Class-based graph anonymization for social network data

Smriti Bhagat
Rutgers University
smbhagat@cs.rutgers.edu

Graham Cormode
AT&T Labs–Research
graham@research.att.com

Balachander Krishnamurthy
AT&T Labs–Research
bala@research.att.com

Divesh Srivastava
AT&T Labs–Research
divesh@research.att.com

ABSTRACT

The recent rise in popularity of social networks, such as Facebook and MySpace, has created large quantities of data about interactions within these networks. Such data contains many private details about individuals so anonymization is required prior to attempts to make the data more widely available for scientific research. Prior work has considered simple graph data to be anonymized by removing all non-graph information and adding or deleting some edges. Since social network data is richer in details about the users and their interactions, loss of details due to anonymization limits the possibility for analysis. We present a new set of techniques for anonymizing social network data based on grouping the entities into classes, and masking the mapping between entities and the nodes that represent them in the anonymized graph. Our techniques allow queries over the rich data to be evaluated with high accuracy while guaranteeing resilience to certain types of attack. To prevent inference of interactions, we rely on a critical "safety condition" when forming these classes. We demonstrate utility via empirical data from social networking settings. We give examples of complex queries that may be posed and show that they can be answered over the anonymized data efficiently and accurately.

1. INTRODUCTION

Many datasets are most naturally represented as graph structures, with a variety of types of link connecting sets of entities in the graph. An important and natural example of this is presented by Online Social Networks (OSNs), which allow users to identify other users as "friends", exchange messages and comments, post and subscribe to blogs, and play games among themselves. OSNs have over half a billion active users, with some OSNs exceeding 100 million members. There are many specialized OSNs catering to professionals, seniors, writers and students, amongst many others. There are many analytical and sociological questions that can be answered using data encoded in these systems, so it is natural to want to share the data with researchers. However, the raw data is particularly sensitive: it contains personal details entered by the users, and sensitive connections between them which are not public and should not be revealed. This leads us to study how to effectively *anonymize* so as to guarantee privacy of data subjects while maximizing the value ("utility") of the resulting anonymized data.

The problem of data anonymization has been the focus of much study in recent years. The initial focus was on anonymizing tabular data, via *k*-anonymization and subsequent variations (see Section 6). Naively applying such techniques does not yield useful results on graph structured data, and so new methods have been proposed for this problem. Recent work has mostly concentrated on the case where the data can be represented as a simple graph of nodes and edges, and considered how to prevent the inference of connections between individuals (represented by nodes).

However, real data sources are typically much richer than this. Taking OSNs as a motivating example, the main entities in the data are individuals who create profiles for themselves. These profiles can list lots of demographic information, such as age, sex and location, as well as other personal data—political and religious preferences, relationship status, favorite music, books, destinations and cuisines. Between users in an OSN, there are many different kinds of interactions. A common interaction is for a pair of people to indicate that they are "friends": this allows each to see more information about the other, and to receive news of updates. Messages can be exchanged between friends via internal email or instant messaging. Interactions can also involve more than just two participants: messages can be sent to several people; games can be played between several players; many users can subscribe to blogs; or larger groups can be formed for almost any purpose. Such complex sets of interactions cannot be easily represented by simple graphs of nodes and edges without further information. We refer to the connections formed in the social networks as "rich interaction graphs" to emphasize that they contain this rich set of data, based on the interactions between different entities.

Example Queries on a Social Network. Dealing with data from OSNs brings its own challenges. To understand the utility of this data, we outline some of the queries which are of interest to answer:

1. How many users are there in specific subpopulations, e.g., in age ranges, locations, with certain political viewpoints etc.?
2. What are the patterns of interaction and friendship, and which subpopulations are interacting? What is the amount and frequency of interaction? When is it occurring (time of day, day of week, month of year)?
3. Can the graph of interactions be partitioned with small cuts? E.g., are there few links between users from different continents, or with different political views? Can the graph structure be characterized into collections of sub-graph types and do the subgraphs have identifiable properties?

VLDB '09, August 24-28, 2009, Lyon, France

4. How are the interaction patterns changing/growing over time? How is the distribution of interaction between subpopulations shifting over time?
5. What can be learnt about the use of applications (such as games) in the network? How does their popularity spread over time and is this correlated with friendship links?

These queries can be answered exactly on the original data, but there are too many possible such queries and variations for the data owner to precompute and release all possible answers. The answers to so many queries could leak sensitive data about individuals in the data set [1]. Moreover, it is not possible to anticipate all possible queries which could be of interest to analysts. Instead, the goal should be to design anonymization techniques so that these queries, and other similar ones, can be answered accurately on the resulting published anonymized data set. Observe that these queries are not simply about properties of the entities in the data, or simply about the pattern of the link structure in the graph, but rather on their combination. Thus it is important for the anonymization to mask the associations between entities and their interactions (for privacy) but not to obscure them entirely, so that such queries can be answered with some degree of accuracy. Our experimental study evaluates several instances of such queries over anonymized data. When the query relies on large numbers of individuals and interactions, then it can be answered with high accuracy. However, when it is highly selective and involves only a very small number it is answered with low accuracy—as one would expect, since an accurate answer could compromise privacy.

1.1 Our Contributions

Based on the above motivations, we address the problem of finding anonymization techniques for rich interaction graphs which represent Online Social Networks (OSNs).

- We adopt a flexible representation of rich interaction graphs which is capable of encoding multiple types of interactions between entities, including interactions which can involve large numbers of participants (not just pairs). We describe two types of anonymization techniques for such data, both based on partitioning the original entities into *classes*, and then analyze how the ability of an attacker to infer additional information is minimized, or is zero, depending on the amount of background knowledge available to them.
- We present the "label list" approach, which allocates a list of labels to each node in the graph, among which lies its true label (Section 3). That is, each node in the graph gets a list of possible identifiers, including its true identifier. We show how these lists can be structured to ensure that the true identity cannot be inferred, and provide a "safety condition" to ensure that the pattern of links between classes does not leak information.
- We describe the "partitioning" approach, which partitions the entities into classes, and describes the number of interactions at the level of classes, rather than nodes (Section 4). This method is resilient to attackers with greater amounts of background information, but has lower utility since less graph structure is now revealed.
- We discuss how to answer queries over the anonymized data and perform experiments on real social network data, showing the impact of these anonymization techniques on the ability to answer such queries as those described above (Section 5). We observe that for many queries, it is possible to compute accurate answers, while giving the privacy guarantees described.

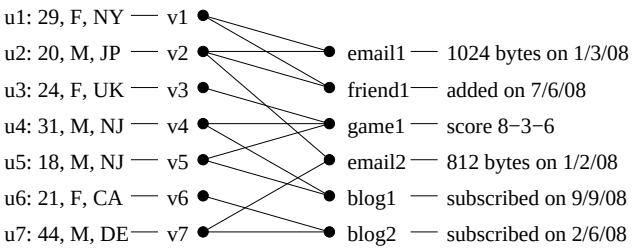


Figure 1: Interaction graph example

2. PRELIMINARIES

2.1 Interaction Graphs.

A rich interaction graph G encodes a variety of interactions between a set of entities V. In the case of an online social network (OSN), V can represent the members of the network. The interactions between them can be, for instance, that an email or IM was sent between a pair, a game was played among four players, or a large group declared their support for a political candidate. These interactions can be represented by a hypergraph, where each hyperedge consists of the set of entities involved in that interaction. This captures the special case of (directed or undirected) graphs, which are hypergraphs in which each hyperedge is constrained to link exactly two nodes.

We choose to represent such rich interaction graphs as bipartite graphs over the sets V and I. Each node in I corresponds to an interaction between a subset of entities from V: an edge $(v \in V, i \in I)$ indicates that the entity represented by node v participates in interaction i. Each entity (corresponding to a graph node) has an identity (such as a user id) and a set of properties. For example, user properties in an OSN include demographic information (such as geographic location, sex, date of birth) and other data (e.g., OSN join date). Each interaction between two or more entities also has an identity and a set of properties. For example, each "friend" relation includes static properties (creation date) and could also have dynamic properties (number of times the friends communicate). We will ensure that each piece of information appears on one "side" of the interaction graph only: if two entities that are friends each record their location, then we do not allow the "friendship" interaction to also record the location of the friends—this could aid an attacker in matching up interactions with entities (see also the discussion on temporal attributes in Section 7).

We show an example in Figure 1. Here, the entities (users and interactions) are linked to their properties (age, sex, location for users, denoted $u_1 \ldots u_7$; and other relevant properties for the interactions). The users engage in different interactions in various combinations: some exchange email messages, establish a "friendship" relation, subscribe to each others' blogs, and so on.

2.2 Anonymization and Utility Requirements.

Let G be an interaction graph over sets V, I and edges E. Our goal is to produce an anonymized version of G, G', say, so that G' retains many properties of G while limiting the amount of information that is revealed. This should hold even under attack from an adversary who knows some limited information about the original graph: given G' it should not be possible to use some (partial) knowledge of G to infer additional linkages between other nodes. Exactly which properties should be protected and which are preserved will depend on the application, and our assumptions about

the other information[1] that is known to an attacker seeking to break the privacy. But there is an inherent tradeoff between the requirements of anonymization and utility: privacy entails removing or masking information in the data, thus potentially reducing the value of the anonymized data.

We now distinguish between a node in the graph, $v \in V$, and the corresponding entity $x \in X$ (in the unanonymized data, these are interchangable). Each entity x can have a number of attributes, such as age, location, and a unique identifier. We write $x(v)$ to denote the identifier, or "true label" of node v. In the unanonymized graph G, full information is published, so $x(v)$ is presented exactly for every node; in the unlabeled version, the mapping between entities (labels or identifiers) and nodes is completely hidden, so nothing is revealed about $x(v)$. We examine settings where information about the mapping $x(v)$ is partially revealed: this will allow non-trivial queries to be computed over the data without exposing the complete data.

For instance, node v_1 in the graph in Figure 1 corresponds to an entity with attributes *29, F, NY*. In this graph, the value of $x(v_1)$ is revealed precisely; in later examples, such the one in Figure 2(a), only partial information is revealed about the mapping. The algorithm to generate the anonymized data is assumed to be public. The privacy conditions should still hold with this assumption. Our focus is on the privacy of the entity-entity interactions, so we do not study the privacy of the entity attributes. Either these are public, or they can be anonymized up to an appropriate level via standard k-anonymity [16, 17] or permutation-based methods [20, 21].

Utility is judged based on the quality with which various queries can be answered on the anonymized graph. Clearly, some queries cannot be answered with high accuracy without compromising privacy. For example, for a query that is so specific that it identifies two unique entities and asks for the graph distance between them, an accurate answer would reveal whether they are directly linked. So our goal is to give accurate answers on queries that are not privacy revealing, and tolerate less accurate answers in other cases. We evaluate the utility of our solutions empirically by measuring the accuracy with which various queries of the type identified in Section 1 can be answered.

We consider privacy requirements related to an attacker being able to learn interactions between entities. Some example requirements could include that: the attacker should not be able to learn any information about interactions beyond what they already know; or the attacker may learn about some types of interaction but not others; or the attacker may learn about interactions between pairs of individuals who are known to the attacker, but not about any interactions involving others (corresponding to the case in some OSNs such as LinkedIn where users can see the connections between their friends). In general we are only concerned with limiting positive inferences, such as determining if two users did send a message between them. It is typically reasonable to allow some negative inferences, such as to learn that two users have never communicated. This is by analogy with prior work on tabular data: for example, in an anonymized medical database, it is not possible for an attacker to determine which of a set of diseases a particular individual suffers from (positive inference), but they can rule out certain diseases (negative inference)[2].

[1]We use "background knowledge" to refer to any information known to the attacker that is pertinent to the anonymized data, and consider different forms of background knowledge in the analysis.

[2]More can be done to hide the presence of an individual in a table, relative to a fixed table of "public knowledge" about individuals [15].

3. ANONYMIZATION VIA LABEL LISTS

We propose to anonymize interaction graphs using "label lists": the identifier of each node in the graph is replaced by a list of possible identifiers (or labels). This reveals partial information about the mapping from nodes to their true identifiers, and the links between them. We show that a carefully chosen labelling ensures that an observer has low probability of guessing the true mapping. Formally, in the *label list* approach, we provide a *list* of possible labels for each node in V, among which the true label is guaranteed to lie. Conceptually, this is related to permutation-based methods for anonymizing tabular data, which can be thought of as providing a short list of possible senstive attributes for each record [20].

Definition 1. The output of a label list anonymization is a bipartite graph G' on vertices V and the original set of interactions I that is isomorphic to the input graph G, so that the (structural) properties of the unlabeled graph are unaltered. The output also includes a function l from V to $\mathscr{P}(X)$ (the powerset of entities X), so that $l(v)$ is the list (technically, a set) of possible labels of v. We insist that the true label of v is included in its list, so that $x(v) \in l(v)$. □

Thus, the unanonymized graph is given by setting $l(v) = \{x(v)\}$, and the unlabeled graph is given by setting $l(v) = X$. In the remainder of this section, we describe a method to build label list anonymizations. First, Section 3.1 shows that simply picking unstructured lists is open to attacks. Instead, we propose more structured ways to generate lists, based on dividing the nodes into smaller groups, and considering each group independently. Section 3.2.1 describes how to generate a set of lists for the group; these are then assigned to nodes, as described in Section 3.2.2. We then analyze the security of this anonymization in Section 3.3.

3.1 Arbitrary List Approach

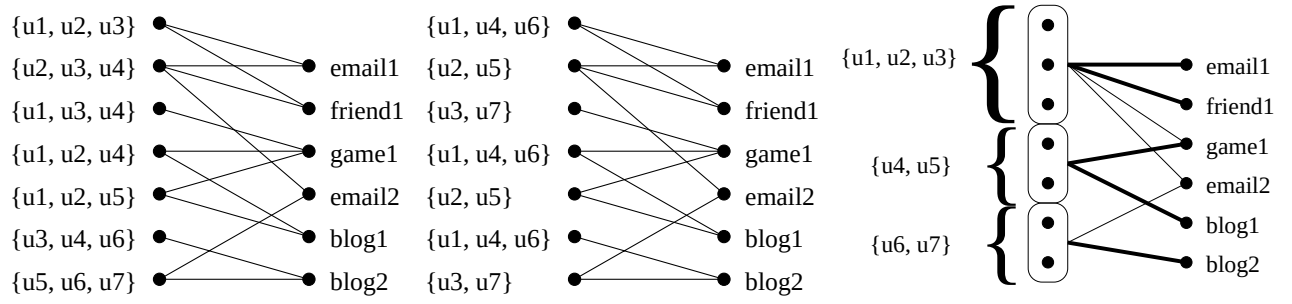
The most general case is when $l(v)$ is an arbitrary list of labels. An example is shown in Figure 2(a), where each list has three entries. Here, the first node (corresponding to v_1 in Figure 1) has the label list $\{u_1, u_2, u_3\}$. Its true label, $x(v_1) = u_1$, is included in the list. Nodes in I are labeled only with their true identity. On first glance, it might seem that given such a set of arbitrary lists, it is impossible for an adversary to recover the true label associated with each entity. However, there are cases which do leak information, motivating us to allow only a subclass of lists, with guaranteed security properties.

Consider again the example in Figure 2(a). The label u_7 only appears for the last node, revealing its identity. For the first four nodes, only the four labels $\{u_1, u_2, u_3, u_4\}$ are used in various permutations. Therefore, since every node corresponds to a unique entity, these identities must belong to the first four nodes. In particular, they cannot be the identities of the fifth or sixth node. Therefore, the fifth node must be u_5 and the sixth node u_6, and so one can deduce that u_6 and u_7 share the blog_2 interaction.

With these examples, it is possible to identify shortcomings in the choice of label lists, and attempt to rule them out. In this example, it seems desirable to insist that each list should contain at least k labels, and each label should appear about k times in the lists, for some parameter k. But a piecemeal approach of adding rules to "patch" weaknesses is not sufficient to show security. Instead, we show that anonymizations using more restricted classes of lists have stronger properties to foil inference attacks.

3.2 Uniform List Approach

We propose an approach to generate sets of lists which are more structured; as a consequence, they avoid "static" attacks, and retain privacy in the face of attacks based on background knowledge.



(a) Arbitrary label lists example (b) Full pattern example (c) Partitioning approach (double links in bold)

Figure 2: Examples of different anonymization approaches

Algorithm 1: DIVIDENODES(V,E)

```
1 SORT(V);
2 for v ∈ V do
3     flag ← true;
4     for class c do
5         if SAFETYCONDITION(c,v) and SIZE(c) < m then
6             INSERT(c,v);
7             flag ← false; break;
8     if flag then INSERT(CREATENEWCLASS(),v,E);
```

First, the nodes V are divided into "classes" of size m, for a parameter m. From the set of nodes, a set of symmetric lists is built for each class, based on a second parameter $k \leq m$. The lists are built deterministically from the set of nodes in the class; it then remains to assign a list to each node so that an attacker has a low probability of guessing the correct label.

3.2.1 Dividing the nodes into classes.

To guarantee privacy, we must ensure that there is sufficient *diversity* in the interactions of nodes in the same class. For example, in Figure 1, suppose entities u_1 and u_2 are placed together in a class of size 2. Then it is clear that u_1 and u_2 were friends and emailed each other, even without knowing exactly which node in the graph corresponds to u_1 and which corresponds to u_2. This inference is possible because the choice of groups made the subgraph involving these nodes *dense*, which implied that there must be a link. The safety property ensures that such inferences are not possible; and more strongly, that even if additional information is learned, certain properties still hold on the remainder of the graph. So we introduce the following definition:

Definition 2. A division of nodes V into classes satisfies the *Class Safety* property if for any node $v \in V$, v participates in interactions with at most one node in any class $S \subset V$. That is,

$$\forall \{v,i\},\{w,i\},\{v,j\},\{z,j\} \in E : w \in S \wedge z \in S \Rightarrow z = w. \quad \square$$

This property holds in the example shown in Figure 2(b). A consequence of this requirement is that if S is the class of v itself, then v can have no interaction with any other node in the same class; i.e.,

$$\forall \{v,i\},\{w,i\} \in E : v \in S \wedge w \in S \Rightarrow v = w.$$

The definition allows two nodes to share multiple interactions (e.g., to participate in a friendship interaction and an email interaction). But it prohibits the case where an entity has multiple friends in the same class. This limits when it is possible to satisfy class safety: for instance, if there is a single entity which has interactions with every other entity, it is not possible to achieve class safety for any $m > 1$. Similarly, if the fraction of pairs of entities which are linked by some interaction (out of the $|V|^2$ possible pairs) exceeds $1/m$, it is not possible for the safety condition to hold. But in real social networks a user typically interacts with only a small fraction of other entities out of the millions of possibilities.

For privacy, we require a solution which observes the safety condition. This leaves room to optimize for utility. One approach is to define a simple utility metric, and try to find an optimal solution relative to this objective (such as maximizing the similarity of entities in the same class). However, since we evaluate utility based on the quality of query answering, such effort is not guaranteed to bring reward. Instead, we use heuristic methods to divide nodes into classes, and compare their utility empirically.

The problem is simplified because the safety condition only restricts the interactions which connect nodes within and across classes: it is based solely on the graph structure linking nodes, and not on any property of the labels of the corresponding entities. So the process can focus on partitioning the nodes into classes of size (at least) m without yet considering how to generate the label lists within these classes and assign them to nodes. A simple greedy approach is illustrated in Algorithm 1. Here, we take each node v in turn, and insert it in the first class that has fewer than m members, provided that performing this insertion would not violate the safety condition (lines 2–7). This can be checked by ensuring for each node that participates in an interaction with v that it does not participate in an interaction with a node already in the class under consideration. If no class can be found which satisfies this condition, or all classes defined have at least m members, then a new class containing only v can be started (line 8); trivially, this class must satisfy the safety condition. The SAFETYCONDITION test can be implemented efficiently by maintaining for each class a list of all nodes which have an interaction with any member of the class. It is safe to insert v into a class if neither v nor any w that shares an interaction with v is present in the list.

LEMMA 1. *The cost of Algorithm 1 is at most* $O(|V||E|\log|V|)$.

PROOF. There can be at most $O(|V|)$ classes created by the algorithm. In evaluating the SAFETYCONDITION test, we have to test a node v and all its neighbors against a list for each class. These tests can be implemented in time $O(\log|V|)$ using standard data structures. Over all nodes v, there are a total of $O(|E|)$ nodes and neighbors, hence a total of $O(|V||E|)$ tests. Creating or updating the list for the class that V is placed in also takes $O(\log|V|)$ time for each neighbor, but this cost is dominated by the (worst case) cost

of testing the SAFETY CONDITION in each potential class. □

It is possible that this heuristic can fail to find a solution when there is a dense pattern of interconnections. But in our experiments this approach still has many degrees of freedom, which can be used to improve the utility of the resulting anonymization. Queries which involve selections on entity attributes (say, selecting users located in Japan) will be unsure exactly which nodes these correspond to. But when entities in a class have the same value on this attribute, the uncertainty in the query answer is reduced, since either all nodes in the class are selected by the query, or none of them are. Such groupings can be created in the anonymized data, subject to satisfying the safety condition. Given a "workload" describing which attributes are seen as most important for querying (say, location is first, followed by age), the input can be sorted under this ordering of attributes, and the above greedy partitioning performed on the resulting list of entities. This sorting step is indicated in line 1 of Algorithm 1. This aims to place nodes that are adjacent in the ordering in the same or nearby classes, unless this would violate safety. Because the safety condition is respected, the privacy conditions still hold: nodes with very similar properties will have to be put in different classes if there are interactions between them. However, such anonymizations may yet be vulnerable to "minimality attacks" [19]. Note that "structural" attributes, e.g., total degree of the nodes, or number of emails sent, can also be incorporated in this scheme for cases that anticipate many queries based on such graph features. We evaluate the effect of these optimizations on utility for different query types in our empirical study.

3.2.2 *Generating and assigning (k,m)-Uniform lists*

After nodes are divided into classes of size m, we generate lists of labels to attach to graph nodes in each class drawn from the entities present in that class. We define a symmetric method of generating lists for a group of nodes based on a parameter k and a "pattern" p.

Definition 3. Given a class of m entities C_j, a collection of m *label lists* is formed based on an integer "pattern" $p = \{p_0, p_1 \dots p_{k-1}\}$, which is a subset of $\{0 \dots m-1\}$ of size exactly k. The label lists generated from p and $0 \leq i < m$ for entities labeled $u_0 \dots u_{m-1}$ are:

$$\text{list}(p,i) = \{u_{i+p_0 \mod m}, u_{i+p_1 \mod m}, \dots u_{i+p_{k-1} \mod m}\}. \quad \square$$

This definition is chosen to be very symmetric: essentially, there is a single pattern which is cyclically shifted to generate the lists. This symmetry is key to proving security properties. After relabelling, we can assume that the pattern p includes 0.

Uniform List Example. Given entities $u_0, u_1, u_2, u_3, u_4, u_5, u_6$ and the pattern $0, 1, 3$, we form label lists to assign to nodes as:

$$\begin{array}{llll} \{u_0,u_1,u_3\} & \{u_1,u_2,u_4\} & \{u_2,u_3,u_5\} & \{u_3,u_4,u_6\} \\ \{u_4,u_5,u_0\} & \{u_5,u_6,u_1\} & \{u_6,u_0,u_2\} & \end{array} \quad \square$$

We identify two special cases of uniform lists:

Prefix patterns. A *prefix pattern* occurs when the pattern $p = \{0,1,2,\dots k-1\}$. These have an important symmetric structure which aids the subsequent analysis.

Full pattern. In the *full pattern* case $k = m$ and so the only possible pattern is $p = \{0,1,2,\dots m-1\}$. In this case, each label list in a class is identical, and consists of all labels of nodes in that class, similar to the structure designed in [6] for different types of graph. This can also be seen as a special case of a prefix pattern.

An example of the full pattern case, with the labels assigned to nodes, is shown in Figure 2(b). Here, the classes on V are $\{u_1,u_4,u_6\}, \{u_2,u_5\}$ and $\{u_3,u_7\}$.

Subsequently, we use the term (k,m)-uniform list to refer to lists generated over classes of size (at least) m with a pattern of size k. We refer to a (k,k)-uniform list as a *full list* for short, and talk about a *prefix list* to denote the case when a prefix pattern with $k < m$ has been used to generate label lists. The two parameters k and m clearly affect the tradeoff between privacy and utility: a $(1,1)$ uniform list associates each node directly with the corresponding entity, allowing full utility but no privacy; a $(|V|,|V|)$ uniform list associates each node with the list of all possible labels, and represents an extreme (within this model) of minimal utility and maximal privacy. The choice of the parameters k and m will depend on the data and the degree of privacy desired. As shown below, k intuitively corresponds to the size of groups in permutation-based anonymization of tabular data (and to a lesser extent, in k-anonymity): it is the size of a group of entities which are indistinguishable. The parameter m allows a wider range of possible anonymizations to be considered: for a fixed k, any $m \geq k$ can be chosen. A (k,k) anonymization generates $k!$ possible worlds, while (k,m) generate more possible worlds, giving a different privacy/utility trade off: for example, $(3,3)$ list gives 6 possibilities, while a $(3,4)$ list generates 9 possibilities. We investigate the effect of varying settings of m in the experimental evaluation.

After the sets of label lists have been generated, they must then be assigned to nodes. Each node must be assigned a list which includes its true label. Schemes which are completely predictable should be avoided, else an attacker who learns part of the mapping could reverse engineer the remainder. The assignment can be modeled as a matching problem on a bipartite graph with m nodes on each side: m corresponding to entities, and m corresponding to label lists. Each edge connects a node to a label list in which the true identity of that node lies, and the goal is to find a *matching* in this graph: a set of m edges with no common vertices. The problem is to pick an *arbitrary* matching from this graph, which corresponds to assigning the label lists to their matching nodes: if it is not arbitrary, then an attacker knowing the strategy could use this knowlege to break the privacy. A natural search procedure is to pick an arbitrary node, and assign an arbitrary matching list to it, then repeat this procedure after deleting all edges which are incident on the two matched vertices. Note that the task is simpler in the full pattern case: all nodes in the same class are given the same label list, containing the set of all labels in the class.

3.3 Security of Uniform Label Lists

The class safety requirement is chosen to ensure an adversary cannot make inferences about interactions between nodes from the anonymized data alone. This is distinct from notions of k-anonymity, and is more akin to the security that arises from permutations of tabular data [20] in conjunction with diversity requirements [13].

THEOREM 1. *An attacker who observes data published using the (k,m)-uniform list approach and who has no knowledge about the original data can correctly guess which entities participate in an interaction with probability at most $1/k$.*

PROOF. We prove the theorem by demonstrating that each "possible world", W in which an entity v participates in a particular interaction i is matched with at least $k-1$ others where this is not the case. Since each is equally plausible, guessing that any one corresponds to the interaction in the original data would succeed with probability at most $1/k$. This relies on the symmetric structure of (k,m) uniform lists: examples above allowed some assignments to be ruled out, and by a process of elimination, allowed other associations to be inferred.

Consider a class S containing entities $u_0 \dots u_{m-1}$ and labels generated by the pattern $p = \{p_0 \dots p_{k-1}\}$. This labeling is consis-

tent with each node v being given the ith label in its list $l(v)$, for $i = 1 \ldots k$. Each such choice of a single label for each node gives a one-to-one mapping $x(v_j) \leftarrow u_{j+p_i \mod m}$ where each label is picked exactly once. This shows that for each label $u \in l(v)$, there is a possible world where $x(v) \leftarrow u$.

Consider $(v,i) \in E$. The class safety condition ensures it is possible that $x(v) = u$ for any $u \in l(v)$. By the safety condition, there is at most one node in S, the class of v, and hence at most one entity in $l(v)$, which participates in the interaction. Since there is no information associated with entities which allows the symmetry to be broken, $x(v) \leftarrow u$ is consistent $\forall u \in l(v)$. Thus, guessing which entity has the interaction with i succeeds with probability *at most* $1/|l(v)| = 1/k$.[3]

Lastly, observe that the class safety condition limits the number of pairs of nodes between a pair of classes that can participate in interactions. This prevents attacks based on a lack of diversity: if (almost) every pair of nodes between two groups participates in some interaction, then an attacker could guess that some pair were friends and succeed with high probability. Consider the case of nodes $v_1 \in S_1$, $v_2 \in S_2$, where there is some interaction i so that $(v_1,i) \in E$ and $(v_2,i) \in E$. The class safety condition ensures that $S_1 \neq S_2$, and so for every possible world W where $v_1 \leftarrow u_1, v_2 \leftarrow u_2$, there are $k-1$ possible worlds where this is not the case. In particular, each of $v_1 \leftarrow u_{1+p_i \mod m}$ and $v_2 \leftarrow u_{2+p_i \mod m}$ is possible, wherein u_1 and u_2 do not share interaction i. Thus, with no further information, the best strategy is again to guess a possible assignment of labels to nodes, and this succeeds in correctly placing entities in an interaction with probability at most $1/k$. □

This result shows that with no background knowledge, no inference is possible. But the label lists are also secure against certain kinds of background knowledge. We analyze the case when an attacker is able to use their knowledge to identify a small number of nodes with their true identity.

THEOREM 2. *An attacker who observes data published using a full list and is able to use background knowledge to find the true identity of at most r nodes can guess which other entities participate in an interaction with probability at most $1/(k-r)$.*

PROOF. We show the impact on the anonymized data when the true identity $x(v)$ of some node v is known. We show that combining this knowledge with the anonymized data has at least as much security as a full list on classes of size $m-1$ and no background information. By Theorem 1 then, no further information can be deduced without further background knowledge. Inductively, with fewer than r pieces of information of this form, the attacker has at most $1/(m-r)$ probability of correctly guessing interactions.

When $x(v)$ is learned, label lists for nodes v' in the same class S as v are updated as $l'(v') \leftarrow l(v') \backslash x(v)$. Now for each $v' \in S$, $|l'(v')| = m-1$, and $l'(v')$ corresponds to a full list approach on $S \backslash v$. Lastly, we note that the class safety still holds on the reduced data. If the property holds for a class S, then it must also hold for $S \backslash v$: as the property makes statements about nodes in the same class, splitting a class cannot falsify this property. Therefore, the safety property is preserved as classes are partitioned by the revelation of the identity of a node. □

Observe that given the true identity of a node in the graph and data anonymized using the (k,m)-prefix pattern approach, an attacker can see exactly which interactions it was a part of (e.g., how many emails were sent, how many friends are listed), but no further information about interactions is revealed—in particular, without further background information or assumptions, they cannot deduce to whom the emails were sent, or with whom the friendships were made. Under related models of background knowledge, the impact on the security of uniform list anonymizations can also be limited. A further observation is that the analysis is somewhat pessimistic: it analyzes the worst case when all the knowledge relates to entities which the algorithm happens to have grouped together into the same class. Such coincidences seem unlikely in reality, so even an attacker with information about much more than k entities may still be unable to guess the ineractions about other entities with probability higher than $1/(k-1)$. The desired minimum security required thus guides the choice of the value of k (and consequently m). For some applications, it is sufficient to ensure that each entity is classed together with a small number of others, say 5 or 10; in other applications, higher privacy requirements can lead to larger values of k from 20 to 50 or higher.

[3]Without background knowledge, the attacker's best strategy is to guess uniformly over the k possibilities; any other strategy has lower probability of success.

4. PARTITIONING APPROACH

We have studied the strengths of the label list approach. However, an attacker who has complete or near complete information about one node and partial information about other related nodes can combine this with data anonymized by the uniform list approach to infer more about the interactions between those nodes for which partial information is known. A concrete example occurs within a social network, when users know their own interactions, and some properties of the entities that they have interacted with (for example, they can see the age and location of their OSN "friends"). Using the exact knowledge of their own number of interactions, the attacker may be able to identify which node in G' corresponds to their data. Further, they can see which nodes they are connected to via interactions, and potentially identify them. For example, if the attacker v has only one friend in Alaska, and of all the classes containing nodes which share an interaction with v, only one has nodes located in Alaska, then the attacker has found the node corresponding to that friend. The attacker can learn about the interactions of any identified nodes, and in particular, about interactions amongst them (such as which have exchanged email).

To preclude such attacks which leverage greater amounts of background information, we increase the amount of masking of data, at the expense of utility. This leads us to a *partitioning approach*, which partitions the nodes into classes. Instead of releasing the full edge information, only the *number* of edges between (and within) each subset is released. This is similar to the "generalization" technique of Hay *et al.* [8] for simple graphs; a key difference is that we require additional structure of the partitions to ensure attacks are not possible. More context is given in Section 6.

Definition 4. Given a rich interaction graph G, a partition anonymization of G consists of a collection of sets of nodes $\mathscr{C}$ which partition the vertex set V. The partition anonymization is a (weighted) bipartite graph G' on $\mathscr{C}$ and I so that the weight of edge (C,i) is $|\{v \in C | (v,i) \in E\}|$, the number of edges between nodes in class C and interaction i. This is an m-partition if for each $C \in \mathscr{C}$, $|C| \geq m$. □

Figure 2(c) shows an example 2-partition where V is partitioned into $\{u_1,u_2,u_3\}, \{u_4,u_5\}$ and $\{u_6,u_7\}$. In the illustration, thick lines indicate double edges (when there are two edges linking the interaction on the right to members of the class on the left).

Under the partitioning approach, even if an attacker is somehow able to identify which node represents an entity, or a particular interaction, there is still uncertainty about other interactions. Nevertheless, a safety condition is necessary to avoid inference. In fact,

we make use of the *same* safety condition as before (Definition 2). Although the exact connections between nodes and interactions are not revealed by the partition, the safety condition is needed to prevent the attacker using the density of the graph to conclude that entities participate in a particular interaction with high probability. For example, in Figure 2(c), which does not satisfy the condition, an attacker can infer that since u_6 and u_7 are in the same class they alone must share the blog_2 interaction. Likewise, if between two classes of size m there were m^2 friend interactions (and each node pair can participate in at most one friend interaction), then the attacker can infer that there are friend relations between each pair of nodes drawn from the two different classes. The safety condition prevents such inference; more formally:

COROLLARY 1. *An attacker who observes data published using the m-partition approach and who has no background knowledge about the original data can correctly guess which entities participate in an interaction with probability at most* $1/m$.

PROOF. The proof of this corollary is immediate from Theorem 1: any partition which satisfies the safety condition corresponds to a uniform list version of the data, but with strictly less information revealed. Therefore, the impossibility of inference on the (more informative) uniform list version implies that nothing can be inferred on the partition version. □

The impact of background knowledge is further limited in this case.

THEOREM 3. *An attacker with background knowledge about interactions of an entity, modeled as knowing the true identity of some nodes, and the fact that these nodes participate in certain interactions, and data anonymized into an m-partition with the safety condition can correctly guess which entities participate in interactions about which nothing is known with probability at most* $1/m$.

PROOF. Combining the background information with the published data is equivalent to applying the same anonymization to the original data with all the known edges entities removed. This does not give any further information about the remaining unidentified interactions, and so the probability of these being associated with any particular node is uniform across the class. Similarly, if we assume that the attacker knows all participants v that share a particular interaction i this does not determine any other interaction i' involving this pair. Even if there are nodes in the two classes that connect to i' this is consistent with any pair of other members in the classes sharing the interaction. Lastly, by the safety condition, the attacker cannot use the density of interactions to infer anything further about pairs which must be connected, using a similar argument to the label list case. □

In particular, this extends privacy to the case where a user in a social network knows a certain amount of information about themself and about their OSN friends (as mentioned in Section 1.1). Under the partition anonymization, they are unable to use this information to learn anything additional from the anonymized data. This holds even when they interact with a large number of other users, since the safety condition will place each of these in a separate class, preventing further inference. The partition approach is also resilient to attacks based on other information: knowing for example, that an entity has a certain unique degree, as considered in [23, 12], cannot help identify any more information about it. However, this additional resilience comes at the cost of reducing the utility for answering more complex graph queries, as observed in the experimental analysis in Section 5.

5. EXPERIMENTAL STUDY

5.1 Querying Anonymized Data

The result of the anonymization (either via label lists or partitioning) is a graph G' linking nodes to interactions, with a set of possible labels for each node. Given such masked data, an end-user has to perform their desired analysis upon it. Section 1 detailed a set of example queries on social networks, based on selecting subpopulations, and identifying the pattern of interactions between them. We outline approaches to answering such queries on anonymized data:

Sampling Consistent Graphs. A probabilistic approach is for the analyst to randomly sample a graph that is consistent with the anonymized data, and perform the analysis on this graph. That is, for each class, they choose an assignment of nodes to entities consistent with the possible labels (in the style of the methods described in Section 3.2.2). The query can be evaluated over the resulting graph; repeating this several times generates an "expected" answer to the query given the anonymized data. In the full-list and partition cases, sampling a consistent graph takes time linear in the size of the anonymized data.

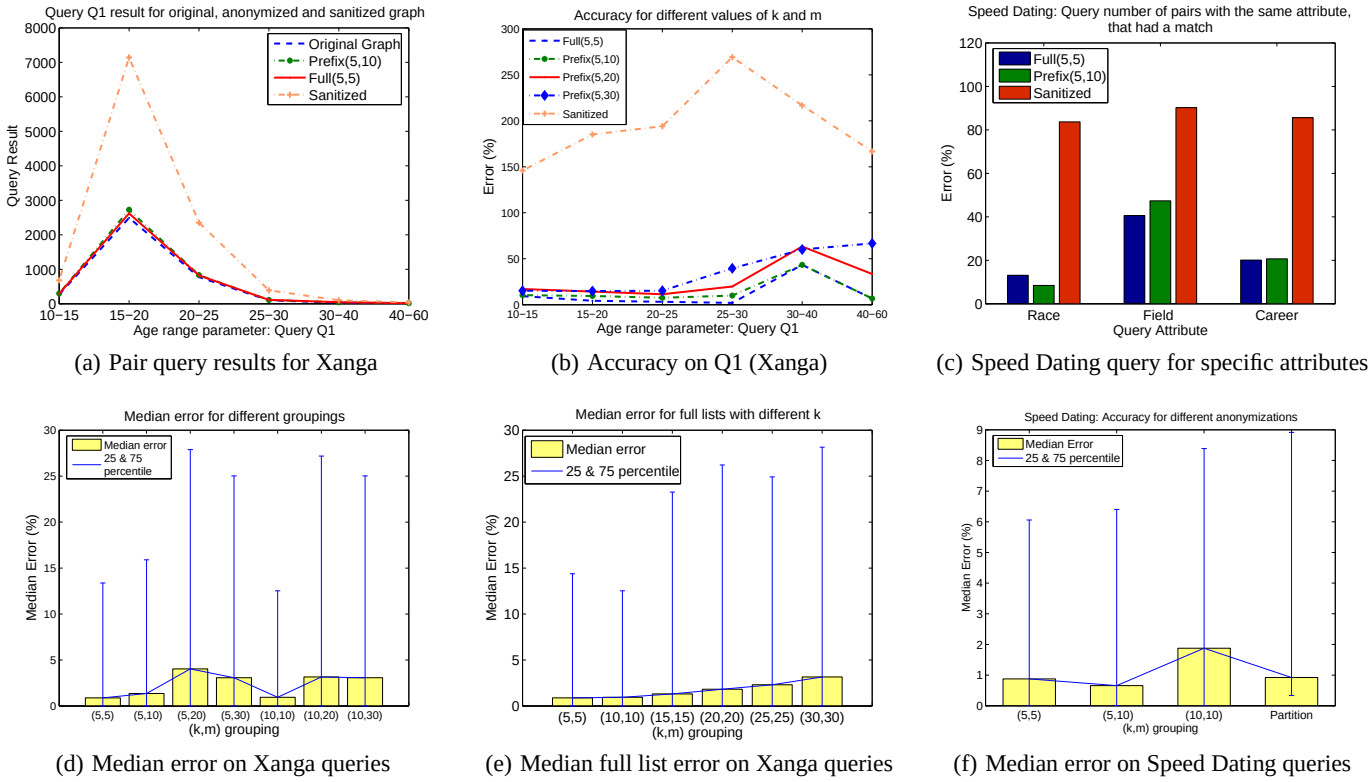
Deterministic Query Bounds. A more costly approach is to search over all possible graphs that are consistent with the anonymized data, and report the range of possible answers to the query. In general, this could be prohibitively expensive, but for many natural queries, the problem can be broken down to consider the result of the query on each class separately, and combine these to get overall bounds. The example query "how many users in the USA have used application X?" can be answered by examining how many users in the USA in each group have used the application. If the group has no users from the USA, or no links to the application in question, then its contribution is zero; similarly, if all users of the group are in the US, or all nodes have an interaction with the application, then the query can be answered exactly for that group.

5.2 Experimental Framework

Our analytical results prove the *anonymity* properties of our schemes. We now present an empirical evaluation of the *utility* of these approaches using two datasets: one from the Xanga social network [3], and the other from a publicly available Speed Dating study conducted by Fisman *et al.* [7]. The two datasets vary appreciably in size and graph structure. The crawled subgraph from the Xanga network consists of about 780K nodes and 3 million edges. Each node in the graph represents a user of Xanga and his corresponding blog (the social network is based primarily around blogging). There are two types of edges between nodes representing the interactions: a subscription (or readership) to another Xanga blog and an explicit friendship relation. Each user has associated profile information, such as the user's location, age, and gender, which comprise attributes of the corresponding node in the graph. All interactions among the users in the dataset are included.

The speed dating dataset has 530 participants, and consists of data about 4150 "dates" arranged between pairs of individuals. For each participant, demographic information was collected (age, race, field of study), and their level of interest in a number of hobbies on a scale of 1 to 10. Each interaction node represents a "date", and records information about whether each participant was positive or negative about their counterpart; we call this a "match" if both were positive. Each individual participated in from 6 to 22 dates.

Queries of interest. Section 1 provides a list of example queries on social networks. These are primarily based on how various subpopulations interact within the social network, and what structures



(a) Pair query results for Xanga (b) Accuracy on Q1 (Xanga) (c) Speed Dating query for specific attributes

(d) Median error on Xanga queries (e) Median full list error on Xanga queries (f) Median error on Speed Dating queries

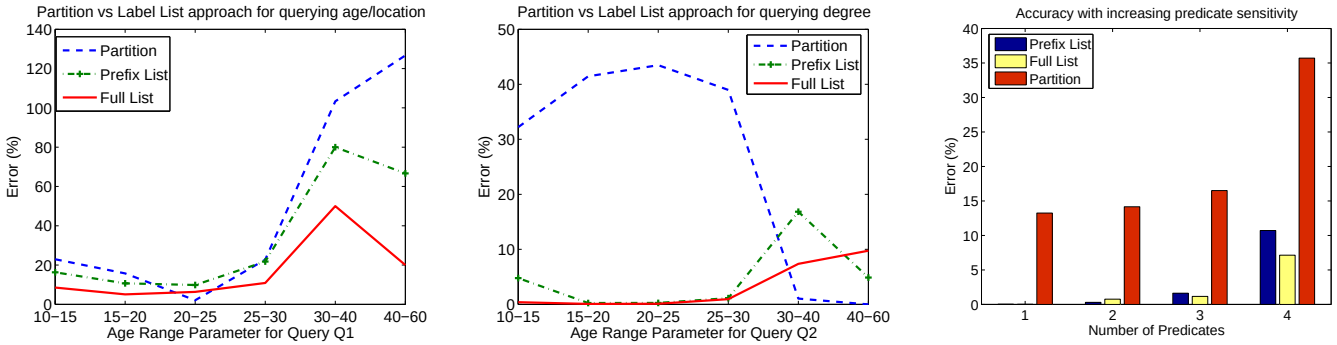
Figure 3: Anonymization of Graphs using Label List Approach

are present in the interaction graph. Queries in bullet 1 can be answered directly from the unlabeled (i.e. sanitized) data, while queries in bullets 4 and 5 require temporal information which is not present in the datasets we study here. So we analyze the utility of the anonymized data over a variety of queries which address the second and third bullets in the earlier list:

- **Pair Queries.** These are single-hop queries of the kind: how many nodes from one subpopulation interact with nodes of another subpopulation. For instance, how many users from US are friends with users from Australia?
- **Trio Queries.** These queries involve two hops in the graph and query for triples such as, how many Americans are friends with Chinese users who are also friends with Australian users?
- **Triangle Queries.** This type counts triangles of individuals (a.k.a. the transitivity property among nodes, or the clustering coefficient), that is, nodes that have neighbor pairs which are connected. For instance, how many Americans are friends with Chinese and Australians who are friends with each other?

Besides working with specific examples of the above query types, we present results on a workload comprised of 100 queries. For the Xanga dataset, the workload consists of a diverse set of queries of the above three types with a variety of constraints on structural properties and node attributes. For each of the pair, trio and triangle queries, we set parameters with varying selectivity for each attribute—age, location (country and continent), gender and degree (number of interactions of a particular type). For instance, we include queries on highly populated age groups (15-25yrs) as well as less populous age ranges (30-60yrs). All query instances mentioned in this section are examples of the queries that form the workload. For the speed dating dataset, we consider a variety of pair and trio queries with attribute selections (there are no triangles in the underlying graph). For instance, "How many participants who love movies (rating>7) date each other?", or "How many Asian participants who dated each other had a *match*?".

We answer queries by sampling 10 consistent graphs, as described in Section 5.1 and averaging the results. The parameters k and m are chosen to provide desired privacy levels. The default parameters for the Xanga dataset for the full list approach are set to $k = m = 10$ and those for the prefix list approach are set as $k = 10, m = 20$, which guarantees individual's privacy with probability at least 90%, similar to previous work [4, 8, 15, 20]. On the smaller speed dating dataset, the parameters chosen are correspondingly smaller: $k = m = 5$ for full lists, and $k = 5, m = 10$ for the prefix list approach. Both anonymization and query answering algorithms scale well: anonymization took less than a minute for the Xanga data on a standard desktop with 1GB RAM, while queries took under a second each. Our code is available at `paul.rutgers.edu/~smbhagat/GraphAnonCode`. We compare the utility of our anonymization approaches by looking at both the absolute error in the query results, and the relative error. The relative error is found by scaling the absolute difference between the true and approximate answers by the true answer, and so can exceed 100%. Comparisons over the workload are performed by computing the median relative error over all the queries in the workload. For these results, the error bars show the 25 and 75 percentile points. We also present utility results over completely anonymized (sanitized) data: here, the details of the entities and the interactions are given, along with a completely unlabeled interaction graph. Query answering on this data is done by evaluating queries after randomly assigning entities to nodes (as in Section 5.1). For the



(a) Query over entity attributes (b) Query over graph structure (c) Queries with variable number of predicates

Figure 4: Anonymization with Partition Approach

speed dating set, we sampled graphs subject to the integrity constraint that only males were paired with females.

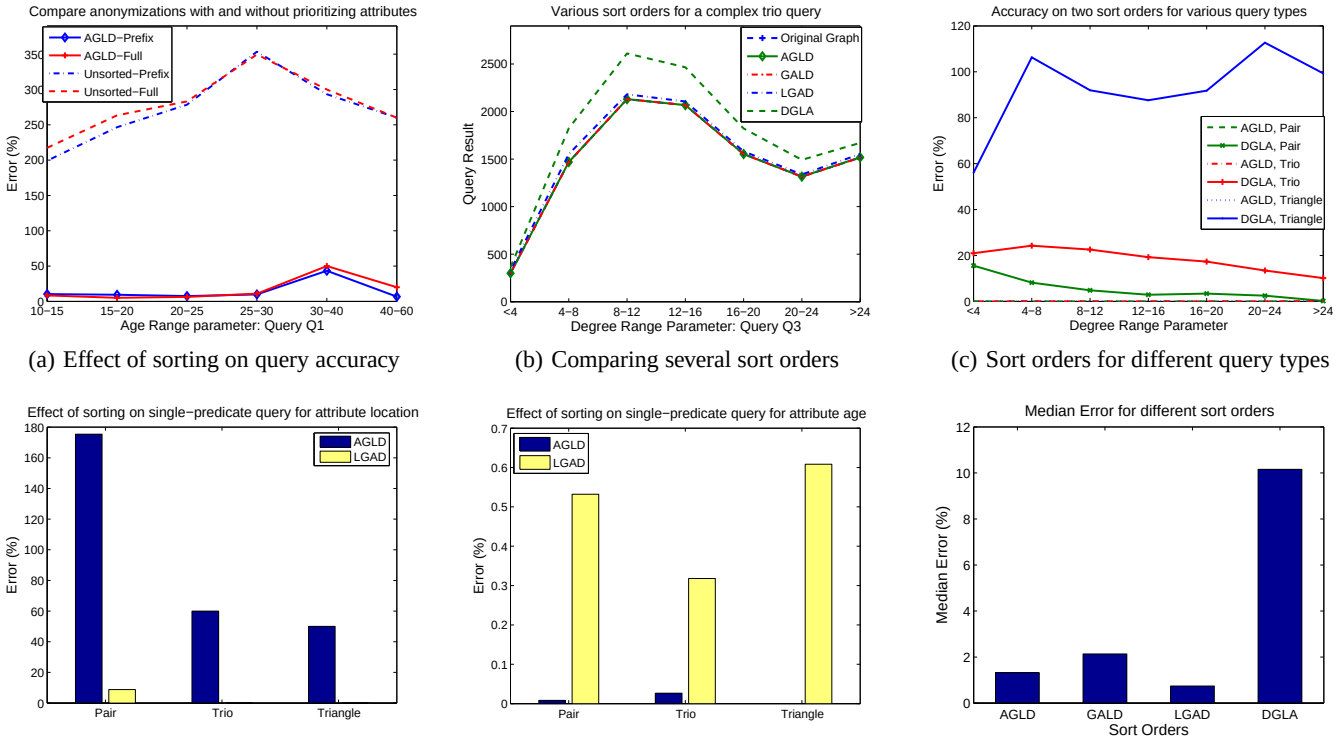
5.3 Experimental Results

Uniform List Anonymization. The accuracy of the uniform list approach was evaluated on several queries, such as the *pair* query Q1: "How many Americans in different age groups are friends with residents of Hong Kong with age less than 20?" on the Xanga dataset. Figure 3(a) shows the query results for prefix and full list approaches, alongside the answers on the completely anonymized (or sanitized) data, and the original unanonymized graph. The figure illustrates similar query results on anonymized and unanonymized data, while those on the sanitized data err significantly. Both prefix and full list anonymizations obtain answers which are close to the true value, even when the query is very selective (e.g., there are very few users with ages in the range 40-60). For the same query Q1, Figure 3(b) shows the accuracy for several values of the parameters m. For fixed values of k, the relative error is appreciably smaller for lower values of m, which correspond to smaller classes. As one would hope, anonymization with smaller classes results in higher utility. The error on sanitized data is at least 3 times higher than that obtained on label list anonymized data. The relative accuracy worsens for age ranges with low support, which we argue is necessary. Since the query touches fewer individuals, requiring a highly accurate answer would compromise privacy.

Figure 3(c) shows results on querying node and edge attributes of the Speed Dating anonymized and sanitized graphs. The query posed is: "How many pairs with the same *attribute* had a match?", where attribute is race, field of study or career goals. The prefix (5,10) list performs slightly better than the full list for the race attribute, while the full (5,5) list performs slightly better for field and career attributes. Our general observation is that while the full list approach more often yields better accuracy, there are datasets and queries where the prefix list approach is better. Based on the design of the lists, the full list works well when the data can be formed into groups of size $m = k$ where all entities behave similarly; prefix lists work well when the data can be formed into groups of size $m > k$, where entities close in the group behave similarly but those on "opposite sides" of the group are less similar.

Figures 3(d) and 3(e) present the median error over the workload of 100 queries of various types on the Xanga dataset. The median relative error for various values of (k,m) tested is in single digits, while that on sanitized data is 184% (not shown in the figure). In conjunction with our other experiments including sanitized data, it seems safe to conclude that there are clear benefits to using these anonymization methods in preference to trivial sanitization, and we do not show any further results with sanitized data. The accuracy for full list anonymization is slightly better than the prefix list approach with the same values of k. Indeed, the general trend in Figure 3(d) is for higher errors with larger m values for the same k. On these data sets, it is better to arrange nodes into groups of size k and assume that all entities behave similarly, rather than put them into larger groups and assume that the similarity varies smoothly within the group. For the full list approach, Figure 3(e) shows a clear trend of increase in error with increasing k $(= m)$. This illustrates the privacy-utility tradeoff, since larger lists correspond to stronger guarantees of privacy. For the Speed Dating dataset, prefix lists perform slightly better than full lists over the Speed Dating dataset (Figure 3(f)). This seems to occur due to the small range of each attribute value (typically 10), and the many combinations of attribute values that are seen. Since the benefits of using prefix lists with $m > k$ are small, and hard to predict, our general experimental conclusion is that the full list approach is likely to be preferred in general, but there are benefits to using prefix lists when the data and typical usage is well understood.

Partition anonymization. Recall that the partition approach does not release edge information between nodes, but releases only the number of edges between groups, say N. To generate a consistent graph, we randomly create N edges between nodes from two connected groups that satisfy the safety condition. In the absence of knowledge of node degrees, the number of edges between a pair of nodes may be very different in a graph sampled from the data anonymized by partitioning. For some simple queries that do not involve node degrees, using data anonymized with the partition approach gives results similar to uniform lists. Figure 3(f) illustrates that the partition approach performs as well as the other uniform list anonymizations over a workload for the Speed Dating graph. However, this data was generated as part of a controlled experiment where the node degrees were carefully managed. For the remainder of this section, we focus on the larger Xanga dataset. Figures 4(a) and 4(b) show the utility of the partition approach on queries Q1 and Q2 respectively. Q1 is as above, and query Q2 in Figure 4(b) is: "How many Americans are connected to users with degree greater than 10?" The utility of the partition approach is similar to the label list approaches when querying for node attributes *only*, as seen in Figure 4(a). For query Q2, which involves node degrees, the partition approach typically has much lower accuracy, as shown in Figure 4(b). The high error for the partition approach occurs on age ranges 10–30 years, each of which covers many individuals, who together have high variation in their degree.



(a) Effect of sorting on query accuracy

(b) Comparing several sort orders

(c) Sort orders for different query types

(d) Queries with Location Predicates Only

(e) Queries with Age Predicates Only

(f) Median error over 100 queries

Figure 5: Analyzing the impact of prioritizing attributes for grouping

The lack of knowledge of the true degrees clearly contributes to the higher error. It achieves better accuracy on age groups with much fewer individuals (30–60 years), and here most nodes have degree 2. Meanwhile, label lists show the greatest variation over groups which are the least populous.

Query Selectivity. The selectivity of a query depends on the query type (number of hops), and the choice of attribute constraints on each node. In the above figures, the choice of the age groups varied the selectivity. In Figure 4(c) we show the effect of changing the number of constraints on attributes of a node. The figure shows the accuracy of *pair* queries on Xanga data with constraints on location, age, gender and degree successively added at each query. The trend is for the error to increase with the number of constraints as the queries become more selective, and hence the number of matching individuals decrease. The utility of full list is somewhat higher than the prefix list approach for queries with high selectivity, while partition is uniformly worse, in line with previous experiments.

Verifying Privacy Guarantees. The analysis of the privacy of the anonymization schemes gives guarantees in terms of the parameters k and m. We verify this by evaluating the accuracy with which we can answer queries that violate the privacy requirements, that is, reveal whether there was an interaction between particular pairs of nodes. We posed the query: "Does node x interact with node y?", which is an instance of a pair query with a very precise selection. In our experiments, we found there was no statistically significant difference between answers for pairs which were connected and those that were not. This indicates that our anonymization does not leak such interaction information.

Prioritizing Attributes for Utility. Section 3.2.1 suggested that queries can be answered more accurately if many classes contain nodes which are identical on the queried attributes; we now study this experimentally. Such classes can be obtained by sorting the nodes based on one or more attribute values before grouping them. We consider a variety of sort orders over the entity attributes and degree of the nodes:

- Age, Gender, Location, Degree (AGLD)
- Gender, Age, Location, Degree (GALD)
- Location, Gender, Age, Degree (LGAD)
- Degree, Gender, Location, Age (DGLA)

The default sort order used in our experiments is AGLD. Figure 5(a) evaluates the benefit of performing an additional sort before grouping on the *pair* query Q1. The figure shows a substantial benefit when using the sorting. We next compare the results when prioritizing different attributes. Figure 5(b) shows the result of a *trio* query Q3: "How many users in the age group 15-20yrs (and varying number of friends) are friends with users in the age group 10-15yrs and also with those in the age group 20-25yrs?" evaluated by grouping on the four different sort orders. Grouping the nodes after sorting with the first three sort orders results in query accuracy of over 99% on query Q3. Figure 5(c) shows the effect of sorting on sample pair, trio and triangle queries involving two attributes, age and degree. The error obtained on this query for sort order AGLD is close to zero. Since the query depends very heavily on age predicates, the sort order DGLA which puts age last has noticeably lower accuracy than orders which rank age higher. The same query gives better results for the sort order DALG (not shown in the figure) since this groups nodes together with similar degree and age. This effect is amplified as the query becomes more complex and hence selects fewer nodes of the graph. For instance, the triangle query Q3 shown in Figure 5(c) is very selective with fewer than

300 users satisfying the query. Consequently, answering the query using anonymized data with a sort order that emphasizes a different set of attributes results in a higher percentage error.

To show the effect of sort order more clearly, we consider queries of the three types (pair, trio, triangle), where each predicate is only over a single attribute. Figure 5(d) shows the results when all predicates are based on location, while Figure 5(e) gives the case when all predicates are based on age. The three queries in Figure 5(e) are: the *pair* query "How many users of age 20 are friends with users of age 21?"; the *trio* query "How many users of age 20 are friends with users of age 21 and with users of age 22?"; and the *triangle* "How many users of age 20 are friends with users of age 21 and age 22 who are also friends?". Similar queries based on locations are used in Figure 5(d): the chosen location query selects about 50 nodes, making the relative error quite high. In Figure 5(e), the error on all three types of age queries is less than 0.1% for the AGLD sorting. Even sorting (using LGAD) with a lower priority on the primary query attribute (age) is effective: the error is less than 1% on all three queries. This is because, in this data set, there are many values of (location, gender, age) which are shared by many users. These fill many classes, meaning that the query can be answered exactly for these classes, and uncertainty only arises for classes with a mixture of attribute values. These plots clearly show that when the sort order matches the attributes specified in the query, the results are substantially improved. When the query selects many entities in the same group, the observed error is close to zero. Thus, to the extent possible, the anonymization should be crafted with the intended usage in mind.

Finally, Figure 5(f) summarizes the accuracy obtained for the four attribute priority orders when evaluated over the full workload of 100 queries. Over this wider variety of queries, the overall accuracy is high, since all the sortings place similar users together in classes. These experiments highlight the fact that choosing a sort order that matches the expected query workload is necessary to optimize for query accuracy.

6. RELATED WORK

The study of techniques to allow safe anonymization of sensitive data has been ongoing for many years. It has a long history in statistical areas, since census, survey and medical data should not reveal personal information about the participants. The work of Sweeney and Samarati which defined the notion of k-anonymity [16, 17] led to much research in the database world. Key subsequent works have included ℓ-diversity [13], (α, k)-anonymity [18], t-closeness [11], (c, k)-safety [14], and anonymization via permutation [20, 21]. These works focused on the core problem of anonymizing a single database table while preserving the utility of natural SQL-style queries; as a consequence, they do not immediately provide useful results for interaction graphs. Prior work shows that applying these methods to appropriately represented graph data retains little utility [6]. Likewise, methods based on cryptography give strong privacy guarantees, but have not been shown to be sufficiently general or scalable for the data publishing scenario we study here.

Our methods can be seen as extending permutation-based approaches to graph-structured data. On tabular data, a permutation divides the rows into classes, and for each class asserts that there is some (unspecified) bijection between the quasi-identifiers and the set of sensitive attributes in the class. This is similar to our notion of classes of nodes, where there is an unspecified bijection between the graph nodes, and the entities (and their attributes). The principal differences are that the use of uniform-patterns describes a different set of possible bijections; and because the graph information requires additional restrictions on which entities can be placed together in the same classes (leading to the class safety condition). A general criticism of anonymization techniques is that sometimes the results offer limited utility gains compared to using data that has been completely sanitized [4]. In our experimental evaluation, we observed that this is not the case for the examples we consider.

Due to the popularity of social networking services such as Facebook, MySpace and LinkedIn, there has been great interest in studying the structures and features of users' interactions, and the implications this has for the transmission of information and ideas. This led many to ask how best to share informative and representative data sets without compromising the privacy of the individuals who intended their details to be shared only with their social network "friends". Privacy issues in existing social networks are surveyed in [10]. In the past few years there has been much study of what can and cannot be achieved in anonymizing social network data or, more generally, graph structured data.

Backstrom *et al.* [2] showed that a powerful attacker with significant background knowledge can learn information about some individuals from an unlabeled graph. In particular, they considered cases when an attacker "plants" enough nodes in the graph, and can link them to legitimate users. If the full link structure is revealed, the "plants" can be reidentified, and the pattern of links between their legitimate neighbors can be recovered. This attack requires legitimate users to "accept" the link requests from the plants before the data is anonymized. A more "passive" attack is when the adversary learns the complete link information of a sufficiently large close-knit group. This can again be located in the graph, revealing the links between neighbors of these group members. In all these scenarios, nothing is learned about individuals not connected to nodes that are compromised by the attacker.

Several recent works try to thwart attackers by modifying the link structure of the graph by adding or removing edges. Liu and Terzi [12] modify the graph by edge additions so that there are at least k nodes with the same degree. Zhou and Pei propose the stronger requirement that each node must have k others with the same neighborhood characteristics, and focus on the case of immediate neighbors, with labels drawn from a hierarchy [23]. Hay *et al.* study the amount of privacy present when an attacker might have background knowledge of (multi-hop) neighborhood information in anonymized graphs when the graph structure is not altered [8]. In a similar setting, Korolova *et al.* [9] study the case when the attacker can "buy" information about the neighborhood of nodes, and analyze the cost necessary to learn about certain individuals.

A criticism of methods which add or remove graph edges is that simply counting the number of such alterations does not correlate with any meaningful measure of loss in utility (similar criticisms apply to tabular data anonymization [4]). In some cases altering a single edge can have global impact on the graph: changing whether a component is connected, or affecting the diameter of the graph. As a result, several more recent works attempt to mask the information in the graph without explicitly adding or removing edges.

Zheleva and Getoor model an attacker as a machine learning algorithm, and try to mask the graph data to limit the ability of such algorithms to correctly predict links [22]. They compare anonymizations based on removing some *types* of edges, and grouping nodes so that only the number of edges between groups is revealed. Hay *et al.* [8] proposed forming nodes into groups and revealing only the number of edges between pairs of groups. This is related to the partitioning method we propose, but is defined only in terms of simple graphs. Our work differs, in that we study richer classes of graphs with node attributes, and we identify the need to limit the density of edges between groups to prevent inference. In prior work, there was no analysis of the residual anonymity following

partial revelation.

Extending this approach, Campan and Truta propose building "clusters" (groups) of nodes, and revealing only the number of edges within a group and between pairs of groups [5]. The nodes have additional properties, which are generalized so that all nodes in the same cluster have the same generalized representation (but only a single type of interaction is allowed). Lastly, Cormode *et al.* studied anonymization of the subclass of bipartite graphs which link two different types of entity, and proposed a permutation-based approach [6]. Our work here differs by studying a much richer class of graphs which can handle many different types of entities and edge types, which allows social network data to be represented, an open problem in [6]. Our techniques are based on many different kinds of grouping, and allow representations that are not possible in prior work.

7. EXTENSIONS

Classes of interactions. Our anonymization methods place entities together in classes to mask their identity. They leave interactions unmasked since the primary objective is to disguise which interactions an entity participates in. It is possible to also create classes of interactions, which mask *their* identity. While we are concerned with the privacy of individuals, in general there is less concern about the privacy of inanimate objects such as single emails or friendship relations per se. However, one can imagine scenarios where correlations between certain interactions should also be masked: in the uniform list approach, it is possible to see which interactions share a common participant, even though the identity of that participant remains hidden.

Directed Graph Representations. For simplicity, the interaction graph has been assumed to be undirected. In general, such graphs contain directional information—that an email was sent from one person to another, or a set of others; or one user is subscribed to the blog of another (a "following" relation). These can be encoded as directed edges in our setting. Directed graphs can be anonymized using our techniques without losing the directionality: our methods give the same guarantees. Including directions may give more chance for an attacker with copious background information to break the privacy, but it remains safe against attackers within the background knowledge bounds described.

Temporal information. Interaction graphs, in particular social networks, are rapidly evolving objects. Our discussion is focused on releasing an anonymized "snapshot" version of the data. This allows many queries to be evaluated, but may limit the possibility for observing how the data changes over time (longitudinal studies). This is not a concern for historical data: temporal data (to indicate that a user joined the network on a certain date, or that a message was sent at a particular time) is treated like any other attribute. Care is needed: e.g., if a member joined only in 2008, then they can not have started a friendship in 2007. This is an instance of the same type of information being present on both sides of the graph at the same time, which was precluded in Section 2. Such inference can be prevented by placing temporal information on one side of the graph only (entities or interactions), or by ensuring that all interactions are feasible for all entities that they could be assigned to.

8. CONCLUDING REMARKS

The problem of publishing sensitive social network data to allow complex ad hoc questions to be answered accurately while preserving privacy of the data subjects remains a challenging one. We have introduced new representations and techniques which allow rich data comprising many different interaction types to be anonymized effectively. Our empirical study validated the privacy-preservation and utility claims of our anonymization schemes over a variety of queries on two different datasets. It remains a challenge to fully understand how to translate the aim of privacy for data subjects into precise quantifications of privacy goals, and to better quantify the utility of a resulting anonymization against an unknown future query workload. It may be profitable to model the inherent trade-off between privacy and utility via information theoretic methods.

Acknowledgements. The authors thank Irina Rozenbaum for the Xanga data set; and the referees for helpful suggestions.

9. REFERENCES

[1] N. Adam and J. C. Wortmann. Security-control methods for statistical databases: a comparative study. *ACM Computing Surveys*, 21(4):515–556, 1989.

[2] L. Backstrom, C. Dwork, and J. Kleinberg. Wherefore are thou R3579X? Anonymized social networks, hiddern patterns and structural steganography. In *WWW*, 2007.

[3] S. Bhagat, G. Cormode, S. Muthukrishnan, I. Rozenbaum, and H. Xue. No blog is an island analyzing connections across information networks. In *Int. Conf. on Weblogs and Social Media*, 2007.

[4] J. Brickell and V. Shmatikov. The cost of privacy: Destruction of data-mining utility in anonymized data publishing. In *ACM KDD*, 2008.

[5] A. Campan and T. M. Truta. A clustering approach for data and structural anonymity in social networks. In *PinKDD*, 2008.

[6] G. Cormode, D. Srivastava, T. Yu, and Q. Zhang. Anonymizing bipartite graph data using safe groupings. In *VLDB*, 2008.

[7] R. Fisman, S. Iyengar, E. Kamenica, and I. Simonson. Gender Differences in Mate Selection: Evidence from a Speed Dating Experiment. *Quarterly Journal of Economics*, 121(2):673–97, 2006.

[8] M. Hay, D. Jensen, G. Miklau, D. Towsley, and P. Weis. Resisting structural reidentification in anonymized social networks. In *VLDB*, 2008.

[9] A. Korolova, R. Motwani, S. Nabar, and Y. Xu. Link privacy in social networks. In *ACM CIKM*, 2008.

[10] B. Krishnamurthy and C. Wills. Characterizing privacy in online social networks. In *Workshop on Online Social Networks*, 2008.

[11] N. Li, T. Li, and S. Venkatasubramanian. *t*-closeness: Privacy beyond *k*-anonymity and *l*-diversity. In *IEEE ICDE*, 2007.

[12] K. Liu and E. Terzi. Towards identity anonymization on graphs. In *ACM SIGMOD* , 2008.

[13] A. Machanavajjhala, J. Gehrke, D. Kifer, and M. Venkitasubramaniam. ℓ-diversity: Privacy beyond *k*-anonymity. In *IEEE ICDE*, 2006.

[14] D. J. Martin, D. Kifer, A. Machanavajjhala, and J. Gehrke. Worst-case background knowledge for privacy-preserving data publishing. In *IEEE ICDE*, 2007.

[15] M. E. Nergiz, M. Atzori, and C. Clifton. Hiding the presence of individuals from shared databases. In *ACM SIGMOD* , 2007.

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

[17] L. Sweeney. *k*-anonymity: a model for protecting privacy. *Uncertainty, Fuzziness and Knowledge-based systems*, 10(5):557–570, 2002.

[18] R. Wong, J. Li, A. Fu, and K. Wang. (α,k)-anonymity: An enhanced *k*-anonymity model for privacy-preserving data publishing. In *ACM SIGKDD*, 2006.

[19] R. C.-W. Wong, A. W.-C. Fu, K. Wang, and J. Pei. Minimality attack in privacy preserving data publishing. In *VLDB*, 2007.

[20] X. Xiao and Y. Tao. Anatomy: Simple and effective privacy preservation. In *VLDB*, 2006.

[21] Q. Zhang, N. Koudas, D. Srivastava, and T. Yu. Aggregate query answering on anonymized tables. In *IEEE ICDE*, 2007.

[22] E. Zheleva and L. Getoor. Preserving the privacy of sensitive relationships in graph data. In *PinKDD*, 2007.

[23] B. Zhou and J. Pei. Preserving privacy in social networks against neighborhood attacks. In *IEEE ICDE*, 2008.