On Anonymization of Multi-graphs

Chun Li* Charu C. Aggarwal† Jianyong Wang‡

Abstract

The problem of privacy-preserving data mining has attracted considerable attention in recent years because of increasing concerns about the privacy of the underlying data. In recent years, an important data domain which has emerged is that of graphs and structured data. Many data sets such as XML data, transportation networks, traffic in IP networks, social networks and hierarchically structured data are naturally represented as graphs. Existing work on graph privacy has focussed on the problem of anonymizing nodes or edges of a single graph, in which the identity is assumed to be associated with individual nodes. In this paper, we examine the more complex case, where we have a collection of graphs, and the identity is associated with individual graphs rather than nodes or edges. In such cases, the problem of identity anonymization is extremely difficult, since we need to not only anonymize the labels on the nodes, but also the underlying global structural information. In such cases, both the global and local structural information can be a challenge to the anonymization process, since any combination of such information can be used in order to de-identify the underlying graphs. In order to achieve this goal, we will create synthesized representations of the underlying graphs based on aggregate structural analytics of the collection of graphs. The synthesized graphs retain the properties of the original data while satisfying the *k*-anonymity requirement. Our experimental results show that the synthesized graphs maintain a high level of structural information and compatible classification accuracies with the original data.

Keywords: Graphs, privacy, anonymization

1 Introduction

The problem of privacy-preserving data mining has gained importance in recent years because of increasing concerns about compromising the safety and security of private information. Consequently, considerable research has been devoted in recent years to the problem of privacy in a variety of data domains [1, 6, 11, 15, 16, 18, 19]. Detailed surveys on privacy may be found in [3].

A key method in privacy-preserving data mining is that of *k*-anonymity. In the *k*-anonymity method, we transform the data such that each record is indistinguishable from at least *k* other records in the data set. Most *k*-anonymization work is focussed on continuous and categorical data domains [1, 15, 16, 19]. In this paper, we will discuss the problem of privacy-preservation of graphs and other kinds of structured data. Many important forms of data such as XML, semi-structured data, and web data are naturally expressed in structured form. Recently, some studies focus on the anonymization of graphs, such as [10, 17, 24, 22, 23]. Those methods use techniques such as duplicating, removing, switching or grouping vertices or edges on input graphs. This is because these techniques are focussed on anonymization of nodes or edges of *individual graphs*, and the identity is associated with portions (i.e. nodes or edges) of a *single graph*. The focus of anonymization is to apply transformations to this graph such that the privacy of the nodes or edges is maintained. However, in many applications such as semi-structured data, we need to anonymize a *collection of* graphs rather than a single graph. In such cases, the identity is associated with entire graphs rather than portions of it. The problem of anonymization and de-identification of collections of graphs has not been discussed in past work. Such cases arise in the following kinds of applications: **(1)** The XML representation of attributes about individuals is in the form of a graph representation. The structure of this graph will vary with the individual. **(2)** The click-graph derived from a proxy-log in a user-session is typically a sparse graph on the underlying web graph. Each click graph is associated with a particular user. **(3)** Many GPS based mobile devices in vehicle control systems collect the route of a person traveled in a given vehicle. In general, location-based services are susceptible to privacy attacks [3]. The route for a given individual in a particular time-period can be expressed as a single graph. Such data is useful for a variety of applications (eg. traffic prediction), but it is too sensitive for mining because of privacy concerns.

We note that anonymization of entire graphs is an extremely difficult problem, because information about small portions of the graph can expose the identity of the entire graph. In general, if background knowledge is available about a small portion of the structure of the graph for particular individuals, then it can be used in order to mount a privacy-attack. Furthermore, methods such as *k*-anonymization are typically dependent upon some notion of proximity. We note that the notion of proximity is very difficult to define in the case of structured data. This is because

*Tsinghua University, socrates.lee@gmail.com
†IBM T.J. Watson Research Center, charu@us.ibm.com
‡Tsinghua University, jianyong@tsinghua.edu.cn

structural similarity can often manifest itself in the form of an isomorphism which is a computationally difficult problem to begin with. Since the pairwise isomorphism problem is extremely difficult, the problem of partitioning the data into sets of similar structures (with cardinality at least k) is even more challenging.

We will perform the anonymization of the underlying graphs with the use of a structural similarity approach. The key is to create groups of similar graphs which share common structural properties. We utilize these groups in order to construct a set of super-structures. Each super-structure represents the *representative structural properties* of the group. Thus, this is a *condensed representation* of the group. These representative structural properties are used in order to generate synthetic representations of the underlying structures. These synthetic representations retain the aggregate properties of the underlying data, and can therefore be used for a variety of database and data mining applications. In this paper, we will show the utility of this technique for a classification application. We will show that this approach results in an extremely effective representation of the underlying data without losing privacy.

This paper is organized as follows. In the remainder of this section, we will discuss related work. In section 2, we introduce some preliminaries and the concept of mapping which is a critical concept used for developing the framework for the condensation approach. In section 3, we will describe our algorithms for anonymization. In section 4, we will present the experimental results of our approach. In section 5, we will present the conclusions and summary.

1.1 Related Work The problem of privacy-preserving data mining has been studied extensively in recent years [3] because of its numerous applications to a wide variety of problems in the literature. One of the earliest methods for privacy-preserving data mining was that of perturbation [6, 7]. In this method, we add noise to the data set in order to mask the sensitive attributes of the underlying data.

While the perturbation method is able to mask sensitive values, it is not designed for de-identification. For this purpose, group-based anonymization [18, 19] is particularly useful. A recent method to avoid the problem of *de-identification* of data records is that of *k-anonymization*. The idea in k-anonymization is to reduce the granularity of representation of the underlying data, so that it is impossible to use pseudo-identifiers in order to determine the ownership of the underlying records [19]. The broad concept of group-based data anonymization has been studied extensively [1, 8, 12, 15, 16, 18] in the context of multi-dimensional data. Recently, the problem of graph anonymization [10, 13, 17, 22, 24] has found increasing interest in the literature. However, most of these techniques are focussed on the problem of *node and edge identity anonymization* in single graphs. In many applications, such as semi-structured data, identities are associated with individual graphs rather than nodes, and the anonymization needs to be performed over a *collection of* graphs. This paper will explore this difficult case, which has not been handled in previous work on graph anonymization. Graphs pose a special challenge to the anonymization process because of the underlying structural information, which is hard to aggregate and use across multiple graphs with very different structures. Furthermore, graph data is known to be inherently high dimensional, a problem which makes it particularly resistant to methods such as k-anonymization [3].

A recent method for anonymization is the method of condensation [1]. In this technique, we construct *newly generated* anonymized records which preserve the aggregate properties of the original data. This allows the use of the records for classification. The use of condensation techniques is particularly resistant to adversarial attacks, because the individual records are freshly generated. However, the technique in [1] is designed only for the case of multi-dimensional data. In this paper, we will design techniques for generation of representative graphs from the underlying data.

2 Mapping Techniques for Condensation

In this section, we will first introduce some notations and definitions. Then, we will describe the concept of graph mapping which is used for graph condensation. We assume that the data set $\mathcal{D}$ consists of a set of N graph structures which are denoted by $G_1 \ldots G_N$. Each graph G_i is denoted by the notation (V_i, E_i), where V_i is the set of nodes and E_i is the set of edges between nodes. Each node v contains a label which identifies the properties of that node, denoted by $label(v)$. For example, in an XML document collection, the label could correspond to the attribute-value combination. The set of edges represent the structural relationships between nodes.

We note that our condensation-based technique for anonymization uses similarity-mapping between nodes in order to construct templates for anonymization. Such similarity mappings help in understanding the relationships between nodes of approximately similar graphs. They are also necessary for generating a structural representation of the aggregate behavior of a group of graphs. The most strict notion of similarity is that of an isomorphism between two graphs of equal size. The concept of isomorphism is defined as follows:

DEFINITION 1. *(Graph Isomorphism) Let the graphs G_1 and G_2 have vertex sets V_1 and V_2 respectively. Graph G_1 is isomorphic to graph G_2, if there exists a bijection ϕ such that for any vertex $v \in V_1$, we have $\phi(v) \in V_2$, $label(v) = label(\phi(v))$, and for any edge $e = (v1, v2) \in E_1$, we have*

$\phi(e) = (\phi(v1), \phi(v2)) \in E_2$.

In practice, two graphs in a data set are rarely of the same size, and may also have considerable diversity in the labels of the nodes. Therefore, it is necessary to define a *similarity mapping* between nodes. The similarity value between two vertices is defined in terms of a similarity function of the labels of the vertices in the two graphs. The simplest similarity function is one in which two vertices have a similarity value of 1, when the labels match and 0 otherwise. A graph isomorphism is a case of exact mapping between two graphs of equal size. The concept of a similarity mapping is a relaxation of this exact approach, and can also be used effectively for approximately matching graphs of unequal size, or where one-to-one mappings do not exist in the original graphs. In such cases, we can augment each graph with dummy vertices and dummy edges such that every vertex or edge has a corresponding element in the other graph. The extended graph is denoted by $G^*(V^*, E^*)$. All the dummy vertices and edges are specially labeled ε. Dummy vertices have greater flexibility in mapping onto any other vertex. With the help of extended graphs, the mapping between two unequal or partially mapped graphs is defined as follows:

DEFINITION 2. *(Graph Mapping) A mapping between two graphs G_1 and G_2 is a bijection ϕ: $G_1^* \rightarrow G_2^*$, with corresponding vertex sets V_1^* and V_2^* where the following are satisfied:*

- *$\forall v \in V_1^*$, we have $\phi(v) \in V_2^*$, and at least one of them is not a dummy vertex.*
- *$\forall e = (v_1, v_2) \in E_1^*$, we have $\phi(e) = (\phi(v_1), \phi(v_2)) \in E_2^*$, and at least one of them is not a dummy edge.*

2.1 Graph Closures and the Mapping Algorithm In this section, we will define the concept of *graph closure*, which is essential in order to implement the condensation approach. The concept provides a correspondence between the vertices of distinct graphs and is used both for condensed super-structure creation, and for generation of the pseudo-graphs from the condensed super-structure. We will also provide a graph mapping algorithm which is required in order to construct the closures. For graph mapping, we will use a neighbor biased mapping (NBM) technique, which iteratively attempts to use the structural behavior of the neighbors of a partially mapped graph in order to further augment the mapping. The similarity mapping between the different vertices is used in order to relate the aggregate behavior of different graphs in a way which preserves the underlying structural behavior.

The aggregate behavior of a group of graphs is defined in terms of *closures* of various structural properties of the graph. Therefore, we will first define the concept of graph closure. As discussed earlier, the condensation approach attempts to aggregate the behavior of multiple graphs in order to create an effective template for generation of the underlying pseudo-data. Such structural aggregation concepts are handled by the concept of closure: In order to properly represent aggregate properties, it becomes necessary to associate weights with edges and vertices. All nodes and edges in the original graph are assigned a weight of one. However, dummy nodes and edges are assigned a weight of 0. Next, we define the concept of vertex and edge closure.

DEFINITION 3. *(Vertex Closure and Edge Closure) The closure of a set of vertices is a generalized vertex v', with a label which is expressed as a set of the union of the labels of the underlying vertices. Each node label is assigned a weight which is equal to the sum of the weights of the underlying vertex set. The closure of a set of edges is a generalized edge e', with a weight which is the sum of the weights of the underlying edge set.*

Next, we will define the concept of graph closure of two graphs G_1 and G_2 of the same size under a one-to-one vertex mapping ϕ. In the event that the two graphs are not of the same size, or only a partial mapping exists, we can add dummy vertices and edges as discussed in the last section in order to create a one-to-one mapping. Therefore, the assumption of a one-to-one mapping does not lose generality. Let us assume that we have a pair of a graphs such that a one-to-one mapping ϕ exists between the corresponding pair of vertices. In other words, for any vertex $v \in G_1$ we have a mapped vertex $\phi(v) \in G_2$. Note that a one-to-one mapping between nodes also defines an implicit one-to-one mapping between edges. In cases, where an edge exists in one graph but does not exist in the other, we can assume a dummy edge in the latter with zero weight. Therefore, ϕ defines a complete one-to-one mapping between both edges and vertices. We can use ϕ in order to define the graph closure between two graphs. Conceptually, this closure represents a "structural merging" of the two graphs with the use of this mapping.

DEFINITION 4. *(Graph closure of two graphs under ϕ) The closure of two graphs G_1 and G_2 under a mapping ϕ is a generalized graph $(\mathbb{V}, \mathbb{E})$ where $\mathbb{V}$ has the same number of vertices as the constituent graphs. This new vertex set contains the pairwise closure of the corresponding vertices in the two graphs under the mapping ϕ. Similarly, $\mathbb{E}$ contains the pairwise closure of the corresponding edges in the two graphs. The resulting graph is denoted by $\mathbb{C} = closure(G_1, G_2)$.*

Note that the above definition of graph closure is defined in terms of the mapping ϕ between the vertices. Since this mapping will be defined on the basis of structural similarity, the concept of closure defines a structural merging

which retains the key structural aspects in the underlying graphs. We will discuss slightly later how this mapping ϕ is determined in terms of a structural similarity-based technique. Similar to the mapping between two graphs, we can define the mapping between a graph closure and a graph.

DEFINITION 5. *(Graph mapping between a graph closure and a graph) A mapping between a graph closure $\mathbb{C}$ and a graph G is a bijection ϕ': $\mathbb{C}^* \rightarrow G^*$, where the following are satisfied:*

- *$\forall v' \in \mathbb{V}^*$, we have $\phi'(v) \in V^*$, and at least one of them is not a dummy vertex closure or a dummy vertex;*
- *$\forall e = (v'_1, v'_2) \in \mathbb{E}^*$, we have $\phi(e) = (\phi'(v'_1), \phi'(v'_2)) \in E^*$, and at least one of them is not a dummy edge closure or a dummy edge.*

So far, we have described the closure of two graphs. We can also recursively define the closure between a graph closure and a graph. This is needed for successive merging of multiple graphs into a single closure. As before, we can assume (without loss of generality) that both the two structures are of the same size, and a one-to-one mapping exists between the vertices and edges of the two graphs.

DEFINITION 6. *(Graph closure between a graph closure and a graph under ϕ') The closure between a graph closure $\mathbb{C}$ and a graph G under a mapping ϕ' is a generalized graph $(\mathbb{V}', \mathbb{E}')$ whose vertex set $\mathbb{V}'$ is of the same size as the two underlying structures. Here $\mathbb{V}'$ is the pairwise closure of the corresponding nodes in the two structures under ϕ'. Similarly, $\mathbb{E}'$ is the pairwise closure of the corresponding edges in the constituent structures. The resulting closure of the two structures is denoted by $\mathbb{C}' = closure(\mathbb{C}, G)$.*

With the concept of the graph closure between a graph closure and a graph, we can recursively merge a group of graphs into a single graph closure. Thus, the final graph closure results in a substructure-preserved graph mapping. Next, we will describe the NBM method, which is used to construct the one-to-one mapping ϕ between graphs (graph closures). In the NBM algorithm discussed below, we will output a *matchedlist* which provides an approximate mapping of vertices between G_1 and G_2. The *min_threshold* in the input is a constraint that eliminates unqualified pairs of vertices, if they are not matched sufficiently. First, we calculate the similarity for each pair of vertices (u, v), which $u \in G_1, v \in G_2$, and store them in similarity matrix W, which contains an entry for each pair of vertices. Note that the similarity between nodes is based on the underlying vertex labels. The exact similarity function varies with the method used, and therefore, we will specify it very generally at this stage. Correspondingly, the parameter *min_threshold* will also vary with the method used.

A priority queue PQ is used, which ranks the pairs according to a weight function associated with the pairs. This weight function starts off with the similarity values of the matched vertices, but is successively modified as more and more vertices are mapped to one another across the two graphs. The variables *mates* and *maxWeights* store the best matched vertex and the weight of $u \in G_1$ in $v \in G_2$. We dequeue the maximum weighted unmatched pair of vertices from the priority list *PQ* in each iteration and add it to the *matchedlist*. We also add weights to unmatched pairs containing the neighbors of the vertices of the pair. The exact nature of weight addition will depend upon the method used for condensation. The idea of adding such weights to the unmatched neighbor vertex pairs is to encourage successive matching of neighbor vertices, once a given vertex has been matched. This is the "neighbor bias" introduced by the method during the matching process. This approach results in structurally related vertices to be matched effectively in succession. The priority queue is used in order to continuously re-adjust the unmatched pairs based on these modified weights. At the end of the process, a mapping between pairs of vertices is output. The overall algorithm for performing the mapping is illustrated below:

ALGORITHM: **NBM**(G_1, G_2, *min_threshold*)

INPUT: (1) G_1: an input graph or graph closure, (2) G_2: an input graph or graph closure, (3) *min_threshold*: the minimal threshold of similarity between any matched pair.
OUTPUT: *matchedlist*: the list of matched pairs of G_1 and G_2, that each pair is of similarity at least *min_threshold*.

01: *Calculate the similarity matrix W of G_1 and G_2*
02: *Initialize mates with* **unmatched** *//best matches of $\forall u \in G_1$*
03: *Initialize maxWeights with* **-1** *//max weights of $\forall u \in G_1$*
04: *Initialize an empty priority queue PQ*
05: *for each $u \in G_1$*
06: *Find v_m such that $W_{u,v_m} = max\{W_{u,v} | v \in G_2\}$*
07: *if $v_m \geq min\_threshold$*
08: *PQ.Enqueue($W_{u,v_m}, \langle u, v_m \rangle$)*
09: *mates[u] := v_m*
10: *maxWeights[u] := W_{u,v_m}*
11: *set availCount := $|\{u | mates[u] ! =$* **unmatched**, *$u \in G_1\}|$*
12: *while !PQ.Empty() and matchedlist.Size() < availCount*
13: *$\langle u, v \rangle$ = PQ.Dequeue()*
14: *if matchedlist.Contains(u)*
15: *continue*
16: *if matchedlist.Contains(v)*
17: *Find v_m such that $W_{u,v_m} = max\{W_{u,v} | v \in G_2$, matchedlist.notContains(v) $\}$*
18: *if $W_{u,v_m} > min_t hreshold$*
19: *PQ.Enqueue($W_{u,v_m}, \langle u, v_m \rangle$)*
20: *mates[u] := v_m*

21: $maxWeights[u] := W_{u,v_m}$
22: *continue*
23: *matchedlist.add(⟨u, v⟩)*
24: *Let N_u,N_v be neighbors of u,v.*
25: *for each $u' \in N_u$ where matchedlist.notContains(u'), mates[u'] !=* **unmatched**
26: *for each $v' \in N_v$ where matchedlist.notContains(v')*
27: *add weight to $W_{u',v'}$*
28: *if $W_{u',v'} > maxWeights[u']$*
29: *mates[u'] := v'*
30: *maxWeights[u'] := $W_{u',v'}$*
31: *if mates[u'] has changed*
32: *PQ.Enqueue($W_{u',v'}$, ⟨u', v'⟩)*
33: *return matchedlist.*

3 Structural Anonymization Approach

In this section, we will discuss the anonymization approach for graph data. Since our approach requires anonymization of complete graph collections, this requires us to determine the broad properties of the underlying graphs. We will define these broad properties in terms of the frequent structural behavior of the anonymized groups. Since each such group has size at least *k*, it follows that *k*-anonymity is guaranteed as well. The key accuracy issue is to ensure that the anonymized graphs preserve the aggregate properties of the input graphs. This is necessary to ensure that the anonymized data can be effectively used for data mining applications which are dependent upon aggregate characteristics. An example of such an application tested in this paper is that of classification.

We propose two methods for constructing the anonymized representations of the graph structures. Both methods contain three phases:

(1) In the first phase, we create constrained clusters of size at least *k* from the underlying data. The key in the first phase is to define clusters which retain the *broad structural patterns* in the underlying data and are less influenced by the random artifacts in the data.

(2) In the second phase, we utilize the groups in order to construct a super-structure to represent the *representative structural properties* of the group. This super-structure needs to retain the key statistical properties of the underlying group of graphs for the purpose of data generation. Thus, this is a *condensed representation* of the group.

(3) In the third phase, we use these representative structural properties in order to generate synthetic representations of the underlying structures. These synthetic representations retain the aggregate properties of the underlying data, and can therefore be used for a variety of database and data mining applications. In this paper, we will show the utility of this technique for a classification application.

The first and third phases of both methods are almost the same. The second phase of the two methods is however quite different:

(1) In the first method, the super template generated is a graph closure of a group of graphs. Therefore, we refer to this method as **ClosureG**. **(2)** In the second method, we construct a graph merged from the set of patterns with respect to a group of graphs. Therefore, we refer to this method as **PatternG**.

Next, we will discuss the details of these different steps.

3.1 Constructing the Anonymized Groups

If we generate synthesized graphs from the *global* statistical properties of the original graphs in $\mathcal{D}$, the synthetic graphs will not contain useful structural information for mining purposes. This is because of the diversity of the graphs in a given data set. On the other hand, if we generate a synthesized graph with the statistics of a small number of relatively similar graphs, the quality of the generated data will improve, but the anonymity will not meet our demands. This is the natural trade-off between privacy and accuracy, which is encountered in many applications.

Therefore, we create groups of graphs which share similar substructures to form an anonymized group, and generate synthetic graphs according to the graphs in the same group. The size of this group *k* defines the anonymity level of the data set. Since the output graphs are defined only in terms of the statistical characteristics of this group, it can be safely said that the group satisfies the *k*-anonymity requirement. Clearly, the value of *k* defines the tradeoff between privacy and accuracy.

We construct the anonymized groups of graphs in $\mathcal{D}$ by designing a constrained partitioning algorithm based on frequent substructures. Each anonymized group is a cluster in this partitioning, and the constraint on the partitioning is that each cluster contains at least *k* graphs. This constraint on the cluster cardinality arises from the corresponding constraint on the anonymity requirement. This additional constraint requires some care in terms of the design of the underlying algorithm. A key step in this process is the determination of the frequent subgraphs in the different clusters. We will see that these frequent subgraphs are very useful in comparing the similarity behavior across different clusters. Therefore, it is helpful to explore the nature of the frequent pattern mining algorithm which will be useful for our purposes.

Frequent patterns are defined in terms of the number of graphs for which a particular pattern is a subgraph. The number of such patterns is referred to as the support. Given a minimal support threshold *min_sup*, a frequent subgraph pattern is a subgraph whose support is no less than a user specified threshold *min_sup*. The problem of mining frequent patterns has been studied extensively in the graph mining literature [4, 5]. In this paper, we will use the technique of

closed pattern mining in order to perform the anonymization. A frequent subgraph is closed if there is no supergraph of that pattern with the same support. The set of (closed) frequent subgraphs with respect to the minimal support threshold contained in a graph G_i is called the pattern set of graph G_i. This pattern set is denoted by P_i, and the cardinality of P_i is denoted by $|P_i|$. The advantage of using closed pattern mining is that it generates a compact (yet complete) set of patterns for anonymization purposes. We used a very efficient algorithm called CloseGraph[21] in order to mine the frequent patterns for our algorithm. The algorithm outputs the graph structure of each pattern and the graph set which contains the pattern. The pattern set of a graph comprises all the frequent substructures contained in it. Correspondingly, the pattern set of a cluster is the union of all pattern sets of graphs contained in the cluster. We adopt the Jaccard distance *on the underlying pattern behavior* in order to calculate the distance between two graphs or two clusters. Consider two graphs G_i and G_j, which contain the pattern sets P_i and P_j. Then, the *pattern-based Jaccard distance* between them is defined as follows:

$$d_{ij} = 1 - \frac{|P_i \cap P_j|}{|P_i \cup P_j|} \tag{3.1}$$

We note that the use of a pattern-based similarity function is much more robust than a technique which simply utilizes matching on individual edges. This is because pattern-based distances provide much higher similarity to pairs of graphs containing large sub-structures in common. This helps in increasing the coherence of the distance function, and removing the noise which arises as a result of random artifacts in the underlying data.

We can easily extend our similarity function to the case of two clusters of graphs (rather than two graphs). The main difference is that we use the average pair-wise distances between the constituent graphs rather than the clusters themselves. Thus, for two clusters C_a and C_b containing $|C_a|$ and $|C_b|$ graphs, the overall distance $D(C_a, C_b)$ is defined as follows:

$$D(C_a, C_b) = \frac{\sum_{G_i \in C_a, G_j \in C_b} d_{ij}}{|C_a| \cdot |C_b|} \tag{3.2}$$

We note that the cardinality constraint on the clusters makes it difficult to adapt most clustering schemes to this problem. In this scenario, a hierarchical clustering technique provides the greatest effectiveness and flexibility in terms of dealing with the cluster cardinality constraint.

We start off with N singleton clusters. We iteratively combine two clusters into a larger cluster, and continue the process until all clusters contain at least k graphs. From the point of view of the tradeoff between anonymity and accuracy, it is important to not have clusters which are too unequal in size. This is because the graphs generated from clusters containing too many graphs may not be of high quality. Therefore, we do not allow merging between pairs which are such that any of the clusters contain at least k graphs. For each iteration step, we will perform the following steps:

- We will compute the distances between each pair of clusters. The distance is computed using Equation 3.2 discussed above.
- We will find a pair of clusters with the minimum distance. As discussed earlier, we exclude those pairs for which the cardinality of any of the clusters is at least k.
- We will merge the two clusters into a larger cluster.

These iterative steps are continued until each of the groups has cardinality at least k, which is also the anonymity requirement. This approach creates groups of records, each of which are anonymized separately. The process of anonymization requires the creation of *super-templates* from these groups, a process which will be discussed shortly.

3.2 Construction of Super-Template from Condensed Groups In this section, we will discuss the construction of the super-template from the condensed anonymized groups. We will construct a super-template preserving the most structural information of the graphs in an anonymized group. This super-template could then be used in order to generate the anonymized graphs.

In order to facilitate our discussion, we will introduce some notations. Let us consider an anonymized group containing the graphs $G_1 = (V_1, E_1)$, $G_2 = (V_2, E_2) \ldots G_l = (V_l, E_l)$, respectively. It is assumed here that l is at least equal to k, since each group has a minimum cardinality of k. The process of anonymized graph generation will be discussed in terms of the above notations. In the next subsections, we will discuss two methods for construction of the super-templates. We refer to these methods as **ClosureG** and **PatternG** respectively.

3.2.1 Super-template construction for ClosureG In the **ClosureG** method, the super-template of an anonymized group is built as a closure of all the graphs of the group. Consider the edge-weighted graph closure denoted by $\mathbb{G} = (\mathbb{V}, \mathbb{E})$, where $\mathbb{V} = (V_1 \cup V_2 \ldots V_l)$ and $\mathbb{E} = (E_1 \cup E_2 \ldots E_l)$. Each edge closure is associated with a weight which is specific to the edges in the resulting graph.We will see in a later subsection that the weights on these different edges will play a critical role in the synthesis of the anonymized graphs from the super-template. The weight of an edge is the number of times the edge is included in an edge set of the constituent edges in $E_1 \ldots E_l$. Clearly, the weight $w(e)$ of

an edge e can be no larger than the number of graphs l in that particular group.

In order to build such a graph closure from an anonymized group, we recursively merge graphs into the closure. When merging graph $G_1 = (V_1, E_1)$ into a closure $\mathbb{G} = (\mathbb{V}, \mathbb{E})$, we use the NBM algorithm discussed in section 2 in order to obtain a matched list (*matchedlist*) between vertices of a graph and vertex closures of a graph closure. The matches between vertices also implicitly define the matches between edges. We use this implicit matching in order to augment the edge sets and weights of (G) by iteratively processing the edges of the graph G_1. If no match exists in $\mathbb{G}$ for an edge in E_1, then we add that edge to $\mathbb{G}$, and set its weight to 1. On the other hand, if an edge $e \in E_1$ has a match $\phi e \in \mathbb{E}$, we add 1 to the weight of $\phi(e)$.

The NBM subroutine described in section 2 is expressed quite generally in terms of its similarity function. Next, we describe how the similarity function is defined for the case of the **ClosureG** method. The similarity between two nodes in G_1 and $\mathbb{G}$ is computed by comparing their labels. When two vertices have the same label, the corresponding similarity is set to 1. Otherwise, the similarity is set to 0. We note that the NBM algorithm requires a minimum threshold in order to determine the validity of matched pairs. In this case, we set the value of *min_threshold* to 0. This means that any of the vertex-pairs are available for mapping in the algorithm. This non-restrictive approach is used because the underlying graphs may have some diversity in the underlying structure.

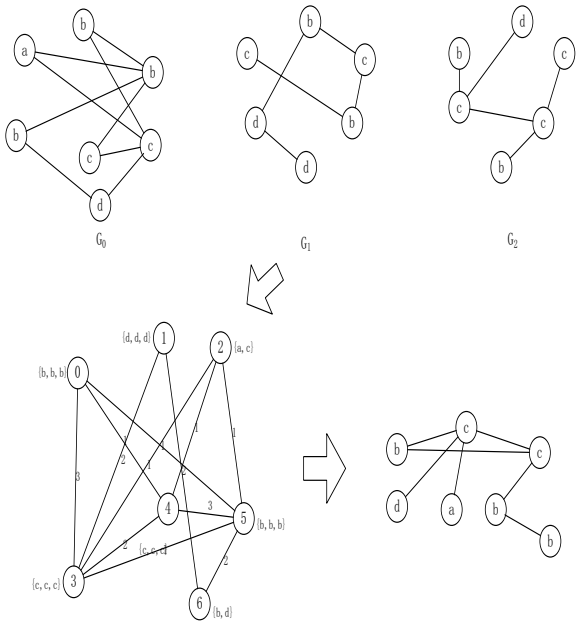

Figure 1: Example of **ClosureG** technique

3.2.2 Super-Template Construction for PatternG In the **PatternG** method, the super-template of an anonymized group is built as a compact super-graph of all *patterns* in the pattern set of the cluster corresponding to an anonymized group. The use of *patterns* rather than the graphs themselves has the advantage that the approach tends to filter out the noise and only preserves the broad trends from the group. These broad trends are used for generation of the anonymized records. We can obtain a super-template by iteratively merging the underlying *pattern* graphs. In each iteration, we merge a patten graph into the super-template in two steps. First, we determine a vertex mapping between the super-template and the pattern graph with the use of the NBM algorithm discussed in section 2. Then, we use this vertex mapping in order to combine the pattern graph into the super template with the use of graph closure definition. This approach is applied repeatedly in order to create a single super-template.

It remains to discuss how the similarity function and thresholds for the NBM algorithm are defined. For constructing super-templates in **PatternG**, the similarity computation between two vertices uses both vertex label matching and vertex degree matching. However, we would like to ensure that vertex matching is provided strict precedence over degree matching. In order to achieve this goal, we pick a sufficiently large value b, and use it as a step value. For the purposes of our application, we use the number of vertices in the underlying graph as the value b. The similarity between two vertices is b if the vertex labels are matched, and the similarity between vertices is $2 \cdot b$, if both vertex labels and vertex degrees are matched. During the matching process, if we need to add weights to a pair of vertices, we just add 1 to the weight of the pair. The idea is that the parameter b should be large enough such that the operation of adding weights only has marginal influence on the matching process. Specifically, we set *min_threshold* of NBM to b, and if b is large enough, the process of adding weights during the matching process will not affect the similarity sufficiently to allow matching unless the matched vertices have at least the same label.

Since the list corresponding to *matchedlist* is not exhaustive in terms of vertices, the super-template may not contain some of the vertices and edges. For each unmatched vertex of the pattern graph which is not contained in *matchedlist*, we will add a corresponding vertex to the super-template, which is used for matching to the pattern graph. Therefore, each vertex in the pattern graph has a match in the super template. We denote this mapping by ϕ_t. We use the same procedure for edges. For each edge (e_1, e_2) in the pattern graph, if there is no edge between $\phi_t(e_1)$ and $\phi_t(e_2)$ in the super-template, we add edge $(\phi_t(e_1), \phi_t(e_2))$ to the super template. We ensure every edge in the pattern graph has a corresponding edge in the super-template.

3.3 Construction of Anonymized Graphs from Super-template The super template can be used to construct the anonymized graphs. In this subsection we shall discuss two methods of synthesizing anonymized graphs from super-templates. These two methods correspond to the **ClosureG** and **PatternG** techniques respectively. Since the base group of the super-templates contain r input graphs, we need to generate r such anonymized graphs from the super-template for that group. This ensures that the output of the algorithm has the same size as the base data. In the next subsections, we will discuss the approaches for constructing the anonymized graphs in detail.

3.3.1 Generation Techniques for ClosureG The super-templates generated the previous phase of the **ClosureG** method are much larger than the base graphs. On the other hand, we would like the newly generated graphs to be of somewhat similar size to the base graphs. We would also like the label distribution to be similar to the case graphs. In order to achieve this goal, we need to determine which edge closures and vertex labels should remain in the newly generated graphs.

The **ClosureG** technique constructs a synthesized graph with the use of a probabilistic approach on every edge-closure in the super-template. For each super-template, we obtain the maximum weight among its edge closures, which is denoted by mw. For each edge closure, we generate a random number in $[0, 1]$. If it is less than the probability $p = w(e)/mw$, we include the corresponding edge and its incident vertices into the synthesized graph. Once the choice of edges has been decided, we decide the choice of vertex labels. For each vertex closure connected to an included edge closure, we flip an unbiased coin to decide which label is applied to the vertex of synthesized graph.

An example of the **ClosureG** method is illustrated in Figure 1. G_1, G_2 and G_3 are three graphs in an anonymized group, and the letters in the circle are the labels of the vertices. We illustrate the generation of the corresponding super-template and the generation of an anonymized pseudo-graph from this template.

3.3.2 Generation Techniques for PatternG The **ClosureG** method uses a probabilistic approach in order to decide the whether an edge closure is selected. The graphs synthesized in this way may cause a different degree distribution from the original graphs, and therefore the synthesized graphs may sometimes show some variations in aggregate structural properties. The **PatternG** method is targeted to synthesize graphs which have a more similar degree distribution. The **PatternG** method maps the original graphs into super-templates with the NBM algorithm, and uses the information in this mapping in order to better preserve the structural information. Furthermore, we note that the super-templates are themselves constructed from frequent patterns rather than original graphs. This ensures that many of the non-typical edges in an anonymized group are not contained in the super-template. This removes the noise specific to individual graphs, and constructs structures which are based on the high-frequency patterns in the template.

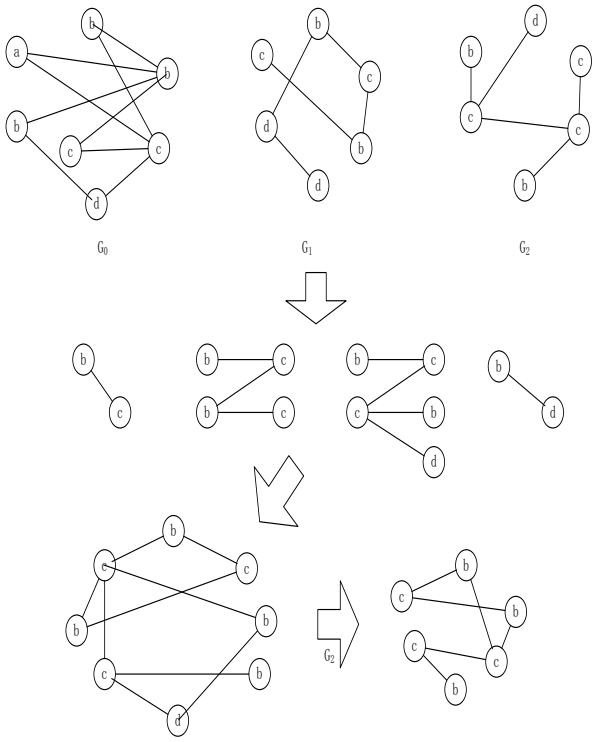

Figure 2: Example of **PatternG** Method

We will use each graph in the base group in order to synthesize an anonymized graph from the super-template. The aim is to ensure that the structural information from the original graph is preserved well. Let us assume that we want to use graph G as the base group to generate an anonymized graph from the super template $\mathbb{C}$. We can perform this operation in two steps. First, we use the NBM algorithm of section 2 in order to create a mapping ϕ_s between a graph $G = (V, E)$ and the super-template $G_t = (V_t, E_t)$. In the **PatternG** method, we consider only the degree of the vertices while defining the similarity function for the NBM algorithm. If the vertex degree difference between a vertex of the super-template and the mapped vertex of the graph is no greater than 2, the step value b is added to the weight of the neighboring vertices in the NBM method. The *min_threshold* of NBM is set to 0 in order to provide the greatest flexibility during the matching process.

For $V_s = \{\phi_s(v_i)|v_i \in V\}$, we can get an induced subgraph $G_s = (V_s, E_s)$ of $G_t = (V_t, E_t)$. In case there are zero vertices

in G_s, for each vertex v with zero degree, we will check if there are any edges connected to $\phi_s^{-1}(v)$. If so, we connect the corresponding edges in G_s, and construct a new graph G'_s. The latter is the synthesized graph created by the **PatternG** algorithm.

An example of the **PatternG** approach is illustrated in Figure 2. The input graphs are the same as those for Figure 1. We obtain four frequent closed patterns with the support threshold 4, and merge them into a super-template. This super-template is used to generate an anonymized graph with respect to G_2.

4 Experimental Results

In this section, we will present the experimental results for our anonymization techniques. In general, we would like to examine whether the structural behavior of the original graphs is preserved, and how the substructure patterns of graphs are affected. We would also like to determine how well the anonymized graphs can be used for a mining application such as classification. This provides us an idea of the utility of the anonymized graphs for a data mining application. We note that all existing work on graph privacy has focussed on anonymization of *nodes and edges* rather than individual graphs. Since this is the first work on anonymization of a *collection of* graphs, there is no baseline to compare our technique against. Nevertheless, we can examine the absolute degradation in quality caused by anonymization of the data, and show that a high level of privacy is possible at the expense of a small amount of degradation.

The experiments were conducted on a chemical compound database which can be derived from the DTP AIDS Antiviral Screen database from the National Cancer Institute. These data sets[1] were also used in [21]. The data set is categorized into three smaller data sets, two of which (CA and CM) were used. The CA data set contained 422 graphs, and the CM data set contained 1081 graphs.

4.1 The Aggregate Properties In this subsection, we will compare the aggregate properties of the anonymized and the base graphs. We mine the set of closed patterns with relative support thresholds of 0.0665 and 0.0931. This corresponds to absolute support thresholds of 28 and 39 for the CA data set, and 72 and 101 for the CM data set. The anonymity level k was set to 20.

The summary properties of graphs include the number of vertices and edges across the different graphs as well as the degree distributions of the different graphs. The summary properties of the base and the anonymized graphs from *CA* are illustrated in Table 1. The corresponding degree distributions are illustrated in Table 2. The summary properties of base and anonymized graphs from *CM* are illustrated in Table 3. From Tables 1 and 3, we can see that the **ClosureG** method generates similar number of edges but more vertices than the original graphs. The **PatternG** method generates similar numbers of edges and vertices to that of the original graphs. Furthermore, it is evident from the degree distribution results in Tables 2, that the **PatternG** method generates more similar distributions to the original graphs than the **ClosureG** method. Since the **PatternG** method uses the structural behavior of the degree distributions, it also results in more similar structural behavior to the original graphs.

We also studied the effect of anonymity level on the effectiveness. We vary the anonymity level k for building anonymized groups, and test the sensitivity of the underlying summary properties. We note that the anonymity level essentially defines the tradeoff between privacy and accuracy. This sensitivity test is performed on the *CA* data set. The support threshold t for mining frequent closed patterns is set to 44. The summary properties of graphs generated by **ClosureG** are illustrated in Table 4, and those generated by **PatternG** are illustrated in Table 5. The results show that the summary properties are not very sensitive to the variation in the anonymity level of the underlying data. It is clear that the average degrees are slightly reduced when the anonymity level k increases. Thus, an increase in privacy results in a reduction of accuracy, though the reduction in accuracy is only modest.

4.2 Classification Application In this section, we will test the classification accuracy on the anonymized data sets. As in [12], we do not split the original data set into a training set and a test set, and anonymize them separately or only one of them. We anonymize the whole data set, and split it into an anonymized training set and an anonymized test set to build classifiers. Each category in the data is anonymized separately, since the category is always available as public information.

For classification, we used LibSVM [9] to classify the data. The anonymized data set is generated by means of the frequent closed pattern set of the original data set with a support threshold t. We will mine the frequent closed patterns from the anonymized data with the same support threshold t, and use the resulting pattern set as features for building the classification model. For each data set, (either original or anonymized), we conducted our experiments using the 10-fold cross validation scheme.

The results for classification accuracy of the **ClosureG** method are illustrated in Table 6, and the results for classification accuracy of the **PatternG** method are illustrated in Table 7. In each case, we have illustrated the accuracy for varying levels of anonymity and support level. The support levels correspond to the thresholds used for pattern mining

[1] The data sets may be downloaded from the following public web site: http://dtp.nci.nih.gov/docs/aids/aids_data.html

	Original	sup=28 ClosureG	sup=39 ClosureG	sup=28 PatternG	sup=39 PatternG
# Graphs	422	422	422	422	422
Tot. Vert.	16714	19872	19700	16714	16714
T. Edges	17853	17929	17717	16800	17065
Vert./Gr.	39.6066	47.09	46.6825	39.6066	39.6066
Edges/Gr.	42.3057	42.4858	41.9834	39.8104	40.4384
Avg. Deg.	1.0682	0.9022	0.8993	1.0051	1.021

Table 1: Summary properties of *CA* and anonymized graphs from *CA*.

Degree	Original	sup=28 ClosureG	sup=39 ClosureG	sup=28 PatternG	sup=39 PatternG
0	87	0	0	58	68
1	3936	10145	10025	5564	5856
2	6969	5512	5578	6452	5762
3	5058	2733	2690	3641	3639
4	663	1047	985	843	1170
5	0	337	314	111	173
6	1	76	87	35	38
7	0	16	17	8	4
8	0	5	4	2	4
9	0	1	0	0	0

Table 2: Degree Distribution of *CA* and anonymized graphs from *CA*.

during the constrained clustering process. From the results, we can see that the classification accuracies on synthesized data sets of both methods are compatible with those on the original data set. Some accuracies on synthesized data sets are even higher than those on that original data set. This is especially the case for the **PatternG** method in which some accuracies are nearly 4% higher than the accuracies on the original data set. This is because the **PatternG** method uses the underlying graph patterns rather than individual edges. This tends to use the aggregate behavior of the underlying data, and avoids much of the noise present in the original data. This has the indirect effect of filtering the noise from the aggregate behavior of the condensed data. As a result, the generated data turns out to be even more effective for classification.

The results of Table 6 and Table 7 also show that the variations in the support level and the anonymized group size do not affect the classification accuracies on the underlying data too much. This suggests that the approach provides a fairly robust classification across a wide variation of the input parameters. For the case of the **ClosureG** method, the degradations in quality were only marginal with increasing anonymity level. On the other hand, for the case of the **PatternG** method, the quality improves at higher anonymity levels. This is because the algorithm is able to determine the most relevant patterns for classification at the higher anonymity levels. This increases the robustness with increasing group size. The robustness with the anonymity level is particularly important because it suggests that a high level of anonymity can be achieved with very little additional cost.

5 Conclusions and Summary

The paper proposes a graph anonymization approach by utilizing the structural behavior of a group of graphs, and generating new anonymized data sets from the super-template of this group. Our approach is a new technique which focuses on generating an anonymized collection of graphs. Previous work has only focussed on the problem of anonymizing nodes and edges in single graphs. This is the first work which examines the anonymity problem in the context of multi-graph collections. Such a problem often arises in the context of many new kinds of data which associates individual records in structural form.

We have proposed two methods under our framework, which are referred to as **PatternG** and **ClosureG** respectively. Both methods are designed to capture the broad structural properties of the underlying data, and use these properties for synthesizing the anonymized graphs. The experimental results show that the anonymized graphs of both the methods are similar to the original graphs in many summary structural properties. The classification accuracies on the generated data set are also compatible with the accuracies

	Original	sup=72 ClosureG	sup=101 ClosureG	sup=72 PatternG	sup=101 PatternG
# Graphs	1081	1081	1081	1081	1081
Tot. Vert.	34387	41428	41123	34387	34387
T. Edges	37032	37075	37068	33994	34319
Vert./Gr.	31.810	38.324	38.042	31.810	31.810
Edges/Gr.	34.257	34.297	34.291	31.447	31.748
Avg. Deg.	1.0769	0.8949	0.9014	0.9886	0.998

Table 3: Summary properties of CM and anonymized graphs from CM.

	k=20, CA	k=30, CA	k=40, CA	k=50, CA	k=60, CA	k=70, CA	k=80, CA
# Graphs	422	422	422	422	422	422	422
Total Vertices	19885	19869	19699	19907	19759	20326	20348
Total Edges	18050	17852	17748	17929	17676	17894	17784
Vertices/Graph	47.12	47.08	46.68	47.17	46.82	48.17	48.21
Edges/Graph	42.77	42.30	42.06	42.49	41.89	42.40	42.14
Avg. Degree	0.908	0.898	0.901	0.901	0.895	0.880	0.874

Table 4: Summary properties of graphs constructed by **ClosureG** method with different cluster sizes.

	k=20, CA	k=30, CA	k=40, CA	k=50, CA	k=60, CA	k=70, CA	k=80, CA
# Graphs	422	422	422	422	422	422	422
Total Vertices	16714	16714	16714	16714	16714	16714	16714
Total Edges	17539	17355	17355	17065	17318	17318	17318
Vertices/Graph	39.61	39.61	39.61	39.61	39.61	39.61	39.61
Edges/Graph	41.56	41.13	41.13	40.44	41.04	41.04	41.04
Avg. Degree	1.049	1.038	1.038	1.021	1.036	1.036	1.036

Table 5: Summary properties of graphs constructed by **PatternG** method with different cluster sizes.

	Original	k=30	k=40	k=50	k=60
sup=100	77.84%	76.25%	75.52%	77.31%	74.39%
sup=120	77.96%	76.04%	75.51%	78.31%	78.23%
sup=140	78.31%	77.78%	78.98%	79.04%	77.80%
sup=160	78.65%	79.17%	79.64%	79.04%	76.33%
Average	78.19%	77.31%	77.41%	78.42%	76.69%

Table 6: Classification Accuracy on synthesized graphs generated by the **ClosureG** method

	Original	k=30	k=40	k=50	k=60
sup=100	77.84%	72.00%	72.72%	74.57%	74.91%
sup=120	77.96%	72.53%	74.98%	83.17%	83.10%
sup=140	78.31%	78.64%	80.58%	82.44%	81.96%
sup=160	78.65%	80.97%	78.78%	82.45%	82.62%
Average	78.19%	76.03%	76.76%	80.66%	80.65%

Table 7: Classification Accuracy on synthesized graphs generated by the **PatternG** method

on the original data set. Thus, the approach provides a high level of anonymity for the underlying graphs at a relatively small cost in terms of accuracy.

Acknowledgements

Jianyong Wang was supported in part by National Natural Science Foundation of China under grant No. 60873171, the National Basic Research Program of China (973 Program) under Grant No. 2011CB302206, and the Program for New Century Excellent Talents in University under Grant No. NCET-07-0491, State Education Ministry of China.

References

[1] C. C. Aggarwal, and P. S. Yu, *A Condensation Based Approach to Privacy Preserving Data Mining*, EDBT Conference, (2004), pp. 183–199.

[2] C. C. Aggarwal, and P. S. Yu, *On Anonymization of String Data*, SDM Conference, (2007).

[3] C. C. Aggarwal, and P. S. Yu, *Privacy-Preserving Data Mining: Models and Algorithms*, Springer, (2008).

[4] C. C. Aggarwal, and H. Wang, *Managing and Mining Graph Data*, Springer, (2010).

[5] C. C. Aggarwal, *Social Network Data Analytics*, Springer, (2011).

[6] R. Agrawal, and R. Srikant, *Privacy-Preserving Data Mining*, ACM SIGMOD Conference, (2000), pp. 439–450.

[7] D. Agrawal, and C. C. Aggarwal, *On the Design and Quantification of Privacy Preserving Data Mining Algorithms*, ACM PODS Conference, (2001), pp. 247–255.

[8] R. J. Bayardo, and R. Agrawal, *Data Privacy through Optimal k-Anonymization*, ICDE Conference, (2005), pp. 217–228.

[9] C.-C. Chang, and C.-J. Lin, *LIBSVM : a library for support vector machines.* (2001), Software available at: http://www.csie.ntu.edu.tw/˜cjlin/libsvm

[10] G. Cormode, D. Srivastava, T. Yu, and Q. Zhang, *Anonymizing Bipartite Graph Data using Safe Groupings*, VLDB Conference, (2008), pp. 833–844.

[11] A. Evfimievski, R. Srikant, R. Agrawal, and J. Gehrke, *Privacy Preserving Mining Of Association Rules*, ACM KDD Conference, (2002), pp. 217–228.

[12] B.Fung, K. Wang, and P. S. Yu, *Anonymizing Classification Data for Privacy Preservation*, IEEE TKDE, 19(5), (2007), pp. 711-725.

[13] M. Hay, G. Miklau, D. Jensen, D. Towsley, and P. Weis, *Resisting Structural Re-identification in Social Networks*, VLDB Conference, (2008), pp. 102–114.

[14] H. He, and A. K. Singh, *Closure-Tree: An Index Structure for Graph Queries*, ICDE Conference, (2006), pp. 38.

[15] D. Kifer, and J. Gehrke, *Injecting utility into anonymized datasets*, SIGMOD Conference, (2006), pp. 217–228.

[16] K. LeFevre, D. DeWitt, and R. Ramakrishnan, *Mondrian Multidimensional K-Anonymity*, ICDE Conference, (2006), pp. 25.

[17] K. Liu, and E. Terzi, *Towards identity anonymization on graphs*, ACM SIGMOD Conference, (2008), pp. 93–106.

[18] A. Machanavajjhala, J. Gehrke, D. Kifer, and M. Venkitasubramaniam, *l-Diversity: Privacy Beyond k-Anonymity*, ICDE Conference, (2006), pp. 24.

[19] P. Samarati, *Protecting respondents identities in microdata release.* IEEE TKDE, 13(6), (2001), pp. 1010–1027.

[20] V. Verykios, E. Bertino, I. Fovino, L. Provenza, Y. Saygin, and Y. Theodoridis, *State-of-the-art in privacy preserving data mining*, SIGMOD Record 33(1), (2004), pp. 50-57.

[21] X. Yan, and J. Han, *CloseGraph: Mining closed frequent graph patterns.* ACM KDD Conference, (2003), pp. 286–295.

[22] X. Ying, and X. Wu, *Randomizing Social Networks: a Spectrum Preserving Approach*, SDM (2008), pp. 739–750.

[23] X. Ying, K. Pan, X. Wu, and L. Guo, *Comparisons of randomization and k-degree anonymization schemes for privacy-preserving social network publishing*, SNA-KDD, (2009).

[24] B. Zhou, and J. Pei, *Preserving Privacy in Social Networks Against Neighborhood Attacks*, ICDE Conference, (2008), pp. 506–515.