The Union-Split Algorithm and Cluster-Based Anonymization of Social Networks *

Brian Thompson
Rutgers University
Department of Computer Science
Piscataway, NJ 08854
bthom@cs.rutgers.edu

Danfeng Yao
Rutgers University
Department of Computer Science
Piscataway, NJ 08854
danfeng@cs.rutgers.edu

ABSTRACT

Knowledge discovery on social network data can uncover latent social trends and produce valuable findings that benefit the welfare of the general public. A growing amount of research finds that social networks play a surprisingly powerful role in people's behaviors. Before the social network data can be released for research purposes, the data needs to be anonymized to prevent potential re-identification attacks. Most of the existing anonymization approaches were developed for relational data, and cannot be used to handle social network data directly.

In this paper, we model social networks as undirected graphs and formally define privacy models, attack models for the anonymization problem, in particular an i-hop degree-based anonymization problem, i.e., the adversary's prior knowledge includes the target's degree and the degrees of neighbors within i hops from the target. We present two new and efficient clustering methods for undirected graphs: *bounded t-means clustering* and *union-split clustering* algorithms that group similar graph nodes into clusters with a minimum size constraint. These clustering algorithms are contributions beyond the specific social network problems studied and can be used to cluster general data types besides graph vertices. We also develop a simple-yet-effective *inter-cluster matching* method for anonymizing social networks by strategically adding and removing edges based on nodes' social roles. We carry out a series of experiments to evaluate the graph utilities of the anonymized social networks produced by our algorithms.

Categories and Subject Descriptors

H.3.5 [**Information Storage and Retrieval**]: Online Information Services—*data sharing*; I.5.3 [**Pattern Recognition**]: Clustering—*algorithms, similarity measures*

General Terms

Algorithms, Measurement, Performance, Security

Keywords

Graph Anonymization, Social Networks, Clustering Algorithms, Social Role

1. INTRODUCTION

Knowledge discovery on social network data can uncover latent social trends and produce valuable findings. A growing amount of research finds that social networks play a surprisingly powerful role in people's behaviors. For example, medical researchers discovered that obesity spread from one person to another through social connections, which follows a similar pattern to that of viruses [17]. A recent study on 12,067 people shows that the decision for a smoker to quit smoking is strongly influenced by the people in the smoker's social network [17]. This evidence indicates the precious value of social network data in shedding light on social behavior, health, and well-being of the general public.

Before the social network data can be released for research purposes, the data needs to be anonymized. Anonymization is a crucial process to ensure that released social network data does not disclose sensitive information of users. Depending on the privacy model considered, anonymization prevents an individual participant in the social network from being identified.

Data anonymization techniques have been extensively studied on relational databases with several privacy models (e.g., k-anonymity [13, 14, 18], l-diversity [9], and t-closeness [7]). Most of the existing methods can only handle relational data. Social networks are usually viewed as an undirected graph with or without edge labels. Thus, most of the known anonymization approaches such as suppression or generalization do not directly apply to social network data.

For the past decade, graphs have been intensively studied to model the web, in particular how web pages and links have an impact on searching and surfing. The field of graph mining generated many exciting results on discovering trends and new knowledge from web graphs. With the availability of multi-core personal computers and cloud computing (a.k.a. high-performance web services coupled with scalable data centers), an unprecedented amount of data from various aspects of our digital society is being collected and analyzed, including social networks, virtual communities, recommendation data, network traces, search queries, and communication patterns. These types of data can be mod-

*This work was partially supported by NSF Grant CNS-0831186.

ASIACCS '09, March 10-12, 2009, Sydney, NSW, Australia

eled as graphs, e.g., Netflix Prize data [15] can be viewed as a huge and sparse bipartite graph [5].

Very recently, several graph anonymization solutions have been proposed to de-identify social networks using techniques such as strategically or randomly inserting or deleting edges or nodes. One major challenge in social network anonymization is the complexity. Zhou and Pei proved that a particular k-anonymity problem trying to minimize the structural change to the original social network is NP-hard [21]. They show that the problem can be reduced to a k-dimensional perfect matching problem, which is NP-hard. Since finding the optimal anonymized social network can be hard, the main goal of social network anonymization is to develop efficient heuristics that strike a balance between *preserving the original graph structure* and the *privacy of individuals*.

Furthermore, it is not clear how to quantify the decrease in utility incurred from anonymizing a graph. One way to measure the difference between the original and anonymized graphs is to count the number of nodes and edges that were added or removed. However, this may not always be an accurate quantification of the effect anonymization has on utility. For example, a new social connection between a pair of co-workers may have little effect on the overall behavior of their social network, but two important business executives meeting over lunch may have a much more significant impact on theirs.

In this paper, we take the first step to reconcile the difference between preserving structure and preserving utility in social network graphs. We argue that an important and unique graph property for social networks is the *social role* of an individual. Intuitively, a social role is "the position or purpose that someone or something has in a situation, organization, society or relationship" [3]. A social network contains the information about the social role of each participant that is usually reflected in the way people interact with each other and in *social connections* in particular. Therefore, a good anonymization algorithm for social networks needs to preserve as much as possible the social connectivity of individuals.

Our Contributions Our contributions are summarized as follows.

1. We give two efficient clustering heuristics, called *bounded t-means* and *union-split*, for clustering individuals in a social network into groups with similar social roles, while satisfying a minimum cluster size constraint. The *bounded t-means* and *union-split* algorithms reach beyond the specific social networks problem considered, as they can be used to cluster structured or unstructured data based on an arbitrary distance metric.

2. We describe a simple yet effective *matching-based anonymization* method for social networks that strategically adds and removes edges based on a node's inter-cluster connectivity. Our experiments show that our anonymization method combined with the union-split clustering algorithm outperforms the recently proposed graph generalization method [6] in preserving the structural properties of social networks.

Organization of the Paper We give the definitions, attack models, and privacy models in the next section. We present our bounded t-means and union-split clustering algorithms in Section 3. In Section 4, we describe our graph anonymization algorithms under our degree-based privacy models. Our experimental evaluation is given in Section 5. Related work is described in Section 6. In Section 7, we give some conclusions and describe plans for future work.

2. DEFINITIONS, ATTACK MODELS, PRIVACY MODELS

In this section, we describe the privacy model and adversary model that we consider in our anonymization algorithms. In particular, we formally define the type of prior knowledge that the adversaries are allowed to have.

DEFINITION 1. *A* **social network graph** *$G(V, E)$ is a simple undirected graph with a set of nodes or vertices denoted by V and a set of unlabeled edges denoted by E. A node $v \in V$ represents an individual in the social network. An edge $(v_i, v_j) \in E$ signifies a social relationship between the two individuals represented by nodes v_i and v_j.* [1] *We denote the degree of a node by $d(v)$. Let $n = |V|$ and $m = |E|$ be the total number of nodes and edges in G, respectively.*

Immediate neighbors of a node v are denoted by $N(v, 1)$, where 1 represents one-hop neighbors. Two-hop neighbors (i.e., neighbors' neighbors) are denoted by $N(v, 2)$, and so on.

We consider a type of re-identification attack where the adversary has prior knowledge about a node and its social connections. In particular, we consider a general type of *degree-based attack* where the adversary has prior knowledge of the degrees of nodes within a given radius of the target node.

DEFINITION 2. *An* **i-hop degree-based attack** *on a social network G is one in which the adversary has prior knowledge of the degree $d(v)$ of a target node v and the degrees of some or all of v's j-hop neighbors for all $j \leq i$. That is, the adversary may know $\{d(u)|u \in N(v, j)\}\ \forall j \leq i$. In a 0-hop degree-based attack (degree-based attack for short), the adversary only has prior knowledge of $d(v)$. Given the anonymized social network graph G', the adversary's goal is to successfully re-identify target node v.*

Based on the above attack model, we define the anonymization goal of a social network. Intuitively, the purpose of anonymization is to prevent re-identification attacks by modifying the graph so that to the adversary, each node is indistinguishable from many other nodes in the graph.

DEFINITION 3. *The i-hop fingerprint of a vertex $v \in V$, denoted $f_i(v)$, is the sequence of degree sets $d(v)$, $\{d(u)|u \in N(v, 1)\}$, ..., $\{d(u)|u \in N(v, i)\}$.*

DEFINITION 4. *A social network graph $G(V, E)$ is k-anonymous against i-hop degree-based attacks, if for each node $v \in V$ there exist at least $k - 1$ other nodes $u_1, \ldots, u_{k-1} \in V$ such that $f_i(v) = f_i(u_j)\ \forall j \in [1, k - 1]$.*

[1] Graphs with labeled edges to express more complex social relationships such as spouse-of, teacher-of, etc, will be studied in our future work. However, our techniques are general enough to apply to many notions of node similarity.

In our paper, we present methods for anonymizing social network graphs. That is, given a social network graph G, a privacy parameter k, and an adversary model (degree-based or 1-hop degree-based adversary), we provide an algorithm for generating a graph G' that is k-anonymous against the given adversary, and preserves as best as possible the structural properties of the original graph.

3. CLUSTERING METHODS

Some previous anonymization techniques ([5], [21]) take a greedy approach to anonymization, where anonymization groups are chosen in an *ad-hoc* way. Instead, we focus on a cluster-based approach that first partitions V into groups of similar vertices, and then anonymizes vertices within each group. This enables us to take a global approach that uses the clustering information to more cleverly anonymize the graph.

In this section, we present and analyze two new clustering algorithms, the bounded t-means and the union-split clustering algorithms. One classic clustering technique is known as the k-means algorithm. To avoid confusion with the k in k-anonymity, we use the term t-means throughout the paper. We start with some definitions.

3.1 Definitions and Tools

The distance metrics that we use to measure similarity of two graph vertices are defined next. Definition 5 can also be generalized to the i-hop degree model for $i > 1$, which is omitted here. These distance metrics are used in both the clustering and anonymization algorithms.

DEFINITION 5. **Distance metric**
In the (0-hop) degree-based model, we define the distance between vertices u and v as

$$D(u, v) = |d(u) - d(v)|.$$

In the 1-hop degree-based model, we define the distance between vertices u and v as

$$D(u, v) = |d(u) - d(v)| + \sum |d(x_j) - d(y_j)|$$

where $x_1, x_2, \ldots \in N(u, 1)$ with $d(x_1) \geq d(x_2) \geq \ldots$, and $y_1, y_2, \ldots \in N(v, 1)$ with $d(y_1) \geq d(y_2) \geq \ldots$ Essentially, the neighbors of u and v are sorted based on their degrees, and then the sum of differences in the corresponding neighbors' degrees is computed. If $|N(u, 1) \neq N(v, 1)|$, then we use zero as the default degree of a node that does not exist. Intuitively, our distance metric attempts to quantify the difference in local structure between two vertices.

We define the distance from a vertex u to a cluster c as $D(u, c) = D(u, v_c)$, the distance from u to the cluster center v_c (also a vertex – see Section 3.2). We define the distance between two clusters to be $D(c_1, c_2) = D(v_{c_1}, v_{c_2})$.

DEFINITION 6. **Marginal cost** *Consider a social network graph $G(V, E)$, and let $c_1, \ldots, c_t \subset V$ be disjoint clusters. Let $v \in V$ be any vertex in the graph, and let c_i and c_j be two arbitrary clusters. We define the marginal cost $g(v, c_i, c_j) = |D(v, c_i) - D(v, c_j)|$.*

DEFINITION 7. **Surrogate** *Consider a social network graph $G(V, E)$, and let $c_1, \ldots, c_t \subset V$ be disjoint clusters. We define the surrogate of v to be the cluster of size $< k$ whose center is closest to v, where k is the privacy parameter. Intuitively, the surrogate is the nearest available cluster to v.*

3.2 Mode-Based Cluster Centers

Many clustering algorithms and applications require the notion of a *cluster center*, usually a point in the domain of the dataset that represents that cluster. When the domain is a vector space of real number coordinates, the most natural choice of center is the vector average of all points in the cluster. Since the domain of graph vertices does not fit nicely into the real coordinate model, [2] we must define our own notion of cluster center.

One idea is to have a *discrete center*. That is, require that the center of a cluster be one of the nodes in that cluster. This type of approach is similar to that taken in [21]. However, this restriction may yield poor results if none of the nodes provide an accurate representation of the cluster. Instead, we opt for a *virtual center*, which need not correspond to an existing node in the graph.

Under the (0-hop) degree model, we define the center of a cluster c to be a vertex v_c with degree equal to the average degree over all vertices in c, rounded to the nearest integer.

Computing a cluster center under the 1-hop degree model is more complicated because the center must have both a degree and a list of neighbors' degrees. To accomplish this task, we find that a *statistical mode-based* method works better than the commonly used mean or median. In statistics, the mode is the value that occurs the most frequently in a data set or distribution. We develop a simple yet effective counting method using mode to find the cluster center. We illustrate an example in Table 1 in the appendix. The detailed procedure is as follows.

1. Let $c = \{v_1, \ldots, v_k\}$ be a cluster of graph vertices. Calculate the degree of the new cluster center to be $d = \text{round}\left(\frac{1}{k}\sum_{i=1}^{k} d(v_i)\right)$, the rounded average degree over vertices in c. Allot space for d neighbor degrees.

2. Iterate d times: Find the mode η, that is, the number that shows up in the most neighbor degree sequences. Add η to the list of neighbor degrees for our new center. Remove one copy of η from each member vertex's neighbor degree sequence in which it appears.

Now equipped with all the necessary tools, we proceed to the bounded t-means and union-split clustering algorithms.

3.3 Bounded t-Means Clustering Algorithm

The task of clustering data has been studied in great depth. In practice, however, it has been observed that classic clustering algorithms frequently produce small or empty clusters, especially when clustering high-dimensional datasets. In order to realize k-anonymity, we require each cluster to have size at least k. The lack of a simple and efficient algorithm for minimum-size clustering in the literature has led us to develop two of our own. Our first solution builds on the conventional t-means algorithm, which has no

[2] In general, the task of mapping vertices' local subgraphs into a real coordinate system that preserves isomorphism is reducible from the graph isomorphism problem, which is not known to have a polynomial-time solution. [20]

minimum-size constraint. We refer readers to machine learning literature for details about conventional t-means. [11]

Our bounded t-means clustering method is described as follows.

1. Let $t = \lfloor n/k \rfloor$. Arbitrarily choose t vertices to be cluster centers and denote them as $v_{c_1}, \ldots, v_{c_t}$. Denote the t clusters by $c_1, \ldots, c_t$. Initially, all clusters are empty.
2. For each vertex $v \in V$:
   (a) **Cluster assignment** Add vertex v to the nearest cluster c_i according to a chosen distance metric, e.g. the one given in Definition 5.
   (b) **Bumping** If $|c_i| = k + 1$, i.e. c_i was full and already had k members, then perform the following procedure:
   For each vertex $u \in c_i$, compute the marginal cost $g(u, c_i, c_j)$, where c_j is the surrogate cluster of u (definitions given in Section 3.1). *Bump* the vertex u^* with the lowest marginal cost to its surrogate cluster c_j.
   (c) **Extra vertices** If $|V|$ is not a multiple of k, remainder vertices may be safely placed in their nearest cluster, without fear of violating the minimum-size constraint.
3. **Cluster update** For each cluster c_i, a new cluster center $v^*_{c_i}$ is computed. For our work, we use the methods described in Section 3.2. If the new cluster centers are the same as for the previous iteration, then an equilibrium has been reached and the algorithm terminates. Otherwise, repeat from Step 2. [3]

3.4 Union-Split Clustering Algorithm

Neither the conventional t-means algorithm nor our bounded version guarantee to produce a globally optimal clustering solution. One property of t-means algorithms is that the initial set of cluster centers are chosen arbitrarily, which may affect the clustering outcome. To avoid this variability in clustering results, we design a new deterministic clustering algorithm, the *union-split algorithm*, which is described below.

1. Initialize each vertex to be in its own cluster.
2. Compute all pair-wise distances between cluster centers (see Definition 5). For each cluster, maintain the next nearest cluster to it using a min-heap data structure.
3. While there exists an undersized cluster ($< k$ members):
   (a) Choose an undersized cluster c whose distance to its nearest cluster (full or undersized) is the shortest. Union cluster c with its nearest cluster.
   (b) If the combined cluster c' is overfull (size $\geq 2k$), split it into two clusters each of size $\geq k$. Splitting may be accomplished by finding the two vertices in c' that are farthest from each other, and then applying our bounded t-means algorithm from Section 3.2 with $t = 2$ to ensure that the size constraint is satisfied.
   (c) Update all relevant cluster distances.
4. When all clusters are full, stop.

[3]In our experiments, we impose an upper limit on the number of iterations, but equilibrium is usually achieved before the limit is reached.

THEOREM 3.1. *The union-split clustering algorithm converges in a finite number of iterations. In particular, at each iteration the number of clusters whose sizes are under k is strictly reduced.*

THEOREM 3.2. *The complexity of the union-split clustering algorithm is $O(n^2 \log n)$, where n is the number of vertices in the original graph.*

Proofs of the theorems are given in the appendix.

4. INTER-CLUSTER MATCHING FOR ANONYMIZATION

Once we have clustered the vertices in a graph, the problem still remains of anonymizing the vertices within each cluster. That is, given a set of vertices in a graph, modify the graph so that those vertices are indistinguishable to a particular adversary.

One approach is called *graph generalization* [6]. The idea is simple: once the nodes are grouped based on their similarities, a general description of the graph consisting of the number of nodes in each cluster and the numbers of edges between each pair of clusters is revealed, and nothing else. For example, in a generalized graph, we may know that cluster 1 has 5 nodes, 10 internal edges, and 6 external edges to cluster 2, etc. To use the published generalized graph for research purposes, one must randomly generate a sample graph in accordance with the generalized description. Although the generalization procedure provides strong privacy and is simple to carry out, we find that it may have negative impacts on the utility of the anonymized graphs, as we show in our experiments in Section 5.

Figure 1 illustrates a simple example where the generalized graph approach to anonymization might perform poorly. Due to the way edges are randomly inserted when generating a sample anonymized graph, samples of a single generalized graph may produce large variations in graph properties. In Figure 1, the solid black vertices represent individuals with social roles of great influence over their local networks. While the sample anonymized graph preserves some local structure, much of the high-level graph structure is lost. In our work, we strive to preserve social connectivity by minimizing the changes introduced to the original graph.

In the following sections, we present a novel approach to graph anonymization that we call *inter-cluster matching*. Our method takes a clustered graph and strategically adds and removes edges to anonymize the graph. We present algorithms for both the degree-based and 1-hop degree-based privacy models.

4.1 Basic Inter-Cluster Matching Method

The *Basic Inter-Cluster Matching* algorithm is for the 0-hop degree-based privacy model, where the adversary only has prior knowledge about the degree of the target node, not about its neighbors. The anonymization task is to adjust the

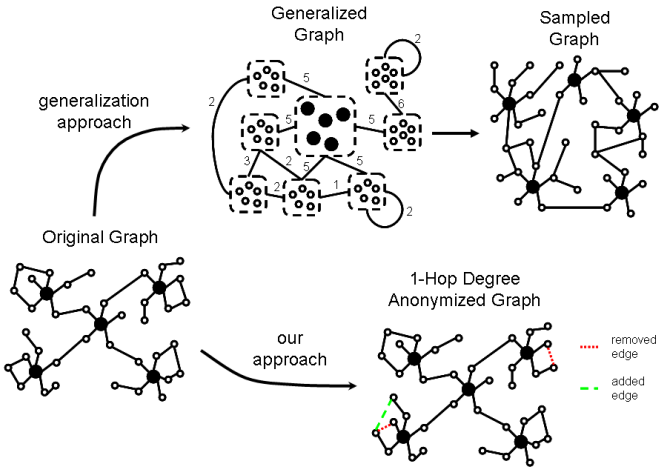

Figure 1: A drawback of the generalized graph approach to anonymization.

degrees of nodes so that nodes within a cluster have the same degree. Our procedure is described as follows:

1. Cluster using any clustering algorithm and compute the (rounded) average degree of nodes within each cluster.
2. For each node, determine how many edges it must add or remove in order to match the degree of its cluster center.
3. Match up vertices that are adjacent but both have too many edges, and remove the edge between them.
4. If there are still vertices with too many edges, remove the necessary number of edges arbitrarily.
5. Finally, match up vertices that have too few edges and join them with an edge.
6. Add a fake vertex if needed. It can easily be shown that at most one fake vertex needs to be added, and it will likely be easy to anonymize because low-degree vertices are common in most social networks.

4.2 Extended Inter-Cluster Matching Method

We build on the above simple method to design an *Extended Inter-Cluster Matching* algorithm so that the resulting anonymized graph is robust against 1-hop degree-based attacks. The extended method is more complicated, as it needs to match up not only nodes themselves but also their neighbors. Suppose a vertex v has neighborhood degrees $\{5, 4, 3, 2\}$, and the cluster center is $\{4, 4, 3\}$. Then v needs to remove the edges to the vertices of degrees 5 and 2, and add an edge to a vertex of degree 4. The removal process is simple - just remove those edges from the graph. To add an edge, however, is more complicated. To accomplish this, we use a more sophisticated version of inter-cluster matching.

In this case, the other endpoint needs to be a vertex of degree 4 *that also needs to gain a neighbor of degree 3*. To state it generally, a vertex of degree X that needs to gain a neighbor of degree Y must be matched with a vertex of degree Y that needs to gain a neighbor of degree X. Note that for all these purposes we use the anonymized degrees of the vertices, not the actual degrees. Our algorithm proceeds as follows:

1. Cluster using any clustering algorithm and compute the cluster centers using the mode-based method from Section 3.2.
2. Match up vertices that are adjacent but both desire to lose their common edge, and remove the edge.
3. If there are still vertices who desire to lose edges, remove those edges accordingly.
4. Precompute a *neighborhood matching table* as follows. The rows represent the anonymized degree of a vertex, or the *I am* of that vertex. The columns represent the neighbor degrees that need to be gained, or the *I need* of a vertex. The cell at (X, Y) contains a multi-set of all vertices with anonymized degree X that need to gain a neighbor of degree Y.
5. Match up remaining vertices in a way that brings them mutual benefit: for each vertex at cell (X, Y) in the table, pair it up with a vertex at cell (Y, X), adding an edge between the two and removing them from the table. Anytime multiple options are available, heuristics may be used to choose the edge that best preserves the social roles of the vertices involved.
6. If there are vertices left in the table for which the complementary table entry is vacant, then create fake vertices with the required degrees to pair up with these left-over vertices.

The above inter-cluster matching approach can be further generalized to handle i-hop degree-based attack models for $i \geq 2$. Due to space limitations, we omit the details here.

5. EXPERIMENTAL RESULTS

We run our experiments on Intel(R) Core(TM)2 CPU 2.40GHz machines with 2G memory, and running Fedora

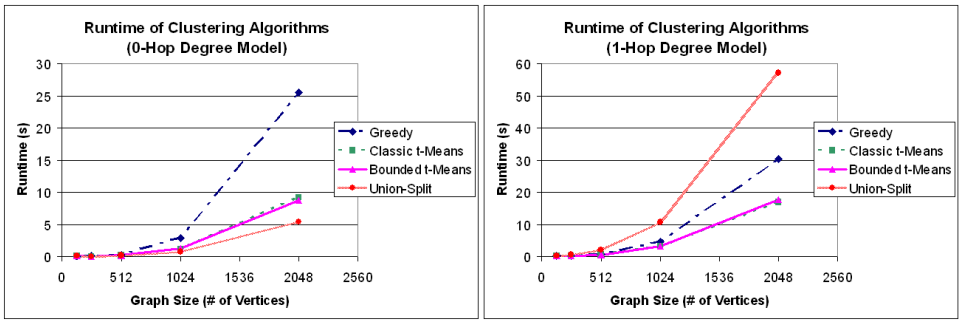

Figure 2: **Running time of several clustering algorithms on graphs of varying size. The graph to the left is under the 0-hop degree model, and the graph to the right is under the 1-hop degree model.**

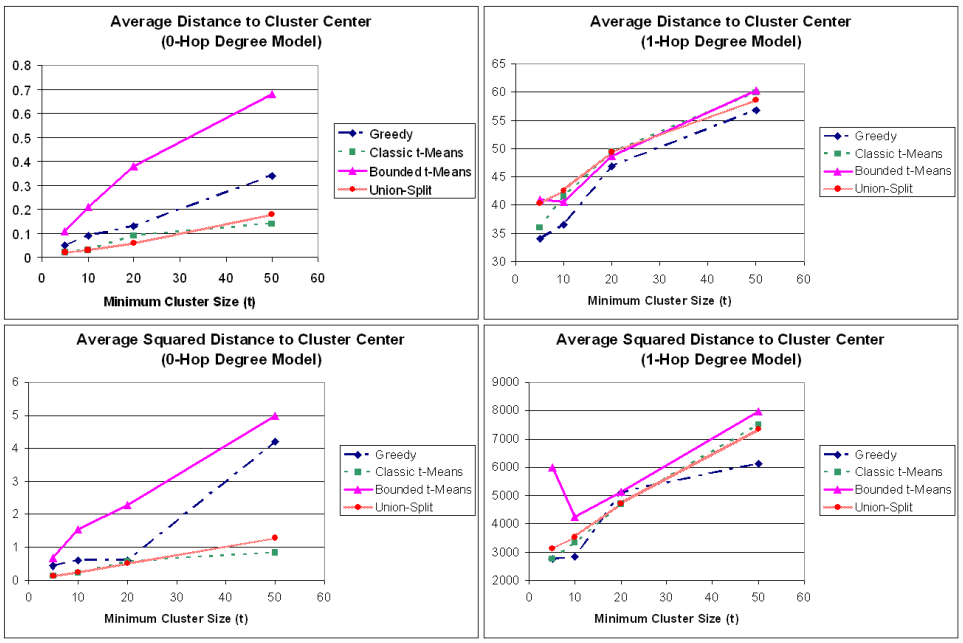

Figure 3: **The average distances and average squared distances of vertices to their cluster centers in an R-MAT(4096, 12) graph using four different clustering methods. The graphs to the left are under the 0-hop degree model, and the graphs to the right are under the 1-hop degree model.**

6 Linux. We implement all of our algorithms in Java. We write a program to generate graphs using the R-MAT algorithm [4]. R-MAT generated graphs have a power-law vertex degree distribution and small-world characteristic, key properties exhibited by social networks [10]. All the experiments described in the following are tested on R-MAT generated graphs. R-MAT(n, d) represents a graph with n vertices and average degree of d. The greedy method in our experiments is briefly described as follows: (1) pick an arbitrary node $v \in V$, (2) find the $k-1$ remaining nodes most similar to v, (3) consider those k nodes an anonymization group and remove them from the set. (4) Repeat steps 1-3 until all nodes are grouped. All versions of t-means algorithms are allowed to run for at most 10 iterations.

The running time experiments are on R-MAT-generated graphs with $|V| = \{128, 256, 512, 1024, 2048\}$, and average degree of $\log|V|$. The results are shown in Figure 2. The performance of various clustering algorithms in terms of the average distance and average squared distance is shown in Figure 3. [4] For the 0-hop degree-based model, our union-split algorithm runs in roughly half the time of the classic or bounded t-means algorithms, all of which are significantly faster than the greedy algorithm. This is expected, since the greedy algorithm has running time complexity of $O(n^2)$ and the t-means algorithms run in $O(nt)$ time, whereas union-split only involves $O(n)$ union or split operations. Although the asymptotic running time complexities are the same for the 1-hop degree-based model, union-split suffers from the high overhead cost of computing the center of a cluster (using our mode-based method), which it must perform at each iteration.

[4]We also measure distances for graph sizes 512, 1024, 2048. Due to space limitations, we only report results on graph size 4096.

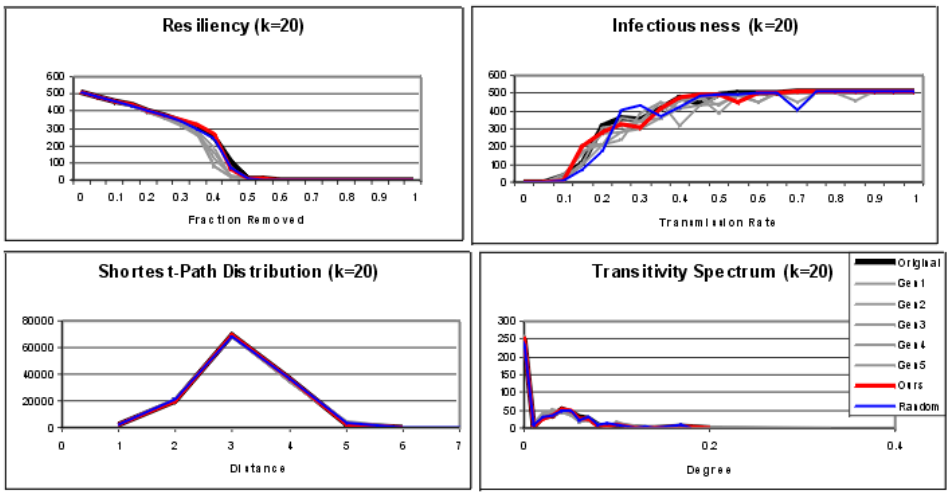

Figure 4: **Utilities of the anonymized graphs in comparison to the original graph R-MAT(512, 9) under four different metrics for $k = 20$. In each graph, the red line represents our anonymization method; the thick black line represents the original graph; the grey lines represent several samples of generalized graphs; and the blue line is the random graph. Y-axis is frequency in the lower two sub-graphs, the size of largest connected component in resiliency, and the number infected in infectiousness.**

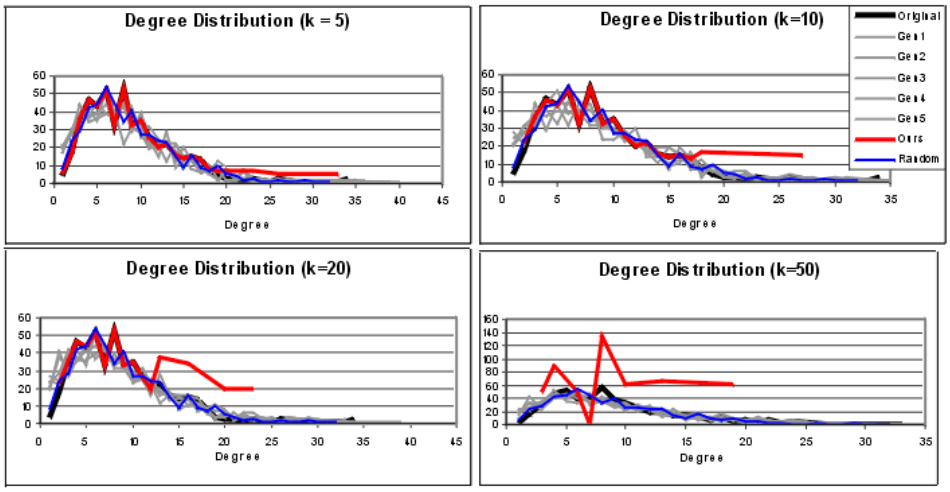

Figure 5: **Comparison of the degree distribution as another utility measure under different $k = 5, 10, 20, 50$ for R-MAT(512,9). Y-axis is frequency.**

Overall, we find that our union-split algorithm performs the best in terms of reasonable run-time and low average distances under both privacy models in most cases. However, in a few cases, union-split produces higher average distances and average squared distances than greedy or bounded t-means. The reason for this observation is that union-split may produce large cluster size between $[k, 2k)$, whereas others produce clusters of size exactly k. As the cluster size increases, the average distance and squared distance from the cluster center increase. On the other hand, larger clusters bring better privacy as the crowd of similar nodes gets larger. Thus, these sets of experiments show a trade-off between clustering performance and privacy. In general, union-split consistently performs well for different graph sizes, cluster sizes, and adversary models. Therefore, in the following tests of the anonymization methods and their utility evaluations, we use union-split as the clustering algorithm.

We have five measures of utility to evaluate how close the anonymized graphs are to the original one. The utility metrics are (1) degree distribution, (2) shortest path distribution (degrees of separation), (3) transitivity spectrum, (4) resiliency, and (5) infectiousness. Our utility results are shown in Figures 4 and 5. The *transitivity* (or clustering coefficient) of a vertex is the fraction of pairs of its neighbors that share an edge. We measure the distribution over the transitivity values of all vertices in the graph. *Resiliency* measures the size of the largest component after deleting a fraction of the nodes in the graph, starting with those of largest degree. *Infectiousness* computes the number of infected nodes corresponding to a transmission rate.

Consider the degree distribution in Figure 5. When the graph size n is relatively small and k is relatively large (e.g., $n = 512$ and $k = 50$), our anonymized graph has larger deviations from the original graph due to the k-anonymity requirement on the degree. In comparison, generalized graphs do not demonstrate this type of behavior as they are under a different privacy model. In most cases, however, graphs anonymized by our method (red lines) produce quite accurate utility values compared to the original graph (black lines). In comparison, samples of the generalized graphs following the approach in [6] (grey lines) produce a large degree of variation in some utilities. Certain utility measures (e.g., degree distribution) are better at distinguishing the different anonymization methods than others (e.g., shortest path distribution).

Interestingly, we find that the generalized graph samples give similar utility values to those of an independently, randomly generated R-MAT graph (blue line). This demonstrates that while they might preserve properties common to many social networks, *generalized graphs may not accurately represent important differences between social networks.*

The advantages of our anonymization approach can be summarized as follows:

1. As demonstrated by Figures 4 and 5, the graphs anonymized via inter-cluster matching very closely represent the original graphs for all five utility measures, except for the discrepancy in degree distribution explained above.

2. The generalized graph method introduces a high degree of variability in anonymized graph samples. Therefore, while there is a chance that their method will produce a graph that very closely resembles the original, there is also a chance that a generalized graph sample will be a very poor representative of the original social network.

3. By attempting to retain both local and global elements of graph structure, our methods may perform well in preserving other properties pertaining to social connections within a network that are not measured here.

6. RELATED WORK

Data anonymization has traditionally been studied by the database community, as the focus is on relational data. However, because the formulation and analysis of privacy problems, including attack models and anonymization goals, bear the same spirit as security and cryptography issues, interest from researchers in the security and cryptography communities has grown significantly in recent years.

Hay *et al.* has recently proposed an anonymization method by grouping similar nodes into anonymous groups and then publishing a *generalized graph* [6]. The generalized graph describes the inter- and intra-group connectivity, e.g., there are 10 edges connecting group A and B, 50 internal edges in group A, 18 internal edges in group B. A sample of a generalized graph is then needed if one wants to make use of the anonymized social networks for data mining. In order to group nodes in the social network, they use simulated annealing to do a random exploration of the search space for a good set of clusters, where the branching factor may be exponential.

Our anonymization algorithm bears a similar spirit to the neighborhood anonymization approach [21], but there are several major differences. Zhou and Pei's approach uses a greedy approach to group and anonymize similar nodes in one phase. Their method also relies on frequent checks for isomorphisms between subgraphs, a problem which has no known polynomial-time solution. Indeed, their algorithm runs in exponential time in the size of the subgraphs examined, although they provide optimizations that make the computation feasible in practice. In comparison, our anonymization method is more lightweight. Our approach happens in two stages: cluster and anonymize. We cluster graph vertices based on simple distance metrics first, and then anonymize to the clusters. Our edge insertion and deletion are performed strategically to match needy nodes and avoid unnecessary modifications.

The t-means algorithm is a data mining technique that clusters n objects into t groups ($t < n$) based on a given distance metric. Bennett, Bradley, and Demiriz proposed a constrained t-means algorithm that accommodates the constraint of minimum cluster size [1]. They reformulate the clustering problem as a Maximum Network Flow problem, which is then solved using dynamic programming. Our solution is more natural, and is easier to implement and use.

Our data anonymization problem is related to privacy-preserving relational data, which has been extensively studied in the past decade. Most of the literature considers the attack model as re-identification of individuals by joining the published table with some external tables modeling the background knowledge of users. To defend against this type of attacks, the mechanism of k-anonymity was proposed in [16, 18]. Specifically, a dataset is said to be *k-anonymous* if every group of tuples that are of the same values on the quasi-identifier attributes (i.e., the minimal set of attributes in the table that can be joined with external information to re-identify individual records) consists of at least k tuples. The larger the value of k, the better the privacy is protected. To improve on k-anonymity, new notions (e.g., l-diversity [9], t-closeness [7], (α, k)-Anonymity [19]) have been proposed to provide stronger privacy. We adapt the concept of k-anonymity to our problem since it is the most essential and most applicable privacy model. However, because relational data and graph data have intrinsically different representations, many techniques for relational data cannot be directly applied to social networks.

Byun *et al.* came up with a clustering idea for anonymizing relational databases [2]. They presented a solution for database records that are in the form of vectors, where each dimension is a numerical value or a label in a fixed hierarchy. Again, their work is for relational databases, and does not have a direct application to data in the form of a graph.

Cormode *et al.* developed a safe grouping approach for anonymizing bipartite graphs [5]. In comparison, we consider general undirected graphs. Compared with social network graphs that we consider, bipartite graphs generated by recommend systems differ in several aspects. First, unlike in the social network graphs where every node is de-identified, recommendation graphs contain considerable numbers of nodes whose identifications are revealed (e.g., the names of movies, music, books, etc.). Second, unlike social network graphs which are normally dense, most real-world recommend system datasets are very sparse [12]. Thus, anonymization techniques developed for recommen-

dation data have a different focus and emphasis from the ones for social networks.

7. CONCLUSIONS AND FUTURE WORK

In this paper, we studied the data anonymization problem in static social networks. In particular, we investigated whether clustering algorithms can be used for anonymization and how effective they are in finding similar nodes in social networks. To that end, we developed new constrained graph clustering methods, namely the *bounded t-means* and *union-split* clustering algorithms, and showed that they are effective and general approaches to grouping similar nodes on large social network graphs for anonymization.

Our similarity metrics and anonymization algorithms were based on the k-anonymity privacy model. We considered an i-hop degree-based neighborhood adversary model that can be generalized to capture more complex neighborhood knowledge of an adversary. We implemented our clustering and anonymization algorithms in Java and ran experiments on synthetic social network graphs. Our experimental results demonstrated that our methods are effective in preserving the statistical graph utilities studied.

For future work, we decide to formally define and study the relationships between *l-diversity and social relationship attacks* in social networks. In this paper, we consider adversaries whose goal is to re-identify a target in the anonymized social networks. A different type of attack is what we call *social relationship attacks*, or *link disclosures* according to Liu and Terzi [8], where the adversary's goal is to gain more information on the target's social connectivity, for example, to learn whether or not the target node has an edge with a socially popular node of high degree. Although the adversary may not be able to identify a target, she may learn that the target is socially connected with an individual that is well-connected (e.g., a popular person with Britney Spears' status). In order to prevent these kinds of social relationship attacks, our privacy model needs to be expanded to introduce the concept of *diversity*, similar to the l-diversity definition in anonymizing relational data [9]. Intuitively, the *diversity in social networks* means that certain sensitive nodes with distinct properties need to connect to at least l diverse anonymous groups. We plan to formalize the definition and investigate how l-diversity affects the utilities of the anonymized graphs.

8. REFERENCES

[1] P. Bradley, K. Bennett, and A. Demiriz. Constrained k-means clustering. Technical Report MSR-TR-2000-65, Microsoft Research, 2000.

[2] J.-W. Byun, A. Kamra, E. Bertino, and N. Li. Efficient k-anonymization using clustering techniques. In *Proceedings of the 12th International Conference on Database Systems for Advanced Applications (DASFAA)*, volume 4443 of *Lecture Notes in Computer Science*, pages 188–200. Springer, 2007.

[3] Cambridge English Dictionary. `http://dictionary.cambridge.org/define.asp?key=68410&dict=CALD`.

[4] D. Chakrabarti, Y. Zhan, and C. Faloutsos. R-MAT: A recursive model for graph mining. In M. W. Berry, U. Dayal, C. Kamath, and D. B. Skillicorn, editors, *SDM*. SIAM, 2004.

[5] G. Cormode, D. Srivastava, T. Yu, and Q. Zhang. Anonymizing bipartite graph data using safe groupings. In *Proceedings of the International Conference on Very Large Data Bases (VLDB)*, 2008.

[6] M. Hay, G. Miklau, D. Jensen, D. Towsley, and P. Weis. Resisting structural identification in anonymized social networks. In *Conference on Very Large Databases (VLDB)*, 2008.

[7] N. Li, T. Li, and S. Venkatasubramanian. t-closeness: Privacy beyond k-anonymity and l-diversity. In *Proceedings of the 23rd International Conference on Data Engineering (ICDE)*, pages 106–115, 2007.

[8] K. Liu and E. Terzi. Towards identity anonymization on graphs. In J. T.-L. Wang, editor, *SIGMOD Conference*, pages 93–106. ACM, 2008.

[9] A. Machanavajjhala, J. Gehrke, D. Kifer, and M. Venkitasubramaniam. l-diversity: Privacy beyond k-anonymity. In *Proceedings of the International Conference on Data Engineering (ICDE)*, 2006.

[10] S. Milgram. The small world problem. *Psychology Today*, 1(1):60–67, May 1967.

[11] A. W. Moore and D. Pelleg. X-means: Extending k-means with efficient estimation of the number of clusters. In *Proceedings of the Seventeenth International Conference on Machine Learning*, pages 727–734. Morgan Kaufmann, 2000.

[12] A. Narayanan and V. Shmatikov. Robust de-anonymization of large sparse datasets. In *IEEE Symposium on Security and Privacy*, pages 111–125. IEEE Computer Society, 2008.

[13] M. E. Nergiz and C. Clifton. Thoughts on k-anonymization. *Data Knowl. Eng.*, 63(3):622–645, 2007.

[14] M. E. Nergiz, C. Clifton, and A. E. Nergiz. Multirelational k-anonymity. In *Proceedings of the 23rd International Conference on Data Engineering (ICDE)*, pages 1417–1421, 2007.

[15] Netflix Prize. `http://www.netflixprize.com`.

[16] P. Samarati and L. Sweeney. Generalizing data to provide anonymity when disclosing information (abstract). In *PODS*, page 188. ACM Press, 1998.

[17] R. Stein. Social networks' sway may be underestimated. Washington Post, May 26, 2008.

[18] L. Sweeney. k-Anonymity, a model for protecting privacy. *International Journal on Uncertainty, Fuzziness and Knowledge-based Systems*, 10(5):557 – 570, 2002.

[19] R. C.-W. Wong, J. Li, A. W.-C. Fu, and K. Wang. (α, k)-anonymity: an enhanced k-anonymity model for privacy preserving data publishing. In T. Eliassi-Rad, L. H. Ungar, M. Craven, and D. Gunopulos, editors, *KDD*, pages 754–759. ACM, 2006.

[20] V. N. Zemlyachenko, N. M. Korneenko, and R. I. Tyshkevich. Graph isomorphism problem. *Journal of Mathematical Sciences*, 29(4):1426Ű1481, May 1985.

[21] B. Zhou and J. Pei. Preserving privacy in social networks against neighborhood attacks. In *Proceedings of the 24th International Conference on Data Engineering (ICDE)*, pages 506–515, 2008.

APPENDIX

A. EXAMPLE OF MODE-BASED CLUSTER CENTER

Vertex	Vertex's Degree	1-Hop Neighbors' Degrees
v_1	2	{3,3}
v_2	3	{5,2,2}
v_3	2	{3,3}
v_4	3	{5,3,2}
v_5	5	{3,3,3,3,3}
v_6	3	{5,3,2}
v_7	3	{5,3,3}
v_8	3	{5,3,3}

Table 1: An example cluster with degrees of nodes and their neighbors.

Table 1 gives an example of our mode-based cluster center generation algorithm. We first compute the cluster center as having a degree of 3, the rounded average degree of vertices in the cluster. The three (virtual) neighbors' degrees are {5,3,3}.

B. PROOFS OF THEOREMS

Proof of Theorem 3.1: Denote the number of undersized clusters by X. The union-split algorithm terminates when there is no undersized clusters, i.e., $X = 0$. At each iteration, three cases can happen and X decreases in all three cases.

1. Two undersized clusters are unioned into one undersized cluster, then $X = X - 1$.
2. Two undersized clusters are unioned into one cluster whose size is $\geq k$, then $X = X - 2$.
3. One undersized cluster is unioned with a cluster whose size is $\geq k$, then $X = X - 1$.

Therefore, each iteration strictly reduces the number of undersized clusters and the algorithm converges in a finite number of iterations. □

Proof of Theorem 3.2:

Let $n = |V(G)|$, $d = avg - deg(G)$, $k =$ minimum allowable cluster size. Let m denote the number of clusters in the current partition. Note $m \leq n/k$. Let $\kappa(c)$ denote the number of points in cluster c. Let κ denote the maximum number of points in any cluster at any time. In the union-split algorithm, $\kappa \leq 3k-2$. Let μ denote the distance metric being used. A cluster is small if it contains $< k$ points. A cluster is large if it contains $\geq 2k$ points.

We will denote by α_G^μ the maximum time to calculate the distance between any two points in the graph G using distance metric μ. We denote by β_G^μ the maximum time it takes to calculate the center of a cluster of $\leq \kappa$ points in G using distance metric μ.

In the union-split algorithm, we maintain two dynamic data structures. The first is the Inter-Cluster Distance Table, which records the distance between every pair of clusters, and is thus an $m \times m$ matrix. The second is the Nearest Cluster Heap List, which for each cluster c stores a heap containing all other clusters, prioritized by their distance from c. We will use this during the union-split algorithm to find the nearest cluster to c. At the beginning of the algorithm we initialize these data structures, which takes $O(m^2 \cdot \alpha_G^\mu)$ and $O(m^2)$ time, respectively, using the linear-time heap-building algorithm.

Choosing which cluster to union by iterating through the current clusters and finding the small cluster with the smallest value at the top of its heap takes $O(m)$ time. After unioning those two clusters, which takes $O(\kappa) = O(k)$ time, we calculate the new center and then update the tables. Calculating the new center takes $O(\beta_G^\mu)$ time. In the Inter-Cluster Distance Table, we must nullify the distances for the rows and columns corresponding to the unioned clusters, and replace it with newly calculated distances from all clusters to the new unioned cluster. This takes time $O(m \cdot \alpha_G^\mu)$. For each modified entry, we update that value in the corresponding heap, which takes $O(m \log m)$ time total. Finally, we create a new heap for the new unioned cluster in time $O(m)$. The total run-time for these operations is thus $O(m \log m + m \cdot \alpha_G^\mu + k + \beta_G^\mu$. Finding the two points in the cluster farthest from each other requires calculating all the pairwise distances, which takes $O(k^2 \cdot \alpha_G^\mu)$ time. The total run-time of step 2b is $O(m \log m + m \cdot \alpha_G^\mu + k^2 \cdot \alpha_G^\mu + \beta_G^\mu$.

We bound the number of iterations of the union-split algorithm by keeping track of the number of small clusters. Initially there are n clusters, and every cluster is small. During every iteration, the number of small clusters decreases by at least one when the union is performed. Since splitting only results in clusters of size $\geq k$, no new small clusters are created. Therefore the algorithm terminates in at most n iterations. Assuming α_G^μ, β_G^μ, and k to be application-specific constants, we conclude that the total run-time of the union-split algorithm is $O(m^2 \log m)$ and thus $O(n^2 \log n)$.