Liu and Terzi proposed an algorithm that remains a reference for k-degree anonymization. The algorithm consists of two phases. The first phase anonymizes the degree sequence of the original graph. The second phase constructs a k-degree anonymous graph with the anonymized degree sequence by adding edges to the original graph. In this work, we propose a greedy algorithm that anonymizes the original graph by simultaneously adding edges to the original graph and anonymizing its degree sequence. We thereby avoid testing the realizability of the degree sequence, which is a time consuming operation. We empirically and comparatively evaluate our new algorithm. The experimental results show that our algorithm is indeed more efficient and more effective than the algorithm proposed by Liu and Terzi on large real graphs.

1 Introduction

The data contained in social media raise the interest of marketers, politicians and sociology researchers, as well as hackers and terrorists. The mining and analysis of the graphs formed by entities and connections in online social networks, messaging systems and the like, should only benefit legitimate users while no one, and, more critically, no malicious user should be able to access or infer private information.

Researchers, such as the authors of [1], quickly observed that simply hiding the identities of the users in a network may not suffice to protect privacy. Indeed, the structure of the graph itself may leak sufficient information for an adversary with minimal external knowledge to infer identity of users, for instance. Consequently several graph anonymization algorithms have been proposed that not only remove identity but also perturb the graph content and structure while trying to preserve utility for the sake of mining and analysis.

1.1 The K-Degree Anonymization Algorithm

Liu and Terzi [13] address the issue of identity disclosure of network users by adversaries with the background knowledge of nodes degree. To prevent such

© Springer-Verlag Berlin Heidelberg 2012

attacks they propose the problem of *k-degree anonymity.* A graph is said to be *k-degree anonymous* when each vertex in the graph has the same degree as at least $k-1$ other vertices. In other words, any vertex cannot be identified with probability higher than $1/k$ if the adversary has the degree information of the graph. The degree sequence of such a graph is said to be *k-anonymous.* Then the problem is to transform a non-k-degree anonymous graph into a k-degree anonymous graph by adding or deleting a minimum number of edges. For the sake of simplicity, we consider only the addition of edges. Liu and Terzi [13] propose a two-phase algorithm. The first phase (degree anonymization) anonymizes the degree sequence of the original graph to be k-anonymous. They propose a dynamic programming algorithm which reproduces the algorithm in [9]. The second phase (graph construction) constructs a k-degree anonymous graph with the anonymized degree sequence based on the original graph. We call this algorithm *K-Degree Anonymization* (KDA).

Typically, the degree distribution of large real world graphs follows a power-law or exponential distribution (see [2,6]). Consequently, there are few vertices with very large degrees and many vertices with the same small degrees. Moreover, the difference between consecutive large degrees is great.

The dynamic programming in the degree anonymization phase of KDA is designed to minimize the residual degrees, namely the difference between the original degrees and the degrees in the anonymized degree sequence. On large real world graphs, it generates a sequence at the expense of large residual degrees for large original degrees, as the differences between these large original degrees are great. It also generates the sequence with a small number of changes from the original degree sequence, as many vertices with small original degrees are already k-anonymous. It may then be impossible to compensate the large residual degrees. The sequence is then unrealizable. Our experience suggests that, unlike what is claimed by Liu and Terzi, this situation is frequent. For instance, as illustrated in the example below, their dynamic programming in the degree anonymization phase does not generate a realizable degree sequence from the given data set.

Example 1. Email-Enron is the network of Enron employees who have communicated by the Enron email. It is an undirected graph with 36692 vertices and 367662 edges. Each vertex represents an email address. An edge connects a pair of vertices if there is at least one email communication between the corresponding email users. The dataset is available at http://snap.stanford.edu/data/email-Enron.html. The first 10 degrees of its degree sequence in descending order are 1383, 1367, 1261, 1245, 1244, 1143, 1099, 1068, 1026, 924. After the degree sequence is anonymized for $k=5$, the 10 degrees become 1383, 1383, 1383, 1383, 1383, 1143, 1143, 1143, 1143, 1143. We see that the degree of the last vertex is increased by $1143-924=219$. This means that 219 vertices with residual degree are required in order to compensate the residual degree of 219. However, during the anonymization the number of vertices that have their degrees increased is 212. Moreover, most of these vertices are those with small original degrees which are already connected to that vertex. Thus there are no enough vertices with residual degrees to be wired to the last vertex. The k-anonymous degree sequence is unrealizable.

Moreover, even if the anonymized degree sequence is realizable, the graph construction phase of the algorithm may not succeed.

Liu and Terzi cater for these two situations by proposing a `Probing` scheme that operates small random changes on the degree sequence until it is realizable and the graph is constructed. Our experience shows that a large number of `Probing` steps are in effect necessary to obtain a realizable sequence for practical graphs. After each `Probing` is invoked, the realizability-testing is conducted. The testing has a time complexity $O(n^2)$ where n is the number of vertices. As `Probing` is invoked for a large number of repetitions, the complete algorithm is very inefficient.

1.2 Our Contributions

Motivated by the above observations, we study fast k-degree anonymization on graphs at the risk of marginally increasing the cost of degree anonymization, i.e., the edit distance between the anonymized graph and the original graph.

We propose a greedy algorithm that anonymizes the original graph by simultaneously adding edges to the original graph and anonymizing its degree sequence. We thereby avoid realizability testing by effectively interleaving the anonymization of the degree sequence with the construction of the anonymized graph in groups of vertices.

Our algorithm results in larger edit distance on small graphs but smaller edit distance on large graphs compared with the algorithm of Liu and Terzi. Our algorithm is much more efficient than the algorithm of Liu and Terzi.

The rest of the paper is organized as follows. Section 2 discusses background and related work on graph anonymization. Section 3 presents our novel algorithm. Section 4 empirically and comparatively evaluates the performance of our algorithm. Finally, we conclude in Section 5.

2 Related Work

The need for more involved graph anonymization stems from the shortcoming of naive anonymization [1]. Naive anonymization replaces identities of vertices with synthetic identifiers before publishing the graph. With minimal external knowledge, adversaries may be able to recover these identities from the graph structure. Hay et al. [10] further study the problem and quantify the risk of re-identification via graph structural queries.

Several graph anonymization techniques have then been proposed [13,23,5,18,17].

Generally speaking these techniques consists in modifying the graph structure so as to prevent re-identification while preserving sufficient utility. Most of these works have built upon the concept of *k-anonymity* [16], which was first introduced to anonymize relational micro-data.

Several works consider rather general structural attacks. Namely, they try and protect against attacks from adversaries with diverse background structural knowledge. Hay et al. [10] propose the *k-candidate anonymity* model that requires that, for any structural query, there be at least k candidate vertices. Liu and Terzi [13] suggest to make the degree sequence *k-anonymous* so that each vertex in the graph has the same degree as at least $k-1$ other vertices. Vertices in *k-degree anonymous* graph cannot be identified with probability higher than $1/k$. Zou et al. [23] propose to modify the graph to be *k-automorphic* before releasing. Any vertex in such a graph cannot be distinguished from other at least $k-1$ vertices via graph structure, thus all kinds of structure attacks are prevented. The modifications are achieved by addition and deletion of edges and, occasionally, addition of vertices. Similarly, Wu et al. [18] propose the *k-symmetry* model to prevent identity disclosure. In a *k-symmetric* graph every vertex is structurally indistinguishable from at least $k-1$ other vertices. Cheng et al. [5] consider the same problem as Zou et al. [23], as they also try to prevent general structural attacks on published graphs and protect against not only identity but also link and attribute disclosure. They propose the *k-isomorphism* model that forms k pairwise isomorphic subgraphs, to provide sufficient privacy guarantee.

Although these approaches take all kinds of structural attacks into consideration, and thus provide strong privacy guarantee, they often incur many changes and therefore potentially a loss of utility. Song et al. [15] make a variety of graph structure measurements on social networks both before and after anonymization. They examine the state-of-the-art anonymization algorithm, *k-automorphism* algorithm [23]. The significant changes on degree distribution, diameter, density, algebraic connectivity and other metrics indicate the anonymization can perturb graph structure to a large degree and thus greatly impair data utility, even though strong privacy guarantee is provided.

K-degree anonymity [13] specifically focuses on attacks leveraging an adversary's background knowledge of degree. By not being concerned with other structural attacks, it can achieve privacy with fewer modification and, therefore, at a lesser utility cost. Stronger privacy guarantees than those of *k-degree anonymity* are provided by models such as k^2*-degree anonymity* by Tai et al. [17]. A k^2*-degree anonymous* graph prevents re-identification by adversaries with the background knowledge of the degrees of two vertices connected by an edge.

While the above works and some others focus on identity disclosure [22,3,8] some research studied link discosure [21,19,11,14]. Several works [4,7,12,20] works also look at graph models other than simple graph, for instance bipartite graphs.

3 The Algorithm

The algorithm that we propose simultaneously adds edges to the original graph and anonymizes its degree sequence in groups of vertices.

The main idea of the algorithm is to cluster and anonymize the vertices of the original graph into several anonymization groups. Each group contains at least k vertices. The graph is transformed so as that vertices in each group have the same degree. In order to achieve small local degree anonymization cost, the vertices in each group should have similar degrees. For this reason, our algorithm sorts, examines and groups the vertices in the descending order of their degrees in the original graph. This choice is motivated by the observation that practical graphs often follow a power or exponential law with a long tail according to which many vertices have and share a small degree. We therefore wire vertices with larger degree to vertices with smaller degree in groups until the degree sequence is k-anonymous, if it can be achieved.

Let $\boldsymbol{v}$ be the sorted vertex sequence. The `greedy_examination` algorithm clusters vertices into an anonymization group. An anonymization group is the smallest subset of $\boldsymbol{v}$ that has at least k members and whose members have a degree strictly higher than the remaining vertices. The cost of the subsequent anonymization of such a group is necessarily the sum of residual degrees after anonymization, namely, for an anonymization group $(v_i, \cdots, v_j)$ in descending order of degrees, $\sum_{l=i}^{j}(d_i - d_l)$, where d_l is the degree of vertex v_l.

The `edge_creation` algorithm adds edges in order to anonymize the vertices in a group. It wires vertices with insufficient degree in the anonymization group to vertices with lesser degree in $\boldsymbol{v}$ until all vertices in the group have the same degree d_i for an anonymization group $(v_i, \cdots, v_j)$ in descending order of degrees. However, we constrain the algorithm never to increase the degrees of vertices in and outside the group beyond that of the highest degree in the anonymization group, namely, d_i, for an anonymization group $(v_i, \cdots, v_j)$ in descending order of degrees. After adding edges, $\boldsymbol{v}$ is reordered according to the new degrees. At the next iteration, vertices outside the group may be further added to the newly anonymized group by `greedy_examination`, if their degree is d_i.

The anonymization group is now k-anonymous, because it contains at least k vertices with degree d_i.

The design choices in the algorithms above, in particular the wiring constraint, have been made in order to minimize the need for reordering $\boldsymbol{v}$ and to allow as sequential as possible a processing of vertices and groups.

Because of the wiring constraint, it is however possible that the above deterministic process does not find enough vertices to wire. Therefore it does not construct a graph with an anonymized degree sequence. The `relaxed_edge_creation` algorithm caters for such possible failures. It relaxes the wiring constraint.

The complete algorithm, *Fast K-Degree Anonymization* (FKDA), combines the above three algorithms. FKDA always constructs a k-degree anonymous graph.

3.1 The `greedy_examination` Algorithm

At each iteration, the input to `greedy_examination` is a sequence of vertices $\boldsymbol{v}$ of length n sorted in the descending order of their degrees, an index i such that the vertex sequence $(v_1, v_2, \ldots, v_{i-1})$ has been k-anonymous and the value of k. The output is a number n_a such that the vertices $v_i, v_{i+1}, \ldots, v_{i+n_a-1}$ are selected to be clustered into an anonymization group. Then `greedy_examination` passes $\boldsymbol{v}$, i and n_a to `edge_creation`.

Algorithm 1. The `greedy_examination` algorithm

```
Input: v: a sequence of n vertices sorted in the descending order of their
       degrees, i: an index, k: the value of anonymity.
Output: n_a: the number of consecutive vertices that are going to be
        anonymized.
1  Find the first vertex $v_j$ such that $d_j < d_i$;
2  if $v_j$ is not found then
3      $n_a = n - i + 1$;
4  else
5      if $d_i = d_{i-1}$ then
6          if $n - j + 1 < k$ then $n_a = n - i + 1$;
7          else $n_a = j - i$;
8      else
9          if $n - i + 1 < 2k$ or $n - j + 1 < k$ then $n_a = n - i + 1$;
10         else $n_a = max(k, j - i)$;
11     end
12 end
13 Return $n_a$;
```

The algorithm begins with an sequential examination of $\boldsymbol{v}$ starting from v_i, until v_j such that $d_j < d_i$. If there is no such v_j found, $v_i, v_{i+1}, \ldots, v_n$ have the same degree already. Below we show that there are at least k vertices from v_i to v_n. Thereby $\boldsymbol{v}$ is already k-anonymous. n_a is set to be $n - i + 1$, i.e., the number of all the remaining vertices. If v_j is found, there are two different cases depending on the result of comparison between d_i and d_{i-1}[1]. If $d_i = d_{i-1}$[2] which means that v_i has the same degree as the degree of the last anonymization group, `greedy_examination` clusters $v_i, v_{i+1}, \ldots, v_{j-1}$ in a group and merges them into the last anonymization group. Then n_a is set to be $j - i$. However, there is an exception when $n - j + 1 < k$. This means that there are less than k vertices after the current group. These vertices cannot be transformed to be k-anonymous in a separated group. Thus `greedy_examination` has to cluster $v_i, v_{i+1}, \ldots, v_n$ into a group. n_a is set to be $n - i + 1$. In the other case where $d_i < d_{i-1}$, `greedy_examination` forms a new anonymization group starting from v_i. If $j - i \geq k$, which means there are at least k vertices having the same degree, `greedy_examination` clusters $v_i, v_{i+1}, \ldots, v_{j-1}$ into the new group. n_a is set to be $j - i$. Otherwise, there are less than k vertices in the sequence $(v_i, v_{i+1}, \ldots, v_{j-1})$. Thereby `greedy_examination` clusters $v_i, v_{i+1}, \ldots, v_{i+k-1}$ in the new anonymization group. n_a is set to be k. However, there are also two exceptions when $n - i + 1 < 2k$ or $n - j + 1 < k$. The former means that $v_i, v_{i+1}, \ldots, v_n$ cannot form two anonymization groups. The latter means that $v_j, v_{j+1}, \ldots, v_n$ cannot be clustered into a separated group. In either exception, `greedy_examination` has to cluster $v_i, v_{i+1}, \ldots, v_n$ into an anonymization group. Then n_a is set to be $n - i + 1$.

The algorithm is described in Algorithm 1.

[1] If $i = 1$, the comparison is between d_1 with n.

[2] This is caused by `edge_creation`.

3.2 The `edge_creation` Algorithm

At each iteration, the input to `edge_creation` is a sequence of vertices $\boldsymbol{v}$ of length n sorted in the descending order of their degrees, an index i and a number n_a. The goal is to anonymize the vertices $v_i, v_{i+1}, \ldots, v_{i+n_a-1}$ to degree d_i by adding edges to the original graph. The output is an index, which equals $i + n_a$ if the anonymization succeeds, or equals j if v_j cannot be anonymized, where $i < j \leq i + n_a - 1$.

For each v_j in the vertex sequence $(v_i, v_{i+1}, \ldots, v_{i+n_a-1})$, `edge_creation` wires it to v_l for $j < l \leq n$, such that the edge (j, l) does not previously exist and $d_l < d_i$, until $d_j = d_i$. The former condition avoids creating multiple edges. The latter condition minimizes the need for reordering $\boldsymbol{v}$. If in the end `edge_creation` successfully anonymizes these n_a vertices, it reorders the new vertex sequence $\boldsymbol{v}$ in the descending order of their degrees. Otherwise, it returns the index j such that v_j cannot be anonymized with the wiring constraint. Then the repairing algorithm `relaxed_edge_creation` is invoked.

The algorithm is described in Algorithm 2.

Algorithm 2. The `edge_creation` algorithm

Input: $\boldsymbol{v}$: a sequence of n vertices sorted in the descending order of their degrees, i : an index, n_a : the number of vertices that are going to be anonymized starting from v_i.

Output: j : an index.

```
1 for $j \in (i+1, i+n_a-1)$ do
2     while $d_j < d_i$ do
3         Create an edge $(j,l)$ where $j < l \leq n$ such that $(j,l)$ does not previously
          exist and $d_l < d_i$.;
4         if The edge cannot be created then Return $j$;
5     end
6 end
7 Sort $\boldsymbol{v}$ in the descending order of degree;
8 Return $j$;
```

We consider three heuristics to examine the candidate vertices in $\boldsymbol{v}$ for the creation of edges.

The first heuristics examines $\boldsymbol{v}$ from v_{j+1} to v_n, that is, in the decreasing order of their degrees, and creates the edge (j, l) whenever the constraint is satisfied. The second heuristics examines $\boldsymbol{v}$ from v_n to v_{j+1}. The last heuristics randomly selects a candidature v_l and creates the edge (j, l). Below we denote by 1, 2, and 3, respectively, the variants of the complete algorithm with these three heuristics.

Intuitively, the first heuristics incurs larger anonymization cost than the second heuristics does. This is because the first heuristics increases the degree of vertices with large original degree, so that the largest degrees in the some anonymization groups might be increase. In order to anonymize these groups, more edges will be added. The third heuristics should behavior in between. On the other hand, the first two heuristics construct deterministic anonymized graphs whereas the third heuristics can generate random anonymized graphs, which, as we shall see, has consequences on the preservation of utility.

3.3 The `relaxed_edge_creation` Algorithm

The `edge_creation` algorithm is not certain to output a k-degree anonymous graph. The failure occurs when an edge (j, l) with the wiring constraint cannot be created for some j. In this case, `relaxed_edge_creation` is invoked. It relaxes the wiring constraint.

The algorithm examines $\boldsymbol{v}$ from v_n to v_1 and iteratively creates an edge (j, l) if only the edge does not previously exist, until $d_j = d_i$. Then `relaxed_edge_creation` returns the index l. Notice that this iteration can always stop because in the worst case v_j will be wired to all the other vertices. Finally `relaxed_edge_creation` sorts the new vertex sequence $\boldsymbol{v}$ in the descending order of degree and feeds it as the input of `greedy_examination` in the next iteration.

The algorithm is described in Algorithm 3.

Algorithm 3. The `relaxed_edge_creation` algorithm

Input: $\boldsymbol{v}$: a sequence of n vertices sorted in the descending order of their degrees, i, j : two indices.
Output: l : an index.

```
1 for $l = n$ to 1 do
2     if $v_j$ and $v_l$ are not connected then
3         Create an edge $(j, l)$;
4         if $d_j = d_i$ then
5             Sort $\boldsymbol{v}$ in the descending order of degrees;
6             Return $l$;
7         end
8     end
9 end
```

Notice that this process may compromise the k-degree anonymity of the vertex sequence $(v_1, v_2, \ldots, v_{i-1})$ if the returned l is less than i, i.e., v_j is wired to some vertex that has been anonymized. In this case, `greedy_examination` needs to examine $\boldsymbol{v}$ from the beginning in the next iteration, i.e., i is set to be 0. In the other case where $l > i$, `greedy_examination` still examines $\boldsymbol{v}$ starting from v_i in the next iteration. However, as `relaxed_edge_creation` examines $\boldsymbol{v}$ from small degree to large degree, there is a high probability that $(v_1, v_2, \ldots, v_{i-1})$ is still k-anonymous.

3.4 The Fast K-degree Anonymization Algorithm

The FKDA algorithm combines the `greedy_examination`, `edge_creation` and `relaxed_edge_creation` algorithms. The input to FKDA is a graph G with n vertices and the value of k. The output is a k-degree anonymous graph G'.

FKDA first computes the vertex sequence $\boldsymbol{v}$ of G in the descending order of degree. Then at each iteration, it invokes `greedy_examination` to compute the number n_a and passes it with i to `edge_creation`. If `edge_creation` successfully anonymizes the n_a vertices, FKDA updates the value of i as $i + n_a$. Then FKDA outputs the anonymized graph G' if $i > n$, or enters the next iteration otherwise. If `edge_creation` fails to construct the graph, FKDA invokes `relaxed_edge_creation` and updates the value of i according to the value of l returned by `relaxed_edge_creation`. Notice that FKDA can always output a valid k-degree anonymous graph because in the worst case a complete graph is constructed.

The complete algorithm is described in Algorithm 4.

Algorithm 4. The Fast K-Degree Anonymization algorithm

```
Input: $G$ : a graph of $n$ vertices, $k$ : the value of anonymity.
Output: $G'$ : a $k$-degree anonymous graph constructed from $G$.
1  $\boldsymbol{v}$=the vertex sequence of G in the descending order of degree;
2  $i = 1$;
3  while $i \leq n$ do
4      $n_a$ =greedy_examination($\boldsymbol{v}, i, k$);
5      $j$ =edge_creation($\boldsymbol{v}, i, n_a$);
6      if $j = i + n_a$ then
7          $i = i + n_a$;
8      else
9          $l$ =relaxed_edge_creation($\boldsymbol{v}, i, j$);
10         if $l < i$ then $i = 0$;
11     end
12 end
13 Return $G'$;
```

We provide the approximate bounds of the edit distance to the original graph produced by FKDA. Suppose ideally the original vertex sequence $\boldsymbol{v}$ is clustered as follows. The sequence $(v_1, v_2, \ldots, v_{ik})$ is clustered into i groups, each of which contains k vertices, i.e., the $(j+1)^{th}$ group contains the vertices $v_{jk+1}, v_{jk+2}, \ldots, v_{(j+1)k}$, $0 \leq j \leq i-1$. The sequence $(v_{ik+1}, v_{ik+2}, \ldots, v_n)$ is already k-anonymous[3]. In the best case (which is encountered in the second heuristics of `edge_creation`), the vertices in the sequence $(v_1, v_2, \ldots, v_{ik})$ are only wired to the vertices in the sequence $(v_{ik+1}, v_{ik+2}, \ldots, v_n)$ by `edge_creation`. Suppose the latter sequence is still k-anonymous after anonymization. Then we get the lower bound which is $bound_l = \sum_{j=0}^{i-1} \sum_{l=1}^{k} (d_{jk+1} - d_{jk+l})$. In the worst case (which is encountered in the first heuristics of `edge_creation`), each vertex in the sequence $(v_1, v_2, \ldots, v_{ik})$ is wired to all of its antecedent vertices. Then the largest degree of the $(j+1)^{th}$ group becomes $d_{jk+1} + jk$. Therefore the upper bound is $bound_u = \sum_{j=0}^{i-1} \sum_{l=1}^{k} (d_{jk+1} + jk - d_{jk+l}) = \frac{i \times (i-1)}{2} k^2 \times bound_l$.

[3] This is the usual case for large graphs.

4 Performance Evaluation

4.1 Experimental Setup

We implement KDA and three variants of FKDA, FKDA 1, FKDA 2 and FKDA 3, corresponding to the three heuristics in C++. We run all the experiments on a cluster of 54 nodes, each of which has a 2.4GHz 16-core CPU and 24 GB memory.

4.2 Datasets

We use three datasets, namely, **Email-Urv**, **Wiki-Vote** and **Email-Enron**.

Email-urv contains the email communication among faculty and graduate students at Rovira i Virgili University of Tarragona, Spain. It is an undirected graph with 1133 vertices and 10902 edges. Each vertex represents an email address. An edge connects a pair of vertices if there is at least one email communication between the corresponding email users. The dataset is available at http://deim.urv.cat/ aarenas/data/welcome.htm.

Wiki-Vote is the network of votes for the administrator election for Wikipedia pages. It is a directed graph with 7115 vertices and 103689 edges. Each vertex represents a user of Wikipedia. An edge links vertex i to vertex j if user i votes on user j. The dataset is available at http://snap.stanford.edu/data/wiki-Vote.html.

Email-Enron has been described in Section 1.1.

We conducts experiments on these three graphs. The different sizes of the three graphs illustrate the performance of KDA and FKDA on small (1133 vertices), medium (7115 vertices) and relatively large (36692 vertices) graphs.

4.3 Effectiveness Evaluation

We compare the effectiveness of the algorithms by evaluating the variation of several utility metrics: edit distance (ED), clustering coefficient (CC) and average shortest path length (ASPL) (following [13]).

We calculate the edit distance between the anonymized graph and the original graph. The edit distance is the number of edges Δm added to the original graph. For the sake of convenience for comparison, we normalize it to the number of edges in the original graph and calculate $\Delta m/m$.

We also calculate the clustering coefficient. The clustering coefficient of a vertex is defined as the fraction of the edges existing between the neighbors of the vertex. Then the clustering coefficient of a graph is defined as the average clustering coefficient of all the vertices.

We also calculate the average shortest path length. The shortest path length between a pair of vertices in a simple graph is defined as the number of hops from one vertex to the other. Then the average shortest path length of a graph is defined as the average of the shortest path lengths between all reachable pairs of vertices.

We vary the value of k in the range $\{5, 10, 15, 20, 25, 50, 100\}$. For each value of k, we run each algorithm 10 times on each dataset and compute the average value of the metrics.

Figure 1-3, 4-6 and 7-9 show the results on Email-Urv, Wiki-Vote and Email-Enron, respectively.

Figure 1, 4 and 7 show the evaluation results of the normalized edit distance on the three graphs.

We see that FKDA adds more edges to Email-Urv but less edges to Wiki-Vote and Email-Enron compared to KDA. In Email-Urv which is a small graph with 1133 vertices, the differences between large degrees are not large. By using KDA the residual degrees of the anonymized vertices with large original degrees can be compensated by enough number of anonymized vertices with residual degrees, that is, the anonymized degree sequence is realizable, with only a small number of repetition of `Probing`. Thus the minimum edit distance found by dynamic programming is still less than the edit distance produced by FKDA. To the contrary, Wiki-Vote and Email-Enron are two relatively larger graphs with 7115 and 36692 vertices, respectively. The differences betweens large degrees of either graph are considerably large. Therefore by using KDA, `Probing` is invoked a significant number of times before a k-degree anonymous graph is constructed, as explained in Section 1.1. Moreover, by comparing our `relaxed_edge_creation` algorithm with `Probing`, we find that `relaxed_edge_creation` increases a small degree only if the corresponding vertex can be wired to an anonymized vertex with residual degree. To the contrary, `Probing` randomly increases a small degree regardless the actual structure of the graph. The corresponding vertex may not be able to be wired to an anonymized vertex with residual degree (There might exist already an edge between the two vertices.). Consequently, more repetitions of `Probing` are invoked. Thus we believe that eventually `Probing` adds more noise than `relaxed_edge_creation` does to the degree sequences of the two large graphs. Therefore FKDA adds less edges than KDA does to the two graphs.

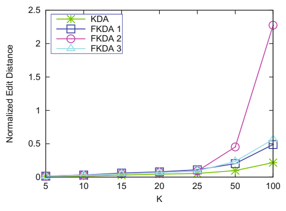

Fig. 1. ED: Email-Urv

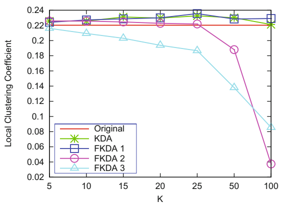

Fig. 2. CC: Email-Urv

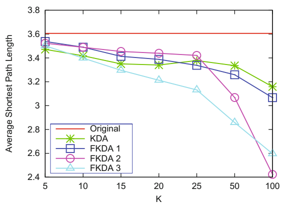

Fig. 3. ASPL: Email-Urv

Figure 2, 5, 8 and Figure 3, 6, 9 show the evaluation results of clustering coefficient and average shortest path length, respectively. The constant line shows the value of corresponding metric in the original graph.

We see that FKDA produces less similar results with that in the original graphs on Email-Urv and more similar results on Wiki-Vote and Email-Enron than KDA does. This is generally consistent with the evaluation results of edit distance, since FKDA adds more edges to Email-Urv and less edges to Wiki-Vote and Email-Enron than KDA does.

We further compare the performances of the three variants of FKDA.

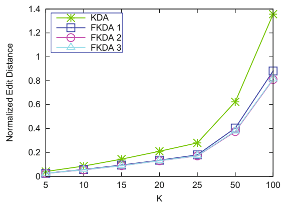

Fig. 4. ED: Wiki-Vote

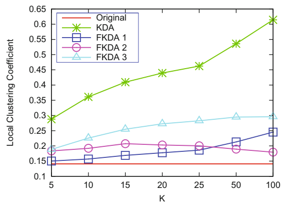

Fig. 5. CC: Wiki-Vote

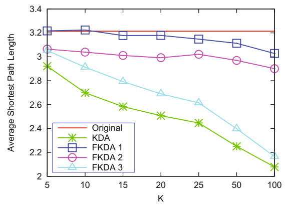

Fig. 6. ASPL: Wiki-Vote

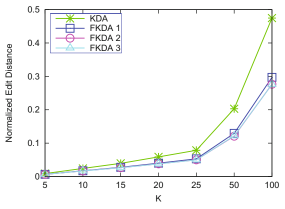

Fig. 7. ED: Email-Enron

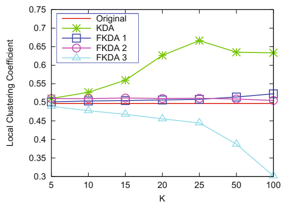

Fig. 8. CC: Email-Enron

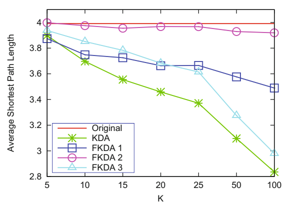

Fig. 9. ASPL: Email-Enron

In Section 3.2 we say the that first heuristics incurs larger anonymization cost, i.e. edit distance, than the second heuristics does, and the third heuristics performs in between. The results in Figure 4 and 7 support this claim, although the differences are small. However, in the small graph Email-Urv, we observe that FKDA 2 incurs much larger edit distance than the other two variants and FKDA 1 incurs the smallest edit distance, for $k = 50$ and $k = 100$. The reason is as follow. When k increases, after anonymization the residual degrees of the vertices with large original degrees become larger. Therefore more residual vertices with smaller original degrees are required to compensate these large residual degrees. As FKDA 2 creates edges by wiring the anonymized vertices to the vertices from with small degree to large degree, it makes the degrees of the anonymized vertices and the degrees of the subsequent vertices closer to each other than FKDA 1 does. Because of the wiring constraint in `edge_creation`, at some point there are no enough residual vertices to compensate the residual degree of a anonymized vertex. Then `relaxed_edge_creation` is invoked. When k is too large for the number of vertices (for example, $k = 50, 100$ and $n = 1133$ in Email-Urv), `relaxed_edge_creation` is invoked several times by FKDA 2. Then the edit distance to the original graph is enlarged. To the contrary, FKDA 1 creates edges by wiring the anonymized vertex with large residual degree to the vertices from with large degree to small degree. It maintains a sufficient gap between the degrees of the anonymized vertices and the degrees of the subsequent vertices. The residual degree of the anonymized vertices can be compensated under the wiring constraint in `edge_creation`, without invoking `relaxed_edge_creation`. Therefore the edit distance is small. FKDA 3 creates edges by wiring the anonymized vertices to random residual vertices, so that it incurs the edit distance to the original graph in between.

The abilities of the three heuristics on the preservation of utility of the original graph differ from each other, depending on the structure of the original graph.

For example, Figure 6 shows that FKDA 1 incurs larger average shortest path length in the anonymized Wiki-Vote than FKDA 2 does. This suggests that the vertices in Wiki-Vote with similar degrees are more connected than the vertices with very different degrees. So creating edges by wiring an anonymized vertex to the vertices from with large degree to small degree (similar degree to different degree) in `edge_creation` of FKDA 1 does not reduce the average shortest path length much. To the contrary, FKDA 2 links vertices with very different degrees in `edge_creation`, which results in a significant reduction in the average shortest path length. However, Figure 9 shows the reverse result in the anonymized Email-Enron, which suggests that the vertices in Email-Enron with similar degrees are less connected than the vertices with very different degrees. The overall results show that FKDA 1 and FKDA 2 preserve the utilities of the original graph better than FKDA 3 does. Nevertheless, FKDA 3 has an interesting property that it can generate a random k-degree anonymous graph.

4.4 Efficiency Evaluation

We compare the efficiency of the algorithms by measuring their execution time.

We vary the value of k in the range $\{5, 10, 15, 20, 25, 50, 100\}$. For each value of k, we run each algorithm 10 times on each dataset and compute the average execution time. We also compute the speedup of FKDA versus KDA for each parameter setting.

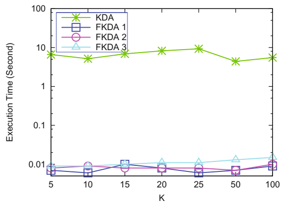

Fig. 10. Execution time on Email-Urv

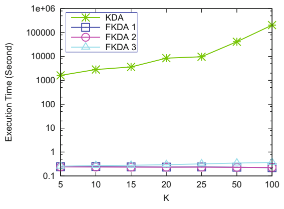

Fig. 11. Execution time on Wiki-Vote

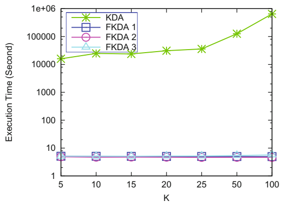

Fig. 12. Execution time on Email-Enron

Figure 10, 11 and 12 show the execution times on Email-Urv, Wiki-Vote and Email-Enron, respectively. Figure 13, 14 and 15 show the corresponding speedups.

We see that FKDA is significantly more efficient than KDA. The speedup varies from hundreds to one million on different graphs. The inefficiency of KDA is due to the decoupling of the checking of realizability of the anonymized degree sequences from the construction of graph.

The efficiency of the three FKDA variants is similar. FKDA 1 and FKDA 2 are slightly faster than FKDA 3. This is because FKDA 3 maintains additional a list of candidate residual vertices in `edge_creation`.

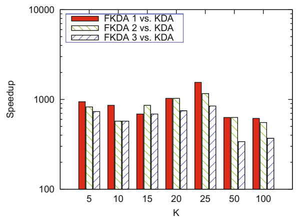

Fig. 13. Speedup of FKDA vs. KDA on Email-Urv

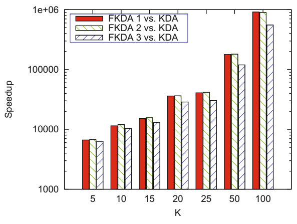

Fig. 14. Speedup of FKDA vs. KDA on Wiki-Vote

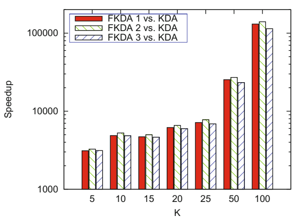

Fig. 15. Speedup of FKDA vs. KDA on Email-Enron

5 Conclusion

In this paper, we propose a greedy k-degree anonymization algorithm that anonymizes a graph by simultaneously adding edges and anonymizing its degree sequence in groups of vertices.

The algorithm is designed to overcome the shortcomings of the KDA algorithm proposed by [13]. The simultaneity of degree anonymization and graph construction in the new FKDA algorithm eliminates the need for realizability testing, which, as confirmed by our experiments, is a significant factor in the poor efficiency of the KDA algorithm.

We propose three variants of the algorithm, corresponding to three wiring heuristics. The comparative empirical performance evaluation on three real world graphs shows that the three variants of FKDA are significantly more efficient than KDA and more effective than KDA on large graphs.

We do not claim that our solution is a panacea for the anonymization of graphs in general, that objective being anyway a chimerical target given the generality of background knowledge potentially available to adversaries. It is however a very effective and efficient solution for the protection of privacy in the presence of background knowledge about vertex degrees. More importantly our solution shows that it is possible to tightly knit realizability and construction into one anonymization process and therefore paves the way to the development of algorithms catering for a variety of background structural knowledge.

References

1. Backstrom, L., Dwork, C., Kleinberg, J.M.: Wherefore art thou R3579X?: Anonymized social networks, hidden patterns, and structural steganography. Commun. ACM 54(12) (2011)
2. Barabási, A.-L., Albert, R.: Emergence of Scaling in Random Networks. Science 286, 509–512 (1999)
3. Bhagat, S., Cormode, G., Krishnamurthy, B., Srivastava, D.: Class-based graph anonymization for social network data. PVLDB 2(1) (2009)
4. Campan, A., Truta, T.M.: A clustering approach for data and structural anonymity in social networks. In: PinKDD (2008)

5. Cheng, J., Fu, A.W.-C., Liu, J.: K-isomorphism: privacy-preserving network publication against structural attacks. In: SIGMOD (2010)
6. Clauset, A., Shalizi, C.R., Newman, M.E.J.: Power-law distributions in empirical data. SIAM Reviews (2007)
7. Cormode, G., Srivastava, D., Yu, T., Zhang, Q.: Anonymizing bipartite graph data using safe groupings. PVLDB 19(1) (2010)
8. Francesco Bonchi, A.G., Tassa, T.: Identity obfuscation in graphs through the information theoretic lens. In: ICDE (2011)
9. Ghinita, G., Karras, P., Kalnis, P., Mamoulis, N.: Fast data anonymization with low information loss. In: VLDB, pp. 758–769 (2007)
10. Hay, M., Miklau, G., Jensen, D., Towsley, D., Weis, P.: Resisting structural re-identification in anonymized social networks. PVLDB 1(1), 102–114 (2008)
11. Korolova, A., Motwani, R., Nabar, S.U., Xu, Y.: Link privacy in social networks. In: CIKM (2008)
12. Li, Y., Shen, H.: Anonymizing graphs against weight-based attacks. In: ICDM Workshops (2010)
13. Liu, K., Terzi, E.: Towards identity anonymization on graphs. In: SIGMOD Conference, pp. 93–106 (2008)
14. Liu, L., Wang, J., Liu, J., Zhang, J.: Privacy preserving in social networks against sensitive edge disclosure. In: SIAM International Conference on Data Mining (2009)
15. Song, Y., Nobari, S., Lu, X., Karras, P., Bressan, S.: On the privacy and utility of anonymized social networks. In: iiWAS (2011)
16. Sweeney, L.: K-anonymity: a model for protecting privacy. International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems 10(5) (2002)
17. Tai, C.-H., Yu, P.S., Yang, D.-N., Chen, M.-S.: Privacy-preserving social network publication against friendship attacks. In: SIGKDD (2011)
18. Wu, W., Xiao, Y., Wang, W., He, Z., Wang, Z.: K-symmetry model for identity anonymization in social networks. In: EDBT (2010)
19. Ying, X., Wu, X.: Randomizing social networks: a spectrum perserving approach. In: SDM (2008)
20. Yuan, M., Chen, L., Yu, P.S.: Personalized privacy protection in social networks. PVLDB 4(2) (2010)
21. Zheleva, E., Getoor, L.: Preserving the Privacy of Sensitive Relationships in Graph Data. In: Bonchi, F., Malin, B., Saygın, Y. (eds.) PInKDD 2007. LNCS, vol. 4890, pp. 153–171. Springer, Heidelberg (2008)
22. Zhou, B., Pei, J.: Preserving privacy in social networks against neighborhood attacks. In: ICDE (2008)
23. Zou, L., Chen, L., Özsu, M.T.: K-automorphism: a general framework for privacy-preserving network publication. PVLDB 2(1) (2009)