

# Resisting Structural Re-identification in Anonymized Social Networks

Michael Hay · Gerome Miklau · David Jensen · Don Towsley · Chao Li

Received: date / Accepted: date

**Abstract** We identify privacy risks associated with releasing network datasets and provide an algorithm that mitigates those risks. A network dataset is a graph representing entities connected by edges representing relations such as friendship, communication, or shared activity. Maintaining privacy when publishing a network dataset is uniquely challenging because an individual’s network context can be used to identify them even if other identifying information is removed.

In this paper, we introduce a parameterized model of structural knowledge available to the adversary and quantify the success of attacks on individuals in anonymized networks. We show that the risks of these attacks vary based on network structure and size, and provide theoretical results that explain the anonymity risk in random networks. We then propose a novel approach to anonymizing network data that models aggregate network structure and allows analysis to be performed by sampling from the model. The approach guarantees anonymity for entities in the network while allowing accurate estimates of a variety of network measures with relatively little bias.

## 1 Introduction

A network dataset is a graph representing a set of entities and the connections between them. Network data

This is an author-created version of an article that appears in the *The VLDB Journal*, Volume 19, Number 6, 797-823, doi: 10.1007/s00778-010-0210-x. The final publication is available at [www.springerlink.com](http://www.springerlink.com). An earlier version appeared at the 34th International Conference on Very Large Data Bases (VLDB), 2008.

M. Hay · G. Miklau · D. Jensen · D. Towsley · Chao Li  
Department of Computer Science  
University of Massachusetts Amherst  
Amherst, MA 01002  
E-mail: [mhay,miklau,jensen,towsley,chaoli@cs.umass.edu](mailto:mhay,miklau,jensen,towsley,chaoli@cs.umass.edu)

can describe a variety of domains: a *social network* might describe individuals connected by friendships; an *information network* might describe a set of articles connected by citations; a *communication network* might describe Internet hosts related by traffic flows. As our ability to collect network data has increased, so too has the importance of analyzing these networks. Networks are analyzed in many ways: to study disease transmission, to measure a publication’s influence, and to evaluate the network’s resiliency to faults and attacks. Such analyses inform our understanding of network structure and function.

However, it can be difficult to obtain access to network data, in part because many networks contain sensitive information, making data owners reluctant to publish them. An example of a network containing sensitive information is the social network studied by Poterat et al. [37], which describes a set of individuals related by sexual contacts and shared drug injections, relationships that are clearly sensitive. In this case, the researchers chose to publish the network. While society knows more about how HIV spreads because this network was published and analyzed, researchers had to weigh that benefit against possible losses of privacy to the individuals involved. Other kinds of networks, such as communication networks, are also considered sensitive and for that reason are rarely published. For example, to our knowledge, the sole publicly available network of email communication was published only because of litigation [7].

We consider the problem of publishing network data in such a way that permits useful analysis yet avoids disclosing sensitive information. Most existing work on privacy in data publishing has focused on tabular data, where each record represents a separate entity, and an individual may be re-identified by matching the individual’s publicly known attributes with the attributes of the anonymized table. Anonymization techniques for tabular data do not apply to network data because they fail to account for the interconnectedness of the entities (i.e., they destroy the network structure).

Because network analysis can be performed in the absence of entity identifiers (e.g., name, social security number), a natural strategy for protecting sensitive information is to replace identifying attributes with synthetic identifiers. We refer to this procedure as *naive anonymization*. It is a common practice and presumably, it protects sensitive information by breaking the association between the real-world identity and the sensitive data.

However, naive anonymization may be insufficient. A distinctive threat in network data is that an entity’s connections (i.e., the network structure around it) can be distinguishing, and may be used to re-identify an otherwise anonymous individual. We consider how a malicious individual (the *adversary*) might obtain partial knowledge about the network structure around targeted individuals and then use this knowledge to re-identify them in the anonymized network. Once re-identified, the adversary can learn additional properties about the targets; for instance, he may be able to infer the presence or absence of edges between them. Since individual connections are often considered sensitive information, such *edge disclosure* constitutes a violation of privacy. Whether naive anonymization provides adequate protection depends on the structure of the network and the adversary’s capability. In this paper, we provide a comprehensive assessment of the privacy risks of naive anonymization.

Although an adversary may also have information about the attributes of nodes, the focus of this paper is on disclosures resulting from *structural* or *topological* re-identification, where the adversary’s information is about the structure of the graph only. The use of attribute knowledge to re-identify individuals in anonymized data has been well-studied, as have techniques for resisting it [31, 32, 41, 42, 44]. More importantly, many network analyses are concerned exclusively with structural properties of the graph, therefore safely publishing an unlabeled network is an important goal in itself. For example, the following common analyses examine only the network structure: finding communities, fitting power-law models, enumerating motifs, measuring diffusion, and assessing resiliency [35].

In this paper, we make the following contributions:

- **Adversary Model** We propose a flexible model of external information used by an adversary to attack naively-anonymized networks. The model allows us to evaluate re-identification risk efficiently and for a range of different adversary capabilities. We also formalize the structural indistinguishability of a node with respect to an adversary with locally-bounded external information (Section 2).
- **Empirical Risk Assessment** We evaluate the effectiveness of structural attacks on real and synthetic networks, measuring successful re-identification and edge disclosures. We find that real networks are diverse in their resistance to attacks. Nevertheless, our results demonstrate that naive anonymization provides insufficient protection, especially if an adver-

sary is capable of gathering knowledge beyond a target’s immediate neighbors (Section 3).

- **Theoretical Risk Assessment** In addition to the empirical study, we perform a theoretical analysis of random graphs. We show how properties such as a graph’s density and degree distribution affect re-identification risk. A significant finding is that in sufficiently dense graphs, nodes can be re-identified even when the graph is extremely large (Section 4).
- **Privacy Definition** We also propose strategies for mitigating re-identification risk. First, we propose a privacy condition, which formally specifies a limit on how much the adversary can learn about a node’s identity. We compare it with other definitions that have been proposed in the literature and discuss its limitations (Section 5).
- **Anonymization Algorithm** Then we propose a novel algorithm to achieve this privacy condition. The algorithm produces a generalized graph, which describes the structure of the original graph in terms of node groups called *supernodes*. The generalized graph retains key structural properties of the original graph yet ensures anonymity (Section 6).
- **Algorithm Evaluation** We perform a comprehensive evaluation of the utility of the generalized graphs. This includes a comparison with other state-of-the-art graph anonymization algorithms (Section 7).

We conclude the paper with a comprehensive review of related work (Section 8).

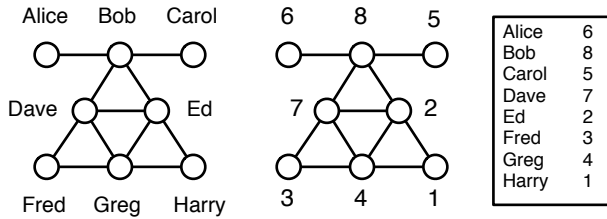
## 2 Modeling the adversary

In this section we describe the capabilities and motivations of the adversary in the context of network data. First, we describe the process of naive anonymization and how the adversary may attack it. Second, we define the threats of node re-identification and edge disclosure. Third, we explain how anonymity is achieved through structural similarity, which motivates a model of adversary knowledge based on degree signatures. Finally we review alternative models of the adversary.

### 2.1 Naive anonymization

Formally, we model a network as an undirected graph  $G = (V, E)$ . The naive anonymization of  $G$  is an isomorphic graph,  $G_a = (V_a, E_a)$ , defined by a random bijection  $\Pi : V \rightarrow V_a$ . For example, Figure 1 shows a small network represented as a graph along with its naive anonymization. The *anonymization mapping*  $\Pi$ , also shown, is a random, secret mapping.

Naive anonymization prevents re-identification when the adversary has no information about individuals in the original graph. Formally stated, an individual  $x \in V$ , called the *target*, has a *candidate set*, denoted  $\text{cand}(x)$ ,



**Fig. 1** A social network represented as a graph (left), the naive anonymization (center), and the anonymization mapping (right).

which consists of the nodes of  $G_a$  that could feasibly correspond to  $x$ . To assess the risk of re-identification, we assume each element of the candidate set is equally likely and use the size of the candidate set as a measure of resistance to re-identification. Since  $\Pi$  is random, in the absence of other information, any node in  $G_a$  could correspond to the target node  $x$ . Thus, given an uninformed adversary, each individual has the same risk of re-identification, specifically  $\text{cand}(x) = V_a$  for each target individual  $x$ .

However, if the adversary has access to external information about the entities, he may be able to reduce the candidate set and threaten the privacy of individuals.

## 2.2 Threats

In practice the adversary may have access to external information about the entities in the graph and their relationships. This information may be available through a public source beyond the control of the data owner, or may be obtained by the adversary’s malicious actions. For example, for the graph in Figure 1, the adversary might know that “*Bob has three or more neighbors,*” or that “*Greg is connected to at least two nodes, each with degree 2.*” Such information allows the adversary to reduce the set of candidates in the anonymized graph for each of the targeted individuals. For example, the first statement allows the adversary to partially re-identify Bob:  $\text{cand}(\text{Bob}) = \{2, 4, 7, 8\}$ . The second statement re-identifies Greg:  $\text{cand}(\text{Greg}) = \{4\}$ .

Re-identification can lead to additional disclosures under naive anonymization. If an individual is uniquely re-identified, then the entire structure of connections surrounding the individual is revealed. If two individuals are uniquely re-identified, then the presence or absence of an edge between them is revealed directly by the naively anonymized graph. Such an edge disclosure, in which an adversary is able to accurately infer the presence of an edge between two identified individuals, can be a serious privacy threat. In the present work, we consider the general threat of *re-identification* as well as the more specific threat *edge disclosure*.

Throughout the paper, we model the adversary’s external information as access to a source that provides

answers to a restricted *knowledge query* evaluated for a single target node of the original graph  $G$ .

An adversary attempts re-identification for a target node  $x$  by using  $Q(x)$  to refine the feasible candidate set. Since  $G_a$  is published, the adversary can easily evaluate *any* structural query directly on  $G_a$ , looking for matches. The adversary will compute the refined candidate set that contains all nodes in the published graph  $G_a$  that are consistent with answers to the knowledge query on the target node.

**Definition 1 (Candidate Set under Q)** For a knowledge query  $Q$  over a graph, the candidate set of target node  $x$  w.r.t  $Q$  is  $\text{cand}_Q(x) = \{y \in V_a \mid Q(x) = Q(y)\}$ .

*Example 1* Referring to the example graph in Figure 1, suppose  $Q$  is a knowledge query returning the degree of a node. Then for targets *Ed*, *Fred*, *Greg* we have  $Q(\text{Ed}) = 4$ ,  $Q(\text{Fred}) = 2$ ,  $Q(\text{Greg}) = 4$ , and candidate sets  $\text{cand}_Q(\text{Ed}) = \text{cand}_Q(\text{Greg}) = \{2, 4, 7, 8\}$  and  $\text{cand}_Q(\text{Fred}) = \{1, 3\}$ .

Given two target nodes  $x$  and  $y$ , the adversary can use the naively anonymized graph to deduce the likelihood that the nodes are connected. In the absence of external information, the likelihood of any edge is simply the density of the graph (the fraction of all possible edges that exist in the graph).

If the candidate sets for  $x$  and  $y$  have been refined by the adversary’s knowledge about  $x$  and/or  $y$ , then the adversary reasons about the likelihood  $x$  and  $y$  are connected based on the connections between the candidate sets for  $x$  and  $y$ . Thus we define the edge likelihood to be the Bayesian posterior belief assuming each candidate is an equally likely match for the targeted nodes.

**Definition 2 (Edge likelihood under Q)** For a knowledge query  $Q$  over a graph, and a pair of target nodes  $x$  and  $y$ , the inferred likelihood of edge  $(x, y)$  under  $Q$  is denoted  $\text{prob}_Q(x, y)$  and defined as:

$$\frac{|\{(u, v) \mid u \in X, v \in Y\}| + |\{(u, v) \mid u, v \in X \cap Y\}|}{|X| \cdot |Y| - |X \cap Y|}$$

where  $X = \text{cand}_Q(x)$  and  $Y = \text{cand}_Q(y)$ .

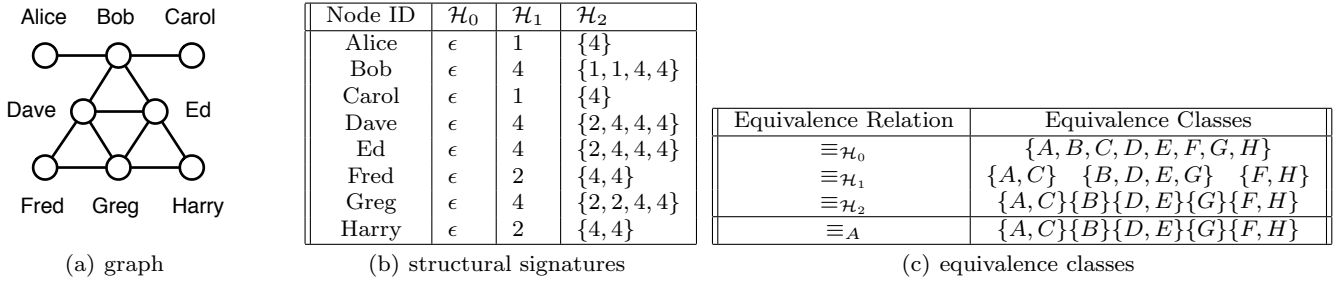
The denominator represents the total number of possible edges from a node of one candidate set to a node of the other candidate set, and accounts for the case where the intersection of the candidate sets is non-empty.

*Example 2* Continuing the example above, the inferred likelihood of edge  $(\text{Ed}, \text{Fred})$  is:

$$\text{prob}_Q(\text{Ed}, \text{Fred}) = (4 + 0)/(4 * 2) = 0.500$$

because there are 4 edges present in  $G_a$  between the disjoint candidate sets  $\text{cand}_Q(\text{Ed})$  and  $\text{cand}_Q(\text{Fred})$ . The inferred edge likelihood of edge  $(\text{Ed}, \text{Greg})$  is:

$$\text{prob}_Q(\text{Ed}, \text{Greg}) = (5 + 5)/(4 * 4 - 4) = 0.833$$



**Fig. 2** (a) A sample graph, (b) external information consisting of structural signatures  $\mathcal{H}_0, \mathcal{H}_1$  and  $\mathcal{H}_2$  computed for each individual in the graph, (c) the equivalence classes of nodes implied by the structural signatures. For the sample data,  $\equiv_{\mathcal{H}_2}$ , corresponds to automorphic equivalence,  $\equiv_A$ .

because 5 edges are present in  $G_a$  between the identical candidate sets  $\text{cand}_Q(\text{Ed})$  and  $\text{cand}_Q(\text{Greg})$ . These edge likelihoods should be compared with the prior edge density of  $2 * 11/8 * 7 = .393$ .

In Section 3, we measure the threats of edge disclosure and node re-identification on real networks.

### 2.3 Anonymity through structural similarity

Intuitively, nodes that look structurally similar may be indistinguishable to an adversary, in spite of external information. A strong form of structural similarity between nodes is *automorphic equivalence*. Two nodes  $x, y \in V$  are automorphically equivalent (denoted  $x \equiv_A y$ ) if there exists an isomorphism from the graph onto itself that maps  $x$  to  $y$ .

*Example 3* Fred and Harry are automorphically equivalent nodes in the graph of Figure 1. Bob and Ed are not automorphically equivalent: the subgraph around Bob is different from the subgraph around Ed and no isomorphism proving automorphic equivalence is possible.

Automorphic equivalence induces a partitioning on  $V$  into sets whose members have identical structural properties. It follows that an adversary — even with exhaustive knowledge of a target node’s structural position — cannot identify an individual beyond the set of entities to which it is automorphically equivalent. We say that two such nodes are *structurally indistinguishable* and observe that nodes in the graph achieve anonymity by being “hidden in the crowd” of its automorphic class members.

Some special graphs have large automorphic equivalence classes. For example, in a complete graph, or in a graph which forms a ring, all nodes are automorphically equivalent. But in most graphs we expect to find small automorphism classes, likely to be insufficient for protection against re-identification.

Though automorphism classes may be small in real networks, automorphic equivalence is an extremely strong notion of structural similarity. In order to distinguish two nodes in different automorphic equivalence classes,

it may be necessary to use complete information about their positions in the graph. For a weaker adversary with limited knowledge, nodes that are not automorphically equivalent may in fact be indistinguishable. For example, for an adversary who only knows the degree of targeted nodes in the graph, Bob and Ed are indistinguishable (even though they are not automorphically equivalent). This motivates the notion of bounded structural knowledge we describe next.

### 2.4 Adversary model based on structural signatures

We now describe the adversary model that we will use throughout the paper. It is based on a class of knowledge queries, of increasing power, which report on the local structure of the graph around a node. These queries are inspired by iterative vertex refinement, a technique originally developed to efficiently test for the existence of graph isomorphisms [10]. In Section 2.5, we discuss alternative adversary models.

The queries are denoted  $\mathcal{H}_i$  for  $i = 0, 1, 2, \dots$ . The weakest knowledge query,  $\mathcal{H}_0$ , simply returns the label of the node. (We consider here unlabeled graphs, so  $\mathcal{H}_0$  returns  $\epsilon$  on all input nodes.) The queries are successively more descriptive:  $\mathcal{H}_1(x)$  returns the degree of  $x$ ,  $\mathcal{H}_2(x)$  returns the multiset of each neighbors’ degree, and so on. The queries can be defined iteratively, where  $\mathcal{H}_i(x)$  returns the multiset of values which are the result of evaluating  $\mathcal{H}_{i-1}$  on the set of nodes adjacent to  $x$ :

$$\mathcal{H}_i(x) = \{\mathcal{H}_{i-1}(z_1), \mathcal{H}_{i-1}(z_2) \dots, \mathcal{H}_{i-1}(z_m)\}$$

where  $z_1 \dots z_m$  are the nodes adjacent to  $x$ .

*Example 4* Figure 2 contains the same graph from Figure 1 along with the computation of  $\mathcal{H}_0, \mathcal{H}_1$ , and  $\mathcal{H}_2$  for each node. For example:  $\mathcal{H}_0$  is uniformly  $\epsilon$ .  $\mathcal{H}_1(\text{Bob}) = \{\epsilon, \epsilon, \epsilon, \epsilon\}$ , which we abbreviate in the table simply as 4. Using this abbreviation,  $\mathcal{H}_2(\text{Bob}) = \{1, 1, 4, 4\}$  which represents Bob’s neighbors’ degrees.

In practice, we might expect that if an adversary can learn the degrees of the target’s neighbors, he would also

be able to learn about edges in the neighborhood. In this case, instead of learning  $\mathcal{H}_i$ , the adversary would learn a subgraph where the subgraph is induced by the edges adjacent to nodes that lie within at most  $i - 1$  edge traversals of the target. This additional knowledge would make the adversary more powerful, and thus the  $\mathcal{H}_i$  signature is a more conservative model. The  $\mathcal{H}_i$  signatures have the advantage that they are efficient to evaluate, whereas measuring subgraph knowledge requires checking for subgraph isomorphisms, an NP-Hard problem. Thus, the  $\mathcal{H}_i$  signature can be viewed as an efficient way to calculate a lower bound on the risk of the subgraph adversary. In Section 2.5, we discuss prior work, including our own, that has considered models based on knowledge of subgraphs surrounding the target.

For each query  $\mathcal{H}_i$ , we define an equivalence relation on nodes in the graph in the natural way.

**Definition 3 (Relative equivalence)** Two nodes  $x, y$  in a graph are *equivalent relative to  $\mathcal{H}_i$* , denoted  $x \equiv_{\mathcal{H}_i} y$ , if and only if  $\mathcal{H}_i(x) = \mathcal{H}_i(y)$ .

*Example 5* Figure 2(c) lists the equivalence classes of nodes according to relations  $\equiv_{\mathcal{H}_0}$ ,  $\equiv_{\mathcal{H}_1}$ , and  $\equiv_{\mathcal{H}_2}$ . All nodes are equivalent relative to  $\mathcal{H}_0$  (for an unlabeled graph). As  $i$  increases, the values for  $\mathcal{H}_i$  contain successively more precise structural information, and as a result, equivalence classes are divided.

To an adversary limited to knowledge query  $\mathcal{H}_i$ , nodes equivalent with respect to  $\mathcal{H}_i$  are indistinguishable. The following proposition formalizes this intuition:

**Proposition 1** *Let  $x, x' \in V$ . If  $x \equiv_{\mathcal{H}_i} x'$  then  $\text{cand}_{\mathcal{H}_i}(x) = \text{cand}_{\mathcal{H}_i}(x')$ .*

Iterative computation of  $\mathcal{H}$  continues until no new vertices are distinguished. We call this query  $\mathcal{H}^*$ . In the example of Figure 2,  $\mathcal{H}^* = \mathcal{H}_2$ . The vertex refinement technique is the basis of efficient graph isomorphism algorithms which can be shown to work for almost all graphs [3]. In our setting, this means that equivalence under  $\mathcal{H}^*$  is very likely to coincide with automorphic equivalence.

## 2.5 Alternative adversary models

Throughout the paper, we use the structural signatures described above as a parameterized model of external information that can capture the power of a range of adversaries. Our structural signatures have the advantage that they are efficient to evaluate even on large graphs, are amenable to theoretical analysis, and they are conservative model of structural knowledge.

One of our guiding principles is that adversary knowledge tends to be local to the targeted node, with more powerful adversaries capable of exploring the neighborhood around a node with increasing diameter.

In practice, external information about a published social network may be acquired through malicious actions by the adversary or from public information sources. In addition, a participant in the network, with some innate knowledge of entities and their relationships, may be acting as an adversary in an attempt to uncover unknown information. A legitimate privacy objective in some settings is to publish a graph in which participating individuals cannot re-identify themselves. For the participant-adversary, whose knowledge is based on their participation in the network, existing research about institutional communication networks suggests that there is a horizon of awareness of about distance two around most individuals [15].

Other work on network anonymity has also focused on adversaries whose structural knowledge is based on a local neighborhood around a target node [8, 30, 50, 51, 52]. An exception is the recent work by Narayanan et al. [34], which uses an auxiliary network to attack a target network, and work by Zou et al. [53], which protects against an adversary with unbounded structural knowledge.

In previous work [19], we considered alternative models of adversary knowledge, including partial subgraphs and signatures determined by connections to hubs. In evaluating adversaries with knowledge of partial subgraphs around a target, re-identification risk is generally lower than with degree signatures, but depends on how complete the known subgraph is. It is also computationally difficult to compute candidate sets because testing a potential candidate requires looking for a subgraph isomorphism.

Hubs are highly connected nodes observed in many network datasets. In a Web graph, a hub may be a highly visited website. In a graph of email connections, hubs often represent influential individuals. Because hubs are often outliers in a graph’s degree distribution, the true identity of hub nodes is often apparent in a naively-anonymized graph. In addition, an individual’s connections to hubs may be publicly known or easily deduced. We found that on real networks, the rate of re-identification using knowledge of hub connections was relatively low.

As mentioned above, the focus of this paper is on supporting the topological analysis of graphs. We therefore assume that attributes are not used to aid in re-identification, and our assessment of utility does not include analyses that depend on attribute values. Other authors have proposed anonymization schemes that protect against re-identification using attributes [8, 9, 52].

## 3 Empirical risk assessment

In this section we evaluate the risk of publishing the naive anonymization of a network through an empirical assessment on several real and synthetic network datasets.

For each dataset, we consider each node in turn as a target. We assume the adversary computes the structural signature of that node, and then we compute the

**Table 1** Descriptive statistics for the real and synthetic graphs studied.

Statistic	Real Datasets			Synthetic Datasets			
	HepTh	Enron	NetTrace	HOT	Power-Law	Tree	Mesh
Nodes	2510	111	4213	939	2500	3280	2500
Edges	4737	287	5507	988	7453	3279	4900
Minimum degree	1	1	1	1	2	1	2
Maximum degree	36	20	1656	91	166	4	4
Median degree	2	5	1	1	4	1	4
Average degree	3.77	5.17	2.61	2.10	5.96	1.99	3.92
Edge density	0.0007	0.0235	0.0003	0.0022	0.0024	0.0006	0.0016
Avg. cand. set size ( $\mathcal{H}_1$ )	558.5	12.0	2792.1	635.5	549.7	1821.8	2138.1
Avg. cand. set size ( $\mathcal{H}_2$ )	25.4	1.5	608.6	81.1	1.4	1659.8	1818.1
Percent re-identified ( $\mathcal{H}_1$ )	0.2	2.7	0.5	0.9	0.9	< 0.1	< 0.1
Percent re-identified ( $\mathcal{H}_2$ )	40.4	73.9	11.1	5.9	82.5	< 0.1	< 0.1

corresponding candidate set. We report the distribution of candidate set sizes across the population of nodes to characterize how many nodes are protected and how many are identifiable.

We use the following seven datasets. The **HepTh** dataset is a graph of coauthors in theoretical high-energy physics. The dataset is derived from arXiv, an online repository of papers. We extracted a subset of the authors and considered them connected if they wrote at least two papers together.

The **Enron** dataset is derived from a corpus of email sent to and from managers at Enron Corporation, made public by the Federal Energy Regulatory Commission during its investigation of the company. Two individuals are connected if they corresponded at least 5 times.

The **NetTrace** dataset was derived from an IP-level network trace collected at a major university. The trace monitors traffic at the gateway; it produces a bipartite graph between IP addresses internal to the institution, and external IP addresses. We restricted the trace to 187 internal addresses from a single campus department and the 4026 external addresses to which at least 20 packets were sent on port 80 (http traffic).

The **HOT** dataset is a model of the Internet of a single service provider (ISP). Its Heuristically Optimal Topology (HOT) is designed to reflect the economic and technological constraints that influence the topology. It has a hierarchical structure with a core of interconnected low degree (high-bandwidth) routers at its center and high-degree (low-bandwidth) routers at its periphery [28].

The **Power-Law** dataset is a random graph that is generated based on a model of growth and preferential attachment [5]. Its degree distribution follows a power-law. In some of the experiments, we also consider a slightly different dataset, **Clustered Power-Law**, which is constructed using the same model except that when edges are inserted into the graph, triangles are formed with some probability (we set  $p = 0.4$ ).

The **Mesh** dataset is a  $50 \times 50$  grid topology, where each node is connected to the four adjacent nodes in the grid. The **Tree** dataset is a balanced tree of arity 3.

All datasets have undirected edges, with self-loops removed. We eliminated a small percentage of disconnected nodes in each dataset, focusing on the largest connected component in the graph. Detailed statistics for the datasets are shown in Table 1.

### 3.1 Node re-identification

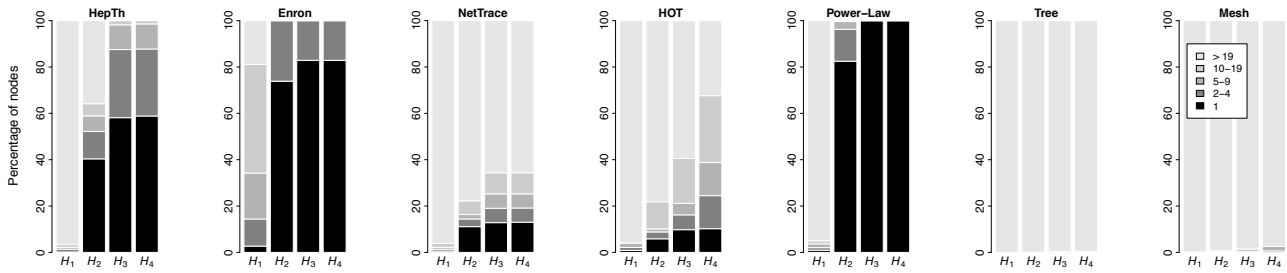
Recall from Section 2.4 that nodes contained in the same candidate set for knowledge  $\mathcal{H}_i$  share the same value for  $\mathcal{H}_i$ , are indistinguishable according to  $\mathcal{H}_i$ , and are therefore protected if the candidate set size is sufficiently large.

Figure 3 is an overview of the likelihood of re-identification under  $\mathcal{H}_1, \mathcal{H}_2, \mathcal{H}_3$  and  $\mathcal{H}_4$  knowledge queries. For each  $\mathcal{H}_i$ , the graph reports on the percentage of nodes whose candidate sets have sizes in the following buckets: [1], [2, 4], [5, 10], [11, 20], [21,  $\infty$ ]. Nodes with candidate set size 1 have been uniquely identified, and nodes with candidate sets between 2 and 4 are at high risk for re-identification. Nodes are at fairly low risk for re-identification if there are more than 20 nodes in their candidate set.<sup>1</sup> Each  $\mathcal{H}_i$  is represented as a different point on the  $x$ -axis.

Figure 3 shows that for the **HepTh** data,  $\mathcal{H}_1$  leaves nearly all nodes at low risk for re-identification, and it requires  $\mathcal{H}_3$  knowledge to uniquely re-identify a majority of nodes. For **Enron**, under  $\mathcal{H}_1$  about 15% of the nodes have candidate sets smaller than 5, while only 19% are protected in candidate sets greater than 20. Under  $\mathcal{H}_2$ , re-identification jumps dramatically so that virtually all nodes have candidate sets less than 5. These two real graphs are roughly similar in behavior to the synthetic **Power-Law** graph, as they display features similar to a power-law graph.

**NetTrace** and **HOT** have substantially lower disclosure overall, with very few identified nodes under  $\mathcal{H}_1$ ,

<sup>1</sup> We do not suggest these categories as a universal privacy standard, but merely as divisions that focus attention on the most important part of the candidate set distribution where serious disclosures are at risk.



**Fig. 3** The relationship between candidate set size and structural signature knowledge  $\mathcal{H}_i$  for  $i = 1..4$  for four real graphs and three synthetic graphs. The trend lines show the percentage of nodes whose candidate sets have sizes in the following buckets: [1] (black), [2, 4], [5, 10], [11, 20], [21,  $\infty$ ] (white).

and even  $\mathcal{H}_4$  knowledge does not uniquely identify more than 10% of the nodes. For **NetTrace**, this results from the unique bipartite structure of the trace dataset: many nodes in the trace have low degree, as they are unique or rare web destinations contacted by only one internal host. The **HOT** graph has high structural uniformity because it contains many degree one nodes that are connected to the same high degree node, and thus structurally equivalent to one another.

The synthetic **Tree** and **Mesh** graphs display very low re-identification under all  $\mathcal{H}_i$ . This is obvious given that these graphs have highly uniform structure: the nodes in **Mesh** have either degree 2 or 4, the nodes in **Tree** have degree 1, 3 or 4. We include them here for completeness as these graphs are studied in Section 7.

A natural precondition for publication is a very low percentage of high-risk nodes under a reasonable assumption about adversary knowledge. Three datasets meet that requirement for  $\mathcal{H}_1$  (**HepTh**, **NetTrace**, **HOT**). Except for the extreme synthetic graphs **Tree** and **Mesh**, no datasets meet that requirement for  $\mathcal{H}_2$ .

Overall, we observe that there can be significant variance across different datasets in their vulnerability to different adversary knowledge. However, across all datasets, the most significant change in re-identification is from  $\mathcal{H}_1$  to  $\mathcal{H}_2$ , illustrating the increased power of adversaries that can explore beyond the target’s immediate neighborhood. Re-identification tends to stabilize after  $\mathcal{H}_3$ —more information in the form of  $\mathcal{H}_4$  does not lead to an observable increase in re-identification in any dataset. Finally, even though there are many re-identified nodes, a substantial number of nodes are *not* uniquely identified even with  $\mathcal{H}_4$  knowledge.

### 3.2 Edge disclosure

We measure the risk of edge disclosure possible under adversaries with knowledge of degree signatures. Our sample datasets are sparse graphs – their edge densities are all quite low, as reported in Table 1. This means that the expectation of any particular edge existing in the graph is low.

To measure the risk of edge disclosure, we considered each edge present in the original graph and considered its inferred edge likelihood under various  $\mathcal{H}_i$ . That is, we imagine an adversary using  $\mathcal{H}_i$  knowledge to re-identify the individuals participating in each edge of the true graph, and report the inferred edge probability over the set of all true edges. For each  $\mathcal{H}_i$  we get a range of inferred edge probabilities, as illustrated in Figure 4.

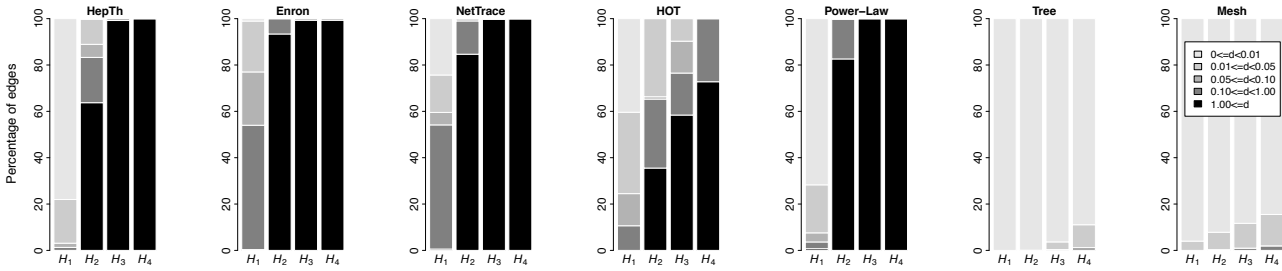
The results show that with  $\mathcal{H}_1$  knowledge alone, the risk of edge disclosure is relatively limited. In the **HepTh** data, 80% of the edges have an inferred edge probability of less than 0.01, which constitutes a small shift in an adversary’s certainty about the presence of those edges. In the **Enron** and **NetTrace** data, roughly half the edges have inferred probabilities between 0.10 and 1, which represent a significant shift in the adversary’s expectation.

Of much more concern, however, is the fact that with  $\mathcal{H}_2$  knowledge (or greater) many edges are disclosed with certainty – the inferred edge probability is 1 for a majority of edges across all datasets. It is also important to note that even when candidate sets tend to be large (such as in **NetTrace** and **HOT**), edges can be disclosed with high likelihood. In **NetTrace** and **HOT** this likely reflects a hub node with a unique degree connected to many degree-one nodes. Even though the candidate set of degree one nodes may be large, every node in that candidate set is connected to the hub, and density of connections between the candidate sets is one, resulting in certain edge disclosure.

## 4 Theoretical risk assessment

The results of the previous section show that re-identification risk varies across graphs. We want to understand and explain this variation. In some cases, such as **Tree** and **Mesh**, the low re-identification risk can be explained by the regular topology, which makes it hard to distinguish nodes by their local structure. However, across the other graphs, the reason for diversity in risk is unclear.

In this section, to gain insight into the factors affecting re-identification risk, we study random graphs.



**Fig. 4** The inferred edge probabilities resulting from attempted re-identification using structural signatures  $\mathcal{H}_1, \mathcal{H}_2, \mathcal{H}_3, \mathcal{H}_4$ .

Random graphs are governed by parameters which control some aspect of the graph’s topology; by varying the parameters, we can measure how this property affects re-identification risk. Here, we study how re-identification risk is affected by two key graph properties, density and degree distribution. To study the relationship between graph density and anonymity, we analyze the Erdős-Rényi (ER) model, the simplest random graph model. Following that, we study random graphs with power-law degree distributions. These results help us to understand under what conditions distinctive structures arise in graphs, and thus provide insight into the foundations of anonymity for graphs.

#### 4.1 Erdős-Rényi graphs

The ER model generates a graph by sampling each of the  $\binom{n}{2}$  edges independently with probability  $p$ . As the number of nodes,  $n$ , increases, these graphs exhibit different behaviors depending on how  $p$  scales with  $n$ .

We consider three cases. In a *sparse* random graph  $p = c/n$ , in a *dense* random graph  $p = c \log n/n$ , and in a *super-dense* random graph,  $p = c$  (where  $c$  is a constant). The first two cases are of interest because when  $c > 1$ , with high probability the graph includes a giant connected component of size  $\Theta(n)$  and a collection of smaller components (in the sparse case) or the graph is completely connected (in the dense case) [14].

To motivate the theoretical results that follow, Figure 5 shows experimental simulations on ER random graph of 100K nodes and varying edge probabilities. The trend lines measure the percentage of nodes *uniquely* identified by  $\mathcal{H}_1, \mathcal{H}_2$ , and  $\mathcal{H}_3$  knowledge.

The figure shows that for sparse graphs, very few nodes are uniquely identified, even with the more powerful  $\mathcal{H}_3$  knowledge. Intuitively, nodes cannot be distinguished because a sparse graph lacks sufficient edge density to create diversity in structure. Because the edge probability is  $p = c/n$ , the expected node degree, which is  $p(n-1)$ , goes to  $c$  as  $n \rightarrow \infty$ . Because the expected degree is constant, for sufficiently large  $n$ , structural patterns must repeat, leading to complete structural uniformity in the limit. The following theorem formalizes this

intuition, showing that no degree of  $\mathcal{H}_i$  knowledge can distinguish nodes in a large sparse ER random graph.

**Theorem 1 (Sparse ER random graphs)** *Let  $G$  be an ER random graph containing  $n$  nodes with edge probability given by  $p = c/n$  for  $c > 1$ . (i) The expected sizes of the equivalence classes induced by  $\mathcal{H}_i$  are  $\Theta(n)$  for any  $i \geq 0$ ; (ii) with probability going to one, the sizes of the equivalence classes induced by  $\mathcal{H}_i$  are  $\Omega(n^\alpha)$ , for any  $i \geq 0$  and any  $0 < \alpha < 1$ .*

*Proof* We begin with  $\mathcal{H}_1$ . Consider a graph of size  $n$ . Let  $N_i$  denote the degree of the  $i$ -th node,  $i \leq n$ . As  $n \rightarrow \infty$ ,

$$P(N_i = k) \rightarrow \frac{c^k}{k!} e^{-c}$$

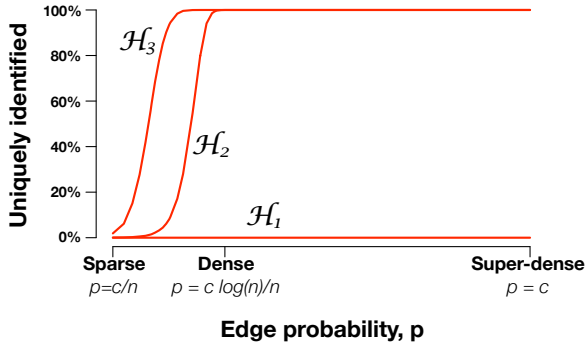
Note that for any  $k = \omega(1)$ , the probability of  $N_i = k$  goes to zero as  $n \rightarrow \infty$ . Thus, it suffices only consider the case where  $k$  is a constant.

Let  $M_{1,k}(n)$  denote the expected size of the equivalence class of  $\mathcal{H}_1$  corresponding to node degree  $k$  when the graph is of size  $n$  and let  $M_{1,k} = \lim_{n \rightarrow \infty} M_{1,k}(n)$ . We have

$$\begin{aligned} M_{1,k} &= \lim_{n \rightarrow \infty} \sum_{i=1}^n P(N_i = k) \\ &= \Theta(n) \end{aligned}$$

In order to establish the second result, we restrict ourselves to a random subset of the  $n$  nodes of size  $n^\alpha$ , where  $\alpha < 1$ . Note that the fraction of nodes in this subset goes to zero as  $n \rightarrow \infty$ . This allows us to show that, as  $n \rightarrow \infty$ , the degrees of the nodes in this subset are independent random variables. Application of a Chernoff bound then produces  $P[L_k \leq (1-\delta)n^\alpha c^k e^{-c}/k!] \leq e^{-(\delta^2 c^k e^{-c}/k!)n^\alpha}$  where  $L_k$  is the number of nodes in the subset having degree  $k$  as  $n \rightarrow \infty$ . As  $|M_{1,k}| \geq L_k$ , we conclude that  $|M_{1,k}(n)| = \Omega(n^\alpha)$  with probability going to one for all  $k$ .

Similar arguments hold for  $\mathcal{H}_i, i = 2, \dots$ . Consider a node  $x$ . We first note that the  $\mathcal{H}_i$  equivalence class that  $x$  belongs to is determined by the subgraph rooted at  $x$  that includes all nodes within distance  $i$  of it. Now, as  $n \rightarrow \infty$ , with probability going to one, this subgraph is a tree. Moreover the probability of the above subgraph



**Fig. 5** For  $\mathcal{H}_2$  and  $\mathcal{H}_3$  the number of uniquely re-identified individuals in a classical random graph goes from zero to 100% quickly when there is sufficient edge density. But regardless of the density, the number of nodes with a unique degree is close to zero, showing that  $\mathcal{H}_1$  is insufficient for unique re-identification.

deviating from a tree is  $O(1/n)$ . Another observation is that every  $\mathcal{H}_i$  induced equivalence class contains at least one node, whose distance  $i$  subgraph is a tree in the limit as  $n \rightarrow \infty$ . This follows because any  $\mathcal{H}_i$  consistent multi-set can be used to construct a tree. Thus any distance  $i$  subgraph centered at a node that is not a tree is hidden by commonly found trees.

Consider a tree,  $t$ , of height  $i$  or less. Let  $N(t)$  be a set containing the numbers of children for all nodes in the tree that are at distance  $j = 0, 1, \dots, i-1$  from the root. Let  $G_i(x)$  denote the distance  $i$  subgraph centered at node  $x$  and let  $T_i$  denote the set of all possible height  $i$  or less trees. Then

$$P(G_i(x) = t) = \prod_{k \in N(t)} \frac{c^k}{k!} e^{-c} + O(1/n), \quad t \in T_i$$

$$= \Theta(1)$$

$$P(G_i(x) \notin T_i) = O(1/n)$$

Note that as  $n$  grows, the distribution of the number of children that a node within the tree has is Poisson.

Since each equivalence class contains at least one height  $i$  or less tree in the limit as  $n \rightarrow \infty$ , it follows from the above expressions that the expected size of each equivalence class is  $\Theta(n)$ . Last a similar argument as used for  $\mathcal{H}_1$  establishes the second property.  $\square$

From the standpoint of protecting anonymity, this is an encouraging result for this class of graphs, assuming we are concerned with publishing large graphs. (In simulations, we found that some re-identification occurs in random graphs of less than  $10^6$  nodes.)

As we consider more dense ER random graphs, structural diversity increases and re-identification becomes a near certainty very quickly. Figure 5 suggests that as graphs become dense ( $p = c \log(n)/n$ ), while nodes remain well-hidden against  $\mathcal{H}_1$  adversaries,  $\mathcal{H}_2$  knowledge

is sufficient to re-identify virtually all nodes in the graph. The following theorem supports the simulations.

**Theorem 2 (Dense ER random graphs)** *Let  $G$  be an ER random graph containing  $n$  nodes with edge probability given by  $p = c \log n/n$  for  $c > 1$ .*

1. *With high probability a node belongs to an equivalence class induced by  $\mathcal{H}_1$  that grows to infinity as  $n \rightarrow \infty$ .*
2. *The expected sizes of equivalence classes induced by  $\mathcal{H}_2$  goes to zero as  $n \rightarrow \infty$ .*

The second property is consistent with simulation results as the most likely cause is that the  $\mathcal{H}_2$  signatures are unique.

*Proof* As  $n \rightarrow \infty$ , the degree distribution converges to the Poisson distribution with mean  $c \log n$ . Let  $N_i(n)$  denote the degree of node  $i$  in a graph of size  $n$  and consider degrees of the form  $N_i(n) = \delta c \log n$ ,  $0 < \delta$ . Then, as  $n \rightarrow \infty$ , we have

$$P[N_i = \delta c \log n] = \frac{1}{\sqrt{2\pi\delta c \log n} n^{c(1-\delta+\delta \log \delta)}}, \quad 0 < \delta$$

A Chernoff bound argument can be used to show that, whp, a node's degree lies within the range  $(\delta_0 c \log n, \delta_1 c \log n)$  where  $\delta_0$  is the largest value of  $\delta < 1$  such that  $c(1 - \delta + \delta \log \delta) = 1$  and  $\delta_1$  is the smallest value of  $\delta > 1$  to satisfy that equation. Note that  $P[N_i = \delta c \log n]$  decreases more quickly than  $1/n$  whenever  $\delta \notin [\delta_0, \delta_1]$  and more slowly otherwise. We focus now on the range of degrees  $(\delta_0 c \log n, \delta_1 c \log n)$ , and let  $L_\delta$  denote the number of nodes with degree  $\delta c \log n$ ,  $\delta \in [\delta_0, \delta_1]$ . Randomly select a set of nodes of size  $n^\alpha$ , where  $\alpha$  is chosen such that,  $c(1 - \delta + \delta \log \delta) < \alpha < 1$ . As in the previous theorem, we can show that the degrees of these nodes become independent random variables as  $n \rightarrow \infty$ . Apply now a Chernoff bound (as  $n \rightarrow \infty$ ) to obtain

$$P[L_\delta < (1 - \beta)n^\alpha (2\pi\delta c \log n)^{-1} n^{-c(1-\delta+\delta \log \delta)}] \leq$$

$$e^{-\beta^2 n^\alpha (2\pi\delta c \log n)^{-1} n^{-c(1-\delta+\delta \log \delta)} / 2}$$

Because of the choice of  $\alpha$ , the right hand side goes to zero. Thus  $L_\delta \rightarrow \infty$  as  $n \rightarrow \infty$  whp and therefore the size of the equivalence class corresponding to degree  $\delta c \log n$  goes to infinity as  $n \rightarrow \infty$ . Since a node takes its degree from the range  $(\delta_0 c \log n, \delta_1 c \log n)$  whp, it belongs to an equivalence class whose size goes to infinity whp as  $n \rightarrow \infty$ .

The proof of the second property is more involved. We sketch the proof. Consider a node with degree  $k$ , we need only consider  $k \in (\delta_0 c \log n, \delta_1 c \log n)$ . Moreover, we need only consider degrees of the neighbors in the same range. Furthermore, we can assume that the degrees of the neighbors are independent of each other as  $n \rightarrow \infty$ . Application of a straightforward generalization of Theorem 5.7 in [33] to the case of a non-uniformly random balls and urns problem allows us to write

$$P[X_1 = k_1, \dots, X_s = k_s] \leq$$

$$e\sqrt{\delta c \log n} \prod_{i=1}^s \frac{(p_i \delta c \log n)^{k_i}}{k_i!} e^{-p_i \delta c \log n}$$

where  $p_i$  is the probability that a neighbor selects degree  $i$ . Here  $(X_1, \dots, X_s)$  constitutes the  $\mathcal{H}_2$  signature of the node. Now, it is easy to argue using Chernoff bounds that neighbors only choose degrees clustered around  $c \log n$  ( $c \log n + l$ ,  $l = 0, \pm 1, \pm 2, \dots$ ). Hence

$$\begin{aligned} P[X_1 = k_1, \dots, X_s = k_s] &\leq \\ e\sqrt{\delta c \log n} \frac{(p_{c \log n} \delta c \log n)^{\delta c \log n}}{\prod_{i=1}^s k_i!} e^{-p_{c \log n} \delta c \log n} &\leq \\ a((\log n)^{-1/2})^{\delta c \log n} (e^{-b(\log n)^{-1/2}})^{\delta c \log n} & \end{aligned}$$

Where  $a$  is a constant. The second inequality follows from  $\prod_i k_i > 1$ . Now consider the expected number of nodes with signature  $(k_1, \dots, k_s)$ ,  $M_{k_1, \dots, k_s}$ . It is upper bounded by

$$M_{k_1, \dots, k_s} \leq an((\log n)^{-1/2})^{\delta c \log n} (e^{-b(\log n)^{-1/2}})^{\delta c \log n}$$

which goes to zero as  $n \rightarrow \infty$ .  $\square$

Lastly, we include a known result for the case of a super-dense graph where  $p = 1/2$ . The following theorem, originally due to Babai and Kucera [3] and rephrased below, shows that with high probability every node will be uniquely identified using  $\mathcal{H}_3$  knowledge:

**Theorem 3 (Super-dense ER random graphs)** *Let  $G$  be an ER random graph on  $n$  nodes with edge probability  $p = 1/2$ . The probability that there exist two nodes  $x, y \in V$  such that  $x \equiv_{\mathcal{H}_3} y$  is less than  $2^{-cn}$  for constant value  $c > 0$ .*

This result provides a sufficient condition for unique re-identification of the entire population in a graph.

Theorems 2 and 3 are disappointing from an anonymity perspective. However, most social and communication networks appear to be sparse, and so Theorem 1 may be more applicable. Furthermore, real networks often have heavy-tailed degree distributions, which is not the case for ER graphs. To capture the heavy-tailed degree distribution, we also study re-identification risk in power-law graphs.

## 4.2 Power-law graphs

Several graph models have been proposed that exhibit the heavy-tailed degree distributions often observed in real networks, including the power law random graph (PLRG) model [1]. In this model, a graph is constructed by first assigning a degree to each node, where the degree is sampled from a power law distribution. Edges are inserted by randomly choosing endpoints until every node has as many edges as its specified degree. (This can result in self-loops or multiple edges between a pair of nodes, which are often removed to form a simple graph that closely approximates the original degree distribution.)

The PLRG, and other power-law models, generate graphs with constant average degree as the number of nodes increases. Thus the edge density is low, and despite the skew in node degree, we find that the structural diversity is insufficient for re-identification. We state this formally for PLRG because it is the easiest power-law graph model to analyze.

**Theorem 4 (Power-law random graphs)** *Let  $G$  be a PLRG on  $n$  nodes. With probability going to one, the expected sizes of the equivalence classes induced by  $\mathcal{H}_i$  is  $\Theta(n)$ , for any  $i \geq 0$ .*

*Proof* The proof of Theorem 4 proceeds in a similar manner to the proof of Theorem 1 except that the Poisson distribution is replaced by  $P(N_i = k) = ak^{-\alpha} > 0$ ,  $k = 0, 1, \dots$  where  $a$  is a constant such that  $\sum_{k=0}^{\infty} P(N_i = k) = 1$ .  $\square$

## 4.3 Discussion

The theoretical results of this section complement the empirical results of the previous section. We see that re-identification risk depends on graph size: the empirical results for the 2500 node **Power-Law** graph show high re-identification risk; however, Theorem 4 shows that once a power-law graph is sufficiently large, nodes will be anonymous.

In fact, the critical factor determining re-identification risk in large random graphs is not the degree distribution, but density. Sparse graphs (including power law graphs) have low re-identification risk, whereas dense graphs have high re-identification risk. This is an important finding as it shows that even in extremely large graphs, nodes are not necessarily well hidden. It depends on the topological properties of the graph. This one reason why the  $\mathcal{H}_i$  structural signatures can be a valuable tool for data owners, as they allow them to efficiently assess re-identification risk even on large graphs.

## 5 Mitigating re-identification risk

The previous two sections were about risk assessment. The main finding was that there is considerable risk in publishing the naive anonymization of a graph because informed adversaries can use their knowledge to re-identify nodes and in some cases, infer particular edges.

The next three sections are about risk mitigation. In this section, we first introduce a new condition, graph  $k$ -anonymity, which is a bound on re-identification risk. Then we relate it to other privacy definitions proposed in the literature. In Section 6, we present an algorithm that achieves graph  $k$ -anonymity by transforming the graph through a process called graph generalization. The benefit of generalization is reduced risk, but the cost is that the generalized graph is an approximation of the original

graph and therefore less useful to the analyst. Finally in Section 7, we present the results of experiments where we measure how generalization affects several graph properties commonly measured by analysts. Also in that section, we compare the proposed algorithm against state of the art graph anonymization algorithms in terms of both privacy risk and utility.

### 5.1 Structural anonymity

In the previous sections, the size of the candidate set is used as a measure of re-identification risk. This is a natural measure for naive anonymization. A node can be a candidate only if its local graph structure is an exact match to the adversary’s knowledge. Therefore each candidate is an equally plausible guess for the target. However, as we move beyond naive anonymization to consider strategies that alter the graph structure, the size of the candidate set is no longer an appropriate measure of risk.

If the graph structure has been altered by the anonymization process, the alterations may have changed the structure around the target. Therefore a candidate may include not only exact matches in the published graph, but also partial matches. In addition, not all matches are equally likely. The probability of a candidate depends on the adversary’s prior belief about the structure around the target, and on the likelihood that the algorithm altered that structure to produce the observed output.

We introduce a new privacy condition to account for these differences. Invariably, the first step of any algorithm is to perform naive anonymization to create uncertainty about the true identities of the nodes. Recall  $\Pi : V \rightarrow V_a$ , the secret mapping between identifiers in the original graph and the synthetic identifiers in the anonymized graph. The adversary’s goal is to learn this mapping; the data owner’s goal is to sufficiently alter the graph so that the adversary fails to achieve its goal.

Our privacy definition is a condition on the adversary’s posterior belief after having seen the published graph. The posterior belief depends on the published graph, the algorithm that produced the published graph, and the adversary’s prior belief. A successful anonymization is one that meets the following definition:

**Definition 4 (Graph  $k$ -anonymity under  $Q$ )** Let  $Q$  be a structural knowledge query. An anonymized graph  $G_a$  satisfies graph  $k$ -anonymity with respect to  $Q$  if

$$\forall x \in V, \forall y \in V_a : Pr[\Pi(x) = y \mid G_a] \leq 1/k$$

where the probability depends on the randomness of the algorithm that produced  $G_a$  and the adversary’s prior probability over input graphs  $G$ .

If we make the natural assumption that the adversary has no other external information other than  $Q$ , then the adversary’s prior probability is uniform over all graphs  $G$  such that in  $G$ , the structure around  $x$  agrees with  $Q(x)$ .

Revisiting naive anonymization, there is a relationship between graph  $k$ -anonymity and our previously used measure of risk, the size of the candidate set. If the probability distribution over candidates is uniform, this condition simply requires at least  $k$  candidates: a naive anonymization satisfies graph  $k$ -anonymity under  $Q$  if for any  $x$ ,  $|\text{cand}_Q(x)| \geq k$ .

Finally, as we will see in Section 6, some anonymizations are graph  $k$ -anonymous with respect to any  $Q$ . We simply say in this case that the output satisfies graph  $k$ -anonymity.

### 5.2 Relation to alternative privacy conditions and limitations

The above condition of graph  $k$ -anonymity is similar to, and in some sense encompasses other definitions recently proposed for graph data. Liu and Terzi [30] propose a condition which requires that in the published graph each degree in the graph occurs at least  $k$  times. Such an output satisfies graph  $k$ -anonymity with respect to  $\mathcal{H}_1$  (i.e., degree). Zhou and Pei [52] require that in the published graph each neighborhood (the subgraph induced by a node and its neighbors) be isomorphic to at least  $k - 1$  others. Such an output satisfies graph  $k$ -anonymity with respect to  $N$  where  $N$  is the knowledge query that returns the neighborhood subgraph of a node. Note it also satisfies graph  $k$ -anonymity with respect to  $\mathcal{H}_1$  since query  $N$  also reveals node degree.

The above definitions are graph analogues of  $k$ -anonymity [41, 42, 44], a privacy condition defined for tables. Each assumes the adversary has some knowledge about a target entity (analogous to knowledge of the quasi-identifier) and the anonymity condition requires that this knowledge cannot be used to distinguish entities in the published data. The graph data privacy conditions differ on how much knowledge the adversary is assumed to have (node degree, neighborhood, etc.); analogous to differences in the choice of quasi-identifier.

Like  $k$ -anonymity, the above definitions also have limitations. In a homogeneity attack, while the adversary is not able to distinguish among a set of candidates, all of the candidates share a common property. Because the candidates are homogenous, the adversary has learned something about the target, even though re-identification did not occur. In tabular data, definitions such as  $\ell$ -diversity [31] and  $t$ -closeness [29] have been introduced to counter the threat of homogeneity attacks.

An instance of the homogeneity attack is edge disclosure (Section 2). A published graph which is graph  $k$ -anonymous may still be vulnerable to edge disclosure. To address the threat of edge disclosure, Cormode et al. [8] introduce an edge safety condition (described in Section 7.1 of this paper). While this prevents edge disclosure, it appears to do so at a significant expense to utility, based on the experimental results in Section 7.3. In addition, we measure the risk of edge disclosure of

our proposed algorithm and find in practice it is low for reasonable  $k$  (Section 7.5).

Other attacks have been proposed on tabular data anonymizations, and analogues of these attacks may apply to graph anonymization. Attacks include the composition attack [17], the minimality attack [48], and the deFinetti attack [23]. While some of these attacks can be remedied by imposing additional conditions (e.g.,  $m$ -invariance [49] defends against the composition of multiple releases of a dynamic table), developing data publication techniques that resist all of them is an open problem, not only for graph data, but for tabular data as well. Differential privacy [13] ensures protection from all of the above attacks, but it remains unclear whether efficient and accurate data publication is possible under differential privacy [12]. As discussed in Section 8, some differentially private algorithms for graph data have been developed, but they output answers to particular queries and do not publish a graph.

## 6 Graph generalization algorithm

In this section we describe an anonymization technique that protects against re-identification by generalizing the input graph. We generalize a graph by grouping nodes into partitions, and then publishing the number of nodes in each partition, along with the density of edges that exist within and across partitions. The adversary attempts re-identification in the generalized graph, while the analyst uses it to study properties of the original graph.

### 6.1 Graph generalization

To generalize a naively-anonymized graph  $G_a = (V_a, E_a)$ , we partition its nodes into disjoint sets. The elements of a partitioning  $\mathcal{V}$  are subsets of  $V_a$ . They can be thought of as *supernodes* since they contain nodes from  $G_a$ , but are themselves the nodes of a undirected generalized graph  $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ . The *superedges* of  $\mathcal{E}$  include self-loops and are labeled with non-negative weights by the function  $d : \mathcal{E} \rightarrow \mathbb{Z}^*$ .  $\mathcal{G}_{\mathcal{V}}$  is a generalization of  $G_a$  under a partitioning  $\mathcal{V}$  if the edge labels report the density of edges (in  $G_a$ ) that exist within and across the partitions:

**Definition 5 (Generalization of graph)** Let  $\mathcal{V}$  be the supernodes of  $V_a$ .  $\mathcal{G}$  is a *generalization* of  $G_a$  under  $\mathcal{V}$  if, for all  $X, Y \in \mathcal{V}$ ,  $d(X, Y) = |\{(x, y) \in E_a | x \in X, y \in Y\}|$ .

$\mathcal{G}$  summarizes the structure of  $G_a$ , but the accuracy of that summary depends on the partitioning. For any generalization  $\mathcal{G}$  of  $G_a$ , we denote by  $\mathcal{W}(\mathcal{G})$ , the set of possible worlds (graphs over  $V_a$ ) that are consistent with  $\mathcal{G}$ . Intuitively, this set of graphs is generated by considering each supernode  $X$  and choosing exactly  $d(X, X)$  edges between its elements, then considering each pair of

supernodes  $(X, Y)$  and choosing exactly  $d(X, Y)$  edges between elements of  $X$  and elements of  $Y$ . The size of  $\mathcal{W}(\mathcal{G})$  is a measure of the accuracy of  $\mathcal{G}$  as a summary of  $G_a$ .

The partitioning of nodes is chosen so that the generalized graph satisfies privacy goals and maximizes utility, as explained in Sections 6.2 and 6.3 respectively. In the extreme case that all partitions contain a single node, then the graph generalization  $\mathcal{G}$  does not provide any additional anonymity:  $\mathcal{W}(\mathcal{G})$  contains just the graph  $G_a$  (the function  $d$  encodes its adjacency matrix). At the other extreme, if all nodes are grouped into a single partition, then  $\mathcal{G}$  consists of a single supernode with a self-loop labeled with  $|E_a|$  (the total number of edges in the original graph).  $\mathcal{W}(\mathcal{G})$  is thus the set of all graphs over  $V_a$  with  $|E_a|$  edges. In this case the generalization provides anonymity, but is unlikely to be useful to the analyst since it reflects only the edge density of the original graph.

In studying a generalized graph, the analyst can sample a single random graph from  $\mathcal{W}(\mathcal{G})$  and then perform standard graph analysis on this synthetic graph. Repeated sampling can improve the accuracy of analysis. We study in Section 7 the bias and variance of estimates of graph properties based on graphs sampled from  $\mathcal{W}(\mathcal{G})$ .

### 6.2 Anonymity of generalized graphs

To ensure anonymity we require that the adversary have a minimum level of uncertainty about the identity of any target node in  $V$ . We use the size of a partition to provide a basic guarantee against re-identification and require that each partition have size at least  $k$ . This ensures that the output satisfies graph  $k$ -anonymity with respect to any structural query  $Q$ .

**Proposition 2** *Let  $\mathcal{G}$  be a generalized graph such that each supernode  $X$  has at least  $k$  nodes. Then  $\mathcal{G}$  satisfies graph  $k$ -anonymity.*

*Proof* The intuition for this claim is that the generalized graph summarizes the graph in terms of supernodes and contains no information that allows the adversary to distinguish between two nodes in the same supernode. Therefore, each of the  $k$  or more nodes in the same supernode must be equally likely candidates and the probability of any one node being the target is at most  $1/k$ .

We now give a formal proof. Given an input graph  $G$ , there are two key steps to producing a generalized graph: (a) first the nodes of the graph are relabeled, as with naive anonymization; and then (b) the nodes are partitioned into groups. We assume the algorithm that chooses the partition does not depend on the particular labels on the nodes; since it receives a naive anonymization, the labels are arbitrary. Therefore we can commute these two operations without affecting the final output. Without loss of generality, we can assume that the nodes are relabeled after the partition is chosen.

Let  $\Pi : V \rightarrow V_a$  denote the function which relabels nodes. Let  $P$  denote the partition of  $V$  into groups. The output  $\mathcal{G}$  is completely determined by  $G$ ,  $\Pi$ , and  $P$ . For convenience, let  $f$  be the function that takes as input  $G, \Pi, P$  and outputs  $\mathcal{G}$ .

To show graph  $k$ -anonymity, we must show that an adversary cannot use  $\mathcal{G}$  to re-identify a target node  $x$ . Formally, we must show that for any  $x \in V$  and any  $y \in V_a$ ,  $Pr[\Pi(x) = y \mid \mathcal{G}] \leq 1/k$  where the probability comes from the randomness in the algorithm and the adversary's prior belief.

To prove this, we will show that

$$Pr[\Pi(x) = y \mid \mathcal{G}] = Pr[\Pi(x) = y' \mid \mathcal{G}]$$

for any two nodes  $y$  and  $y'$  that are in the same supernode of  $\mathcal{G}$ . Since there are at least  $k$  nodes in each supernode, this implies  $Pr[\Pi(x) = y \mid \mathcal{G}] \leq 1/k$  for any  $y$ .

Since the conditional probability  $Pr[\Pi(x) = y \mid \mathcal{G}] = Pr[\Pi(x) = y, \mathcal{G}] / Pr[\mathcal{G}]$  and the denominator does not depend on  $y$ , it suffices to show that  $Pr[\Pi(x) = y, \mathcal{G}] = Pr[\Pi(x) = y', \mathcal{G}]$ .

We can write  $Pr[\Pi(x) = y, \mathcal{G}]$  as:

$$\begin{aligned} & Pr[\Pi(x) = y, \mathcal{G}] \\ &= \sum_{\pi: \pi(x)=y} Pr[\Pi = \pi, \mathcal{G}] \\ &= \sum_{\substack{\pi, g, p: \\ \mathcal{G}=f(g, \pi, p) \\ \text{and } \pi(x)=y}} Pr[\Pi = \pi, G = g, P = p] \\ &= \sum_{\substack{\pi, g, p: \\ \mathcal{G}=f(g, \pi, p) \\ \text{and } \pi(x)=y}} Pr[P = p \mid G = g] Pr[\Pi = \pi] Pr[G = g] \end{aligned}$$

where  $Pr[P = p \mid G = g]$  is the probability the algorithm outputs partition  $p$  given the input graph  $g$ ;  $Pr[\Pi = \pi]$  is the probability of a particular relabeling, which is equal to  $1/|V|!$  for any  $\pi$ ; and  $Pr[G = g]$  is the adversary's prior belief that the input graph is  $g$ .

Consider one term in the above summation by fixing the input graph  $g$ , the partition  $p$ , and the map  $\pi$ . Let  $x'$  denote the node that maps to  $y'$  under  $\pi$ , i.e.,  $\pi(x') = y'$ . Construct an alternate mapping  $\pi_{alt}$  such that the mapping for  $x$  and  $x'$  are flipped and all other mappings are unchanged:  $\pi_{alt}(x) = \pi(x')$  and  $\pi_{alt}(x') = \pi(x)$  and  $\pi_{alt}(x'') = \pi(x'')$  for all  $x'' \notin \{x, x'\}$ . There is a corresponding term in the summation for  $Pr[\Pi(x) = y', \mathcal{G}]$  where  $\pi$  is replaced with  $\pi_{alt}$ . Since  $x$  and  $x'$  appear in the same partition, we can permute their relabelings without changing the generalized graph; i.e.,  $f(g, \pi, p) = f(g, \pi_{alt}, p)$ . Since each term in the above summation for  $Pr[\Pi(x) = y, \mathcal{G}]$  can be paired with an equal term in the summation for  $Pr[\Pi(x) = y', \mathcal{G}]$ , then  $Pr[\Pi(x) = y, \mathcal{G}] = Pr[\Pi(x) = y', \mathcal{G}]$  and this completes the proof.  $\square$

Requiring a minimum supernode size of  $k$  only imposes an upper bound on the adversary's confidence in

the true identity of his target. For some graphs and some adversaries, the adversary's confidence may be much less than  $1/k$ .

For example, consider an adversary who knows only the degree of its target. The candidates for the target include any node such that in some possible world, its degree matches the target's degree. For each supernode, we can determine a range of degrees such that for each degree in that range and each node in that supernode, there exists a possible world where that node obtains that degree. For supernode  $X$ , the range is determined by mindegree and maxdegree, which are defined as  $\text{mindegree}(X) = \max(0, d(X, X) - \binom{|X|-1}{2}) + \sum_{Y \in \mathcal{V}} \max(0, d(X, Y) - (|X|-1)|Y|)$  and  $\text{maxdegree}(X) = \min(|X| - 1, d(X, X)) + \sum_{Y \in \mathcal{V}} \min(|Y|, d(X, Y))$ .

The degree range of each supernode determines the candidates, however, not all candidates are equally likely. Intuitively, a node is more likely if there are more possible worlds in which its degree matches the target.

In general, it may be computationally hard to determine the adversary's posterior probability of a candidate being the target. The brute force solution—enumerating all possible worlds and computing candidate set in each one—requires exponential time. We conservatively require  $k$ -sized partitions but observe that in practice this may provide much stronger protection than that implied by the value of  $k$ .

### 6.3 Algorithm description

We now present the graph generalization algorithm, which we call GraphGen. The input to the GraphGen is  $G_a$  and privacy parameter  $k$ . The output is a generalized graph  $\mathcal{G}$ . Pseudocode for the algorithm is given in Algorithm 1.

Subject to the privacy constraint, which requires the supernodes of  $\mathcal{G}$  to be of size at least  $k$ , we would like to find the generalized graph that best fits the input graph. We estimate fitness via a maximum likelihood approach. We consider a uniform probability distribution over the possible worlds  $\mathcal{W}(\mathcal{G})$ . For a graph  $g \in \mathcal{W}(\mathcal{G})$  we define  $Pr_{\mathcal{G}}[g] = 1/|\mathcal{W}(\mathcal{G})|$  where the number of possible worlds is:

$$|\mathcal{W}(\mathcal{G})| = \prod_{X \in \mathcal{V}} \binom{\frac{1}{2}|X|(|X|-1)}{d(X, X)} \prod_{X, Y \in \mathcal{V}} \binom{|X||Y|}{d(X, Y)}$$

Without regard to the anonymity condition, the generalized graph that maximizes likelihood is the one with each node in a separate partition. Then, as explained above,  $|\mathcal{W}(\mathcal{G})| = 1$  and  $Pr_{\mathcal{G}}[G_a] = 1$ . In general, likelihood is greater with more supernodes because each supernode introduces more parameters to fit a fixed amount of data. But subject to the minimum size constraint, generalized graphs can vary greatly in their fit to the input graph. GraphGen uses local search to explore the exponential number of generalized graphs.

---

**Algorithm 1** GraphGen, an algorithm that generalizes a graph to ensure anonymity.

---

**Input:**  $G_a = (V_a, E_a)$ , graph to generalize  
 $k$ , minimum supernode size  
**Output:**  $\mathcal{G}$ , a generalized graph such that each supernode contains at least  $k$  nodes

- 1:  $\mathcal{G} \leftarrow \text{Initialize}(G_a)$  {All nodes in one partition.}
- 2:  $t_{cycle} \leftarrow 5|V_a|$
- 3: **for**  $t \leftarrow 1$  to  $\infty$  **do**
- 4:    $T \leftarrow \text{Schedule}(t)$  {Temperature  $T$  cools as  $t$  increases.}
- 5:    $S \leftarrow \text{Successors}(\mathcal{G}, k)$
- 6:    $\mathcal{G}' \leftarrow \arg \max_{\mathcal{G}' \in S} \frac{1}{|\mathcal{W}(\mathcal{G}')|}$  {Find max likelihood successor}
- 7:    $\Delta L \leftarrow \frac{1}{|\mathcal{W}(\mathcal{G}')|} - \frac{1}{|\mathcal{W}(\mathcal{G})|}$  {Change in likelihood}
- 8:   **if**  $\Delta L > 0$  **then**
- 9:      $\mathcal{G} \leftarrow \mathcal{G}'$
- 10:   **else**
- 11:      $\mathcal{G} \leftarrow \mathcal{G}'$  with probability  $e^{\Delta L/T}$
- 12:   **end if**
- 13:   **if**  $\mathcal{G}$  updated less than 0.02% of last  $t_{cycle}$  steps **then**
- 14:     **return**  $\mathcal{G}$
- 15:   **end if**
- 16: **end for**

**Successors** subroutine returns a set of generalized graphs that can be derived from  $\mathcal{G}$  by making a small change, such as splitting or merging a supernode in  $\mathcal{G}$ .

**Input:**  $\mathcal{G}$ , current generalized graph  
 $k$ , minimum supernode size  
**Output:** a set of generalized graphs, the successors to  $\mathcal{G}$

- 1:  $S \leftarrow \emptyset$  {The set of successors}
- 2:  $u \leftarrow$  Choose random node
- 3:  $X \leftarrow$  Find supernode that contains  $u$
- 4: **if**  $|X| > 2k$  **then**
- 5:    $\mathcal{G}' \leftarrow \text{Split}(X, \mathcal{G})$  {Choose greedy split of  $X$ }
- 6:    $S \leftarrow S \cup \{\mathcal{G}'\}$
- 7: **end if**
- 8: **for**  $Y$  such that  $X, Y$  are neighbors or share a neighbor **do**
- 9:   **if**  $|X| > k$  **then**
- 10:      $\mathcal{G}' \leftarrow \text{MoveNode}(u, X, Y, \mathcal{G})$
- 11:      $S \leftarrow S \cup \{\mathcal{G}'\}$
- 12:   **end if**
- 13:    $\mathcal{G}' \leftarrow \text{MergeAndSplit}(X, Y, \mathcal{G})$
- 14:    $S \leftarrow S \cup \{\mathcal{G}'\}$
- 15: **end for**
- 16: **return**  $S$

---

The design of the search algorithm is based on techniques for solving a related social network analysis problem: *stochastic block-modeling* [35]. The objective of stochastic block-modeling is to cluster the nodes of the graph so that nodes in the same group play a similar “social role” in the graph. While the high-level idea is the same, there are a few key distinctions from our work. First, our differing motivations result in different likelihood functions. In stochastic block-modeling, the goal is to build a predictive model of the data and so the likelihood includes a penalty term for model complexity; in contrast, our goal is to fit the original graph as closely as possible given the anonymity condition. Second, the anonymity

condition imposes a new constraint on the search space, which makes search more complex.

To find the generalized graph that maximizes the likelihood function, GraphGen searches using simulated annealing [40]. Each valid generalized graph (i.e., those such that each supernode at least  $k$  nodes) is a state in the search space. Starting with a generalized graph that has a single partition (i.e., supernode) containing all nodes, GraphGen proposes a change of state, by splitting a partition, merging two partitions, or moving a node to a different partition. The proposal of changing the current state from generalized graph  $\mathcal{G}$  to some new generalized graph  $\mathcal{G}'$  is evaluated based on the change in likelihood that results. The proposal is always accepted if it improves the likelihood and accepted with some probability if it decreases the likelihood. The acceptance probability starts high and is cooled slowly until, as it approaches zero, a move is accepted only if it increases the likelihood. We terminate search when fewer than 0.02% of proposals are accepted.

GraphGen may return a partitioning that is only locally maximal. Whether this happens depends in part on the cooling schedule of simulated annealing; if cooled slowly enough, it will return the global maximum with high probability [40]. Nevertheless, finding the globally optimal partition is an intractable problem, and we cannot quantify how close the output is to the optimum. In experimental results not shown, we did a more systematic exploration of the search space using random restarts. On the **Enron** graph with  $k = 3$ , the log-likelihood of the output partition ranged from  $-362.6$  to  $-353.3$ ; in contrast, a greedy algorithm returns a partition with log-likelihood of only  $-511.5$ .

To make search more efficient, we cache the statistics needed to compute likelihood. We maintain a cache of edge counts  $d(X, Y)$  to facilitate computing the likelihood. Furthermore, when considering a move in search space, it is only necessary to compute the change in likelihood, which is more efficient since a move only affects a subset of terms in the likelihood equation. For example, to split supernode  $X$  into  $X'$  and  $X''$ , the only affected terms are the ones involving  $X$ . There is a term for each neighbor  $Y$  of  $X$  (i.e.,  $Y$  such that  $d(X, Y) > 0$ ). Since the input graphs are typically sparse,  $X$  has few neighbors, resulting in only a small number of affected terms. In the worst-case, computing the change in likelihood requires time that is linear in the size of the input graph.

We also made a few design choices that make search more efficient. A supernode is split in a greedy fashion: a randomly chosen node is moved from  $X$  to a new group  $X'$ , and then for each of the next  $k-1$  nodes, we select the node that maximizes the likelihood when moved from  $X$  to  $X'$ . Second, when we consider merging two supernodes or moving a node between supernodes, we only consider supernodes  $X, Y$  that are neighbors or share a neighbor. This is locally optimal, in that if  $Y$  does not satisfy this condition, then merging  $X$  and  $Y$  can only decrease the likelihood of the current generalized graph. While these

choices may exclude the optimal assignment, results indicate that they are effective heuristics: they greatly reduce runtime without any decrease in likelihood.

#### 6.4 Capitalizing on limited adversaries

The GraphGen algorithm places each node in a supernode with at least  $k-1$  other nodes. This is a conservative approach in that it ignores the fact that some nodes may be structurally well-hidden in the original graph. Nodes may be automorphically equivalent, or so similar that only an adversary with substantial structural knowledge can distinguish them.

Such a conservative approach has consequences for utility, as graph structure is coarsened to the supernode level. We would like an approach that can take advantage of situations in which the adversary is known to have limited knowledge of graph structure or where the graphs contain many structurally homogenous nodes.

We propose an extension of GraphGen that anonymizes the graph with respect to a fixed model of adversary knowledge. The idea is to only anonymize nodes that are *vulnerable* to re-identification by the given adversary. By focusing the anonymization on the vulnerable nodes, it may be possible to preserve more of the structure of the input graph.

To incorporate into the algorithm, the first step is to identify the vulnerable nodes. Given adversary model  $Q$  and group size  $k$ , a node  $x$  is vulnerable if  $|\text{cand}_Q(x)| < k$ . For example, if  $Q$  is  $\mathcal{H}_1$ , then the only nodes that are vulnerable are the ones whose degree occurs less than  $k$  times. Then, the privacy condition on the generalized graph is altered so that the only requirement is that if a supernode contains a vulnerable node, then its size must be at least  $k$ . This means that an invulnerable node can be placed in a supernode of size 1.

This relaxed privacy condition can be incorporated into the search procedure by allowing state changes that place invulnerable nodes into supernodes of size less than  $k$ . Alternatively, the search can execute as described above, and then supernodes that contain only invulnerable nodes can be replaced with individual supernodes for each invulnerable node. (Supernodes containing a mixture of vulnerable and invulnerable nodes must remain intact to ensure that the vulnerable nodes are protected.) In Section 7.4, we evaluate the latter approach for the  $\mathcal{H}_1$  and  $\mathcal{H}_2$  adversary models and measure the improvement in utility that results. We refer to these variants of the algorithm as  $\text{GraphGen}(\mathcal{H}_1)$  and  $\text{GraphGen}(\mathcal{H}_2)$  respectively. The pseudocode is shown in Algorithm 2.

These alternative anonymization algorithms satisfy graph  $k$ -anonymity, but for restricted adversaries.

**Corollary 1** *The output of  $\text{GraphGen}(\mathcal{H}_1)$  satisfies graph  $k$ -anonymity with respect to  $\mathcal{H}_1$ . Similarly, the output of  $\text{GraphGen}(\mathcal{H}_2)$  satisfies graph  $k$ -anonymity with respect to  $\mathcal{H}_2$ .*

---

**Algorithm 2**  $\text{GraphGen}(Q)$  a modification of Algorithm 1 that protects against  $Q$  adversaries.

---

**Input:**  $G_a = (V_a, E_a)$ , graph to generalize  
 $k$ , minimum supernode size  
 $Q$  knowledge query representing adversary capability  
**Output:**  $\mathcal{G}$ , a generalized graph that satisfies graph  $k$ -anonymity with respect to  $Q$  adversaries.  
1:  $S \leftarrow \{u \in V_a \mid |\text{cand}_Q(u)| < k\}$  {Vulnerable nodes}  
2:  $\mathcal{G} \leftarrow \text{GraphGen}(G_a, k)$   
   {Replace supernodes that contain only invulnerable nodes}  
3: **for** supernode  $X$  in  $\mathcal{G}$  **do**  
4:   **if**  $X \cap S = \emptyset$  **then**  
5:     replace  $X$  with a supernode for each  $u \in X$   
6:   **end if**  
7: **end for**  
8: **return**  $\mathcal{G}$

---

This follows from Proposition 2: vulnerable nodes remain in groups of size  $k$  and are therefore protected, and invulnerable nodes are by definition nodes that the adversary cannot re-identify with confidence greater than  $1/k$  and therefore it is not necessary to generalize them.

## 7 Evaluating graph anonymization algorithms

We now present an extensive empirical evaluation of the GraphGen algorithm. We evaluate its utility, compare it to competing techniques, and measure the effectiveness of the utility enhancements proposed in Section 6.4.

The first goal of our experimental evaluation is to assess the overall utility of anonymized graphs. We would like to quantify the extent to which the anonymized graphs produced by GraphGen (and competing techniques) can serve as an accurate approximation of the original private graph. This is challenging because there are no well-defined metrics to determine the similarity of two graphs. As methods for producing anonymized networks emerge, it is becoming increasingly important to develop a reliable means for assessing their utility.

Our basic approach is to consider a suite of graph properties, measure both the original graph and the anonymized graph and compare the difference. If the anonymized graph differs from the original for some graph property, as it often does, an essential question is whether the difference is substantial. To help answer this question, we include, as a reference point, a random graph of the same size and density as the original graph. With respect to a particular measure, if the original graph looks very different from a random graph, then it is useful to compare the anonymized graph to both the original and the random graph. The more closely the anonymized graph resembles a random graph, the less useful it is. With the GraphGen approach, as group size  $k$  increases, the anonymized graph converges on a random graph, and we can measure the rate of convergence by varying  $k$ . On the other hand, when the original graph and a random

graph appear similar, then the measured property does not distinguish the original from a random graph and thus cannot be used to assess whether anonymization has preserved the structure of the original graph.

As another yardstick for measuring the loss in utility, we evaluate the anonymization algorithms on some carefully chosen combinations of metrics and synthetic graphs. Inspired by research in the networking community [2, 45], we consider a few graphs that have a deliberately engineered structure and then use metrics that capture how well this structure is preserved in the anonymized graph. For instance, we consider a graph that is a tree and measure the extent to which the graph remains tree-like after anonymization. While some of these graphs are unlikely to arise in practice, we find the experiments give useful insights into the effect of anonymization and help distinguish the behavior of competing techniques. It is also important given that real technological networks are often highly structured and poorly approximated by random graphs [27].

The second goal of the experimental evaluation is to compare GraphGen against competing techniques. One challenge is that the privacy guarantees are not always compatible and so an “apples to apples” comparison is not straightforward. We attempt to address these disparities in privacy guarantees by aligning our technique with others so that privacy conditions are comparable (Section 7.4), and by assessing the extent to which our approach is vulnerable to attacks (Section 7.5). Despite the incompatible privacy semantics in some cases, we believe that comparisons of the algorithms are still useful: their strengths and weaknesses are exposed and their tendency to bias graph measures is revealed.

We note that the goal of publishing an anonymized graph is not only to support the specific graph properties studied here. The hope is that the released dataset can be used for a wide range of investigations determined by graph topology. If measuring a specific graph property is the final objective of an analyst, alternative mechanisms for releasing that property alone should be considered (see discussion of some techniques in Section 8). At any rate, many analyses cannot be distilled into simple graph properties, and analysts often require sample datasets to refine their algorithms or interpret results.

## 7.1 Compared anonymization algorithms

In the first set of experiments, we compare the GraphGen algorithm described in Section 6 against two other algorithms for graph anonymization: the algorithm of Cormode et al. [8], denoted BCKS, and the algorithm of Liu and Terzi [30], denoted LT.

The BCKS algorithm is similar to GraphGen in that it partitions nodes into supernodes and outputs a generalized graph. However, in addition to preventing re-identification, the resulting generalized graph is also guaranteed to prevent edge disclosure. The privacy condition

ensures that each supernode contains at least  $k$  nodes and that edge disclosure is bounded by  $1/k$ . This is done by requiring that the supernodes satisfy an additional *safety* condition, which states that if two nodes share a neighbor, they must be placed in separate supernodes. The GraphGen algorithm may not prevent edge disclosure, especially at small  $k$  (see Section 7.5).

Another important difference is that the BCKS algorithm’s strategy for choosing supernodes is guided by privacy concerns—partitions are chosen to ensure low edge disclosure risk—whereas the strategy of GraphGen is guided by utility. As one might expect, we find that GraphGen achieves higher utility than BCKS.

It should also be mentioned that the approaches proposed by Cormode et al. [8] can handle richer graph data representations, including attributes on nodes and edges and multiple edge types. The focus of the empirical evaluation in [8] is on queries that involve attributes and short path queries. The focus of our study is to measure the effects of anonymization on graph topology.

The LT algorithm alters the graph through the insertion and removal of edges with the goal of making nodes more structurally uniform. The output is a single graph, not a generalized graph. The algorithm alters the graph until each node degree occurs at least  $k$  times. This prevents re-identification by an adversary whose knowledge is limited to node degree (i.e., an  $\mathcal{H}_1$  adversary). It may not protect against a more powerful adversary (e.g., an  $\mathcal{H}_2$  adversary). Given the weaker privacy condition, the LT can achieve better utility than BCKS and GraphGen on some measures.

The LT algorithm anonymizes the graph in a two-stage process. First, it finds the minimum change to the degree sequence such that the privacy condition is satisfied (each degree must appear at least  $k$  times), and the degree sequence can be realized (the sequence of integers must satisfy certain graph theoretic constraints). Then, it attempts to transform the original graph into a new graph that matches this degree sequence.

This second stage is non-trivial and Liu and Terzi [30] consider several alternative algorithms. We implement and compare against SimultaneousSwap. This algorithm is the only one that allows both edge insertions and deletions and appears to perform better than some of the alternative approaches proposed in [30] that only allow edge insertions. It is a greedy algorithm that starts with a canonical graph conforming to the anonymized degree sequence and rewires it in such a way that preserves its degree sequence but increases the edge overlap with the original graph.

## 7.2 Overview of experiments

To assess how anonymization impacts utility, we compare the original graph to the anonymized output based on several important graph properties (described below). For each property, we measure it on the original graph

and on the anonymized output. For the algorithms that output a single graph, we simply measure the property on the output graph. For the algorithms that output a generalized graph  $\mathcal{G}$ , we estimate the graph property by drawing 100 sample graphs from  $\mathcal{W}(\mathcal{G})$ , measuring the property of each sample, and then aggregating measurements across samples. We report the average and show the standard deviation using error bars. The error bars give a sense of how much variation there is among the graphs in  $\mathcal{W}(\mathcal{G})$ .

If the samples are drawn uniformly from  $\mathcal{W}(\mathcal{G})$ , this models an analyst who believes that each graph in  $\mathcal{W}(\mathcal{G})$  is equiprobable. In these experiments, we perform biased sampling taking samples uniformly from  $\mathcal{W}(\mathcal{G})$  subject to the constraint that the minimum degree is one. This makes it more likely that the sampled graph will contain a large connected component. All of the input graphs contain a single connected component, and we assume this fact is revealed to the analyst.

As a baseline, we also measure the property on a sample of 100 random graphs that are the same density as the original graph. We refer to this baseline as *Random*. Note this baseline is equivalent to applying a graph generalization algorithm where  $k = |V|$ . It has maximum privacy, but low utility as the only property of the original revealed is the number of nodes and edges.

We repeat this procedure for each graph and each setting of  $k \in \{2, 5, 10, 20\}$ . Note that while  $k$  is a common parameter across the algorithms that controls the size of the group, the resulting privacy is not the same: while GraphGen and BCKS ensure graph  $k$ -anonymity, LT ensures only graph  $k$ -anonymity with respect to degree ( $\mathcal{H}_1$ ).

We report results on the datasets that were described earlier in Section 3.

### 7.3 Results

We now present a comparison of the algorithms across several different graph metrics. Results are presented one metric at a time. We conclude with a general discussion of the findings in Section 7.3.5.

The results of the experiments are shown in Figures 6-8. Each figure presents the results for a single graph metric. The value of the metric for the true graph is shown as a dashed black line. As a reference point, the light gray region shows the value of the metric for a random graph. It is a region because it depicts a range of  $\pm 1$  standard deviation around the average value over conforming random graphs. Note that for each measure, the scales of the y-axis vary across datasets, so in some cases, while the gap between lines is large, the numerical difference is quite small.

#### 7.3.1 Paths

We consider several measures related to paths.

*Connectedness* Each of the anonymization algorithms may alter the connectivity of the graph, either dividing a connected component or merging two components. Each of the input graphs contains a single connected component, so we evaluate whether anonymization divides it. Figure 6(a) shows the results. Generally, the anonymized graphs contain a single large component, encompassing about 95% or more of the nodes. However, on the sparsest graphs—**NetTrace**, **HOT**, and **Tree**—the largest connected component of the anonymized graphs can contain as few as 70% of the nodes.

*Shortest Path Lengths* We evaluate how anonymization affects path lengths in the graph. We measure the length of a shortest path between a pair of randomly chosen nodes and compute the average length over 200 random pairs. When the graph contains multiple connected components, we only sample pairs from the largest connected component. Since the measure itself is random, there can be variation due to sampling. We measured this variation and found it small compared to the bias introduced by anonymization and so for presentation purposes we only report the average from a single sample.

Figure 6(b) shows the results. The effect of anonymization varies greatly across datasets. The greatest change occurs on **Mesh** where path lengths are dramatically shortened. In fact, for LT and BCKS, path lengths are much closer to those of a random graph than to the original graph. With the GraphGen graphs, while paths are shortened, they remain considerably longer. GraphGen tends to group neighboring nodes together, thus it does not introduce as many shortcut paths that can connect distant regions of the mesh graph.

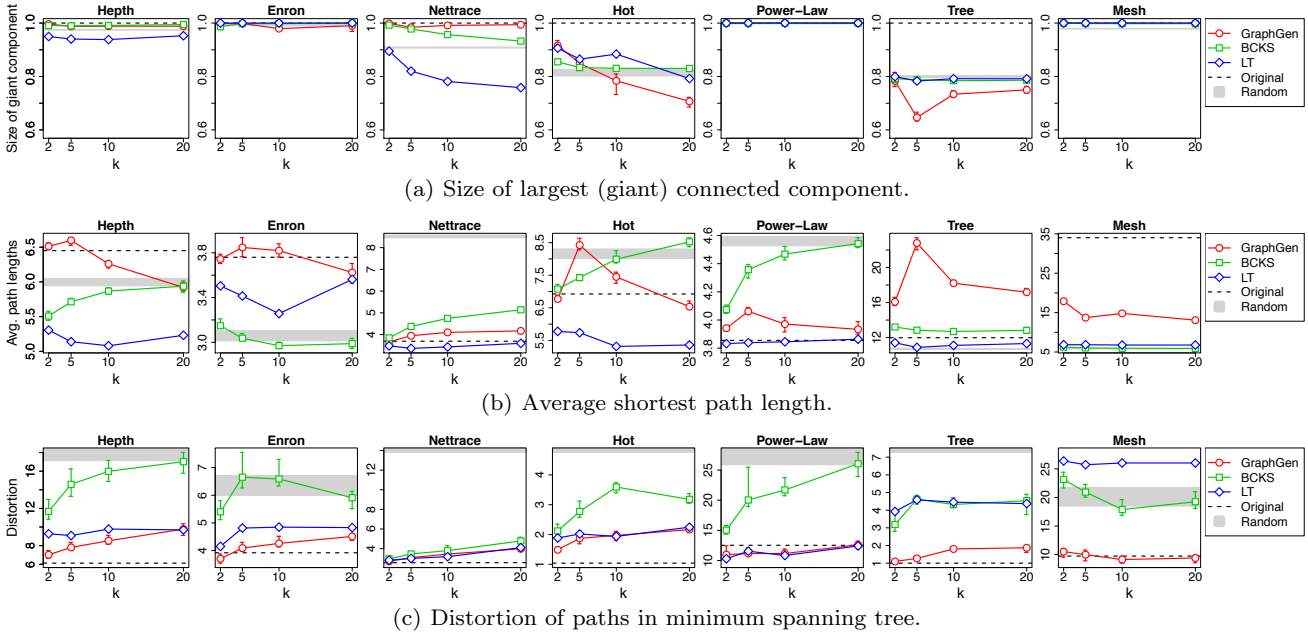
The distortion of path lengths on **Mesh** is perhaps not too surprising. For highly-structured graphs such as a mesh or a lattice, even a small amount of perturbation can greatly shorten paths by introducing a few shortcuts paths that can connect distant regions of the mesh with only a few hops [47] and meshes [24].

Generally, across all input graphs, the average path lengths of an BCKS graph appears to converge to those of *Random* as  $k$  increases. Convergence sometimes occurs at small  $k$  (e.g., **Mesh**, **Enron**, **HepTh**). Convergence occurs whether or not path lengths are shorter or longer in random graphs than with the original.

LT produces graphs with shorter path lengths than the original graph. It is very accurate on some graphs (**NetTrace**, **Power-Law**).

There are no consistent trends for GraphGen. Sometimes paths are shorter, sometimes longer. Increasing  $k$  does not have a consistent effect on path lengths. On some graphs, particularly **Tree**, the path lengths can be considerably longer than on the original graph.

*Tree-like shortest paths* We also include a graph theoretic measure called *distortion*, which in some sense captures how closely a graph resembles a tree [45]. To compute distortion of  $G$ , we first construct a spanning tree



**Fig. 6** The effect of anonymization on three graph measures related to paths. The results for three algorithms are compared, with varying privacy parameter  $k$ , on seven different graphs. The value of the given measure on the true graph is shown as a black dotted line. The value of the measure for sampled random graphs matching the density of the original is shown as a gray region.

$T$ . Then for each edge  $(u, v)$  in  $G$ , we compute the distance between  $u$  and  $v$  in  $T$ . The distortion is the average distance over all edges in  $G$ . Thus, it measures how path lengths of  $G$  are distorted (i.e., lengthened) if we are restricted to only traversing edges in tree  $T$ . If  $G$  is a tree, then distortion is 1. A random graph has a distortion of approximately  $\log n$ .

Figure 6(c) shows the distortion of the anonymized graphs. We focus on **Tree**, because the original graph is in fact a tree and so its distortion is 1. Anonymized graphs have a distortion measure exceeding 1, indicating the anonymized graphs are no longer tree-like. Distortion is high for LT and BCKS across all  $k$ . In fact, the distortion measure of the anonymized graphs is often closer to a random graph than the original tree. For GraphGen graphs, while distortion increases with  $k$ , it is very low at small  $k$ . Thus, it appears as though GraphGen more accurately preserves the tree-like structure of **Tree**.

In the other graphs, anonymization tends to produce graphs with higher distortion than the original graph. LT performs comparably to GraphGen, except on **Mesh**, where the distortion of GraphGen is much lower and closer to the original graph. On **HOT**, which has low distortion indicating tree-like structure, both GraphGen and LT preserve its tree-like structure at small  $k$ .

### 7.3.2 Degree-related measures

The degree distribution of a graph is an important property of a graph. We look at several different metrics that capture how anonymization affects degree distributions.

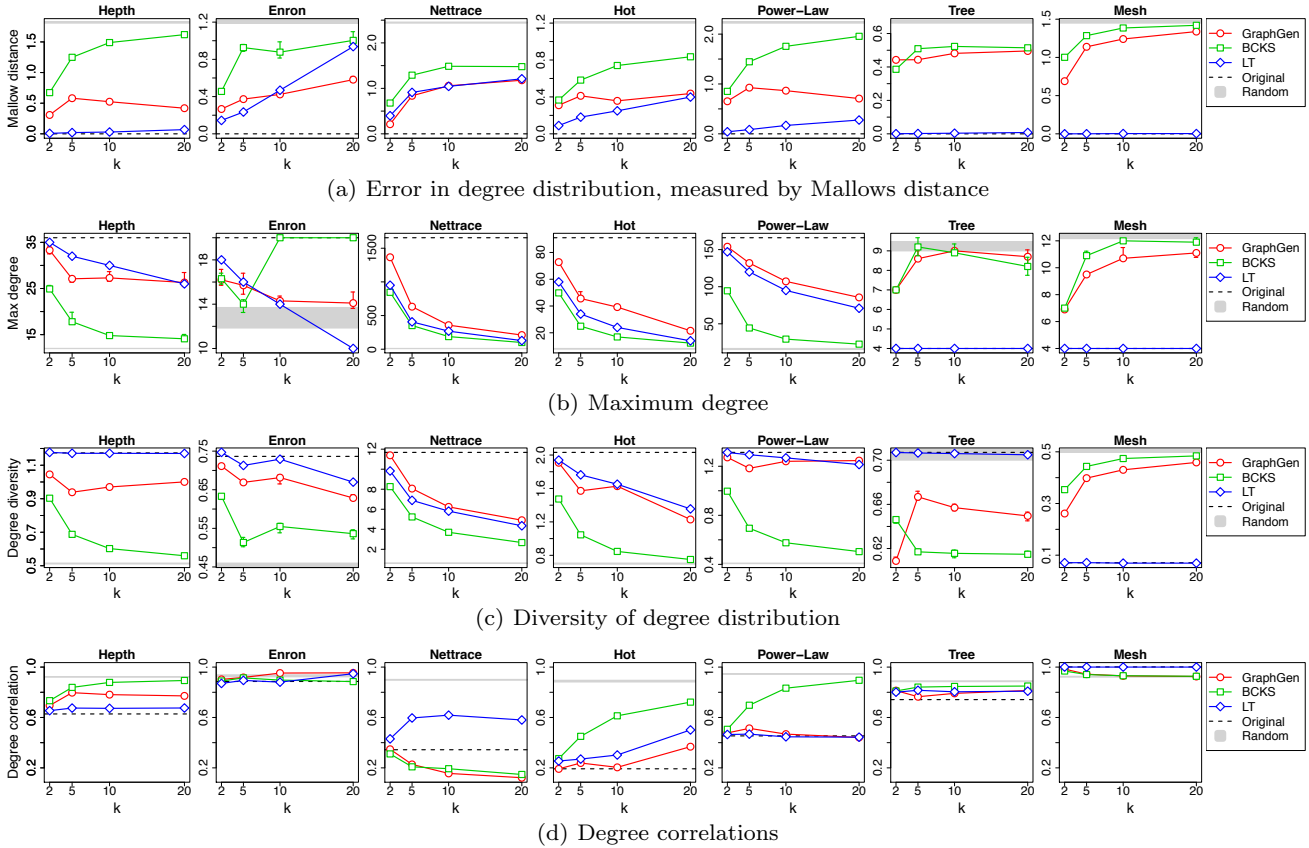
*Mallows distance* First, we compare the distributions using Mallows distance, a standard metric for comparing two distributions. Let  $d = d_1, \dots, d_n$  be the degree sequence of the original graph  $G$  where  $d_i$  corresponds to the  $i^{\text{th}}$  largest node degree in  $G$ . Let  $d'$  be the degree sequence of an anonymized graph. Mallows distance (also known as Earth Mover’s distance [26]) is the  $L_p$  distance between the two sequences

$$\text{Mallows}_p(d, d') = \left( \frac{1}{n} \sum_{i=1}^n |d_i - d'_i|^p \right)^{1/p}$$

We use  $p = 1$ . Thus, the Mallows distance captures how much, on average, each node degree is altered by anonymization. E.g., a distance of 1 means each node’s degree is changed on average by  $\pm 1$ .

Figure 7(a) shows some trends across datasets and  $k$ . Mallows distance tends to increase with  $k$ , though sometimes inconsistently for GraphGen. The anonymized graphs tend to have lower Mallows distance than Random, indicating that the degree sequence of the anonymized graph preserves some of the “structure” of the original degree sequence.

In comparing algorithms, BCKS performs worse than the other approaches, with Mallows distance rapidly approaching that of Random with increasing  $k$ . LT almost always has the lowest Mallows distance, which is expected given that the LT algorithm explicitly tries to minimize the change to the degree sequence. On graphs where the original graph has nearly uniform degree—**Mesh** and **Tree**—the LT alters the degree sequence only



**Fig. 7** The effect of anonymization on four measures related to the degree distribution. Again, the results for three algorithms are compared, with varying privacy parameter  $k$ , on seven different graphs. The value of the given measure on the true graph is shown as a black dotted line. The value of the measure for sampled random graphs matching the density of the original is shown as a gray region.

slightly to satisfy its privacy condition, resulting in a Mallows distance of zero or near zero on these graphs. GraphGen is typically between LT and BCKS.

*Maximum degree* Figure 7(b) compares the maximum degree of the original graph with the maximum degree in the anonymized graph. The figure shows a clear trend: as  $k$  increases, the maximum degree of each anonymized graph converges to the maximum degree of Random. On **Mesh** and **Tree**, the max degree is higher in Random and the max degree of anonymized graphs increase (except for LT which stays constant). For the other graphs, the max degree of Random is lower than that of the original, sometimes *much* lower. For example, on **Net-Trace**, the maximum degree is 1656 but Random has a max degree of around 10. For all approaches, anonymization reduces the max degree by more than half at  $k = 5$ . When the maximum degree is an outlier, such distortion is in some sense inevitable given the privacy condition: each node degree must be homogenous with at least  $k - 1$  other node degree. Nevertheless, such a significant change in degree suggests that the graph structure has been significantly altered.

While all approaches converge to Random, their rates of convergence differ. The max degree of BCKS changes the most rapidly with  $k$ . Surprisingly, on the graphs where the maximum degree is larger than that of a random graph, the max degree of GraphGen decreases less rapidly than LT.

*Degree variability* In addition to measuring the maximum degree, we also measure the variation in the degree distribution. The coefficient of variation  $C_V(d)$  measures the diversity of degree distribution  $d$ . It is defined as  $C_V(d) = \sigma(d)/\langle d \rangle$  where  $\langle d \rangle$  is the average degree and  $\sigma(d) = \sqrt{\sum_{i=1}^n (d_i - \langle d \rangle)^2 / (n - 1)}$ . Graphs with homogenous degrees have low  $C_V$  and graphs with diverse degree sequences, such as power-law graphs, have high  $C_V$  [2].

Figure 7(c) shows that, like maximum degree, the  $C_V$  of anonymized graphs converges towards random graphs as  $k$  increases, except on **Power-Law**, where diversity remains high at  $k = 20$ . The comparison between algorithms is similar as it is with max degree.

*Degree correlations* We also measure degree correlations—i.e., the correlation between a node’s degree and the degrees of its neighbors. It is an important property that

influences processes on networks [11]. We measure correlations using the  $s$  metric. For graph  $G = (V, E)$  it is defined as  $s(G) = \sum_{(u,v) \in E} d(u)d(v)$  where  $d(u)$  is the degree of node  $u$ . A high  $s(G)$  indicates that high degree nodes are connected to one another. We report a normalized  $s$  measure  $s(G)/s_{max}(G)$  where  $s_{max}(G)$  is the maximum possible  $s$  of any graph with the same degree sequence as  $G$ . (In practice, it is computationally intensive to find the true maximum, so we approximate it with the Havel-Hakimi graph [6], which is efficient to construct and tends to have very high  $s$ .)

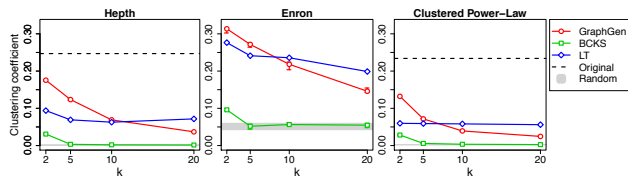
This measure is particularly interesting on the **HOT** graph. The **HOT** graph is explicitly engineered so that high degree nodes are at the periphery of the graph connected to low degree nodes, resulting in a low  $s(G)$  measure. In contrast, in a random graph, high degree nodes are likely to be connected to each other, resulting in a high  $s$  measure [2].

Figure 7(d) shows that in the anonymized version of **HOT**, increasing  $k$  results in an increased  $s$  measure. GraphGen preserves the low  $s$  measure better than LT and substantially better than BCKS. On the other graphs, the performance varies considerably, with correlations sometimes tending to Random (e.g., **HepTh**), sometimes diverging from it (e.g., **NetTrace**), and sometimes remaining constant (e.g., **Tree**).

### 7.3.3 Clustering

Clustering coefficient measures the likelihood that two neighbors of a node are themselves connected (in a social network, whether a friend of a friend is also a friend). It is defined as  $C(G) = \frac{1}{n} \sum_u \frac{\Delta(u)}{(d(u)(d(u)-1))/2}$  where  $\Delta(u)$  is the number of triangles (cliques of size 3) containing  $u$  and  $d(u)$  is the degree of  $u$ .

We report on graphs that have substantial clustering ( $C(G) > 0.15$ ). For the graphs where clustering coefficient is low, the anonymization tends to preserve the low clustering coefficient (they never exceeded 0.15). The graphs with high clustering include **Enron** and **HepTh**. We also include a synthetic graph, **Clustered Power-Law**, which is similar to **Power-Law** except that the random graph generation process is biased to introduce triangles [22]. We set the probability of triangle formation to be 0.4.



**Fig. 8** The effect of anonymization on clustering coefficient.

Figure 8 shows how anonymization reduces the clustering coefficient of clustered graphs. Even at  $k = 2$ , the

BCKS has substantially lower clustering coefficient than the original graph. At larger  $k$ , all anonymized graphs have substantially reduced clustering. At small  $k$ , GraphGen preserves the greatest amount of clustering.

With GraphGen, it is difficult to preserve clustering coefficient, especially at large  $k$ . The process of randomly sampling from  $\mathcal{W}(G)$  tends to destroy clustering coefficient. The sampled structure within each supernode is simply a random graph with a density determined by the weight of the supernode’s self edge. Unless they are very dense, random graphs have low clustering coefficient. Real world graphs, are typically very sparse, and so as  $k$  increases the density within a supernode decreases.

### 7.3.4 Runtime

We also measure the runtime of the different algorithms. We report results on one of the largest graphs, **NetTrace**; runtimes on the other graphs are qualitatively similar. While GraphGen is considerably slower than the alternative algorithms, runtime is a secondary concern as the algorithms are run “offline” by the data owner.

Table 2 shows that the runtime of BCKS does not depend on group size, agreeing with previous theoretical analysis [8]. The runtime of the LT algorithm varies across  $k$ : its runtime is dominated by the graph construction process, which depends on the number of rewiring iterations, something that varies considerably depending on the particular instance, leading to variation in runtime. Finally, the runtime of GraphGen appears to decrease with  $k$ . This is due to the fact that when groups are large, the supergraph is comparably more sparse. Therefore, the number of the successors (see Algorithm 1) is smaller, and so each step in the search runs faster.

**Table 2** On **NetTrace**, a comparison of runtimes (seconds).

Algorithm	$k = 2$	$k = 5$	$k = 10$	$k = 20$
BCKS	0.2	0.3	0.2	0.2
LT	43.4	29.6	74.2	52.7
GraphGen	3628.8	3171.9	15311.1	1560.1

### 7.3.5 Discussion

The experiments give insight into how the topological properties of graphs are affected by anonymization. Path lengths tend to more closely resemble path lengths in a random graph, whether they are shorter or longer than the original graph. Highly variable degree distributions (as occurs in power-law graphs) tend to become more uniform and high degree nodes have their degrees reduced. In graphs that are highly clustered, the effect of anonymization is to substantially reduce the clustering coefficient. However, the results also show that it is possible to provide privacy and still preserve some aspects of the original graph.

For graphs with a deliberately engineered structure (such as **Mesh**, **Tree**, and also **HOT**), anonymization

can introduce significant distortion. The GraphGen algorithm, because it explicitly accounts for structure in its anonymization, preserves these qualities relatively well. For example, paths remain long in **Mesh**, **Tree** remains tree-like, and degree correlation of **HOT** remains low.

In terms of comparing the different algorithms, we find that LT and GraphGen perform consistently better than BCKS. While LT clearly has an advantage over GraphGen on some metrics, the performance of GraphGen is often comparable and sometimes better than the performance of LT. In the next section, we resolve the difference in the privacy standards between these algorithms and present an more apples-to-apples comparison.

Recall that the error bars around the measures for GraphGen and BCKS measure the variability across samples from the  $\mathcal{W}(\mathcal{G})$ . Since the original graph  $G$  is a member  $\mathcal{W}(\mathcal{G})$ , one might expect that the error bars would overlap the measure recorded on  $G$ . This does not always occur, suggesting that while  $G$  is a possible world that is consistent with  $\mathcal{G}$ , it is unlikely to be sampled by chance. It may be possible to bias the sampling to make  $G$  more likely, but it is not clear how this impacts privacy.

As mentioned earlier, the GraphGen and BCKS approaches differ in how the generalized graph is constructed; in GraphGen it is guided by utility concerns and in BCKS it is guided by privacy concerns. The edge safety condition of BCKS requires two neighbors of a node to be placed into separate supernodes. However, the GraphGen often places a node’s neighbors together and it appears to lead to better utility. It may be that the edge safety condition, while it ensures that the output does not allow edge disclosures, may conflict with some of the utility metrics considered here.

#### 7.4 Utility of enhanced graph generalization algorithm

In this section, we evaluate the proposed enhancements to GraphGen described in Section 6.4. By focusing the anonymization only on the nodes that are vulnerable to re-identification, we hypothesize that we can improve the utility of GraphGen, which conservatively generalizes all nodes. We compare GraphGen against two alternatives: GraphGen( $\mathcal{H}_1$ ) which guards against  $\mathcal{H}_1$  adversaries, and GraphGen( $\mathcal{H}_2$ ) which protects against the stronger  $\mathcal{H}_2$  adversary. Since GraphGen( $\mathcal{H}_1$ ) provides the same privacy guarantee as LT, we also include a direct comparison of those approaches.

Based on our earlier assessment in Section 3, we expect that GraphGen( $\mathcal{H}_1$ ) will alter the input graph much less than GraphGen, as most nodes are naturally well-hidden against an  $\mathcal{H}_1$  adversary. For GraphGen( $\mathcal{H}_2$ ), it will depend on the dataset. Many nodes are vulnerable in **HepTh** and almost all nodes are vulnerable in **Enron** and **Power-Law**, so we may not expect much improvement on those datasets. For the other datasets, many nodes are well hidden at  $\mathcal{H}_2$  and so GraphGen( $\mathcal{H}_2$ ) may generalize these graphs much less than GraphGen.

We summarize the performance difference between GraphGen and its variants using a suitably normalized measure of each of the properties described in Section 7.3. We normalize each measure in terms of the distance between between GraphGen and the original graph  $G$ . Let  $P$  denote a graph property and  $P(g)$  denote the evaluation of  $P$  on graph  $g$ . The normalized score of anonymized graph  $A$  is defined as  $\frac{|P(A)-P(G)|}{|P(\text{GraphGen})-P(G)|}$ . A score of less than 1 indicates that algorithm  $A$  preserves the property more accurately than GraphGen.

Since GraphGen( $\mathcal{H}_1$ ) and GraphGen( $\mathcal{H}_2$ ) guard against weaker adversaries than GraphGen, the expectation is that the normalized score will be closer to zero, indicating closer agreement with the original graph.

Table 3 shows the results for GraphGen( $\mathcal{H}_1$ ). The results show in general that by targeting the anonymization to protect against  $\mathcal{H}_1$  adversaries, it is possible to improve utility. The magnitude of the improvement is not consistent across datasets, with datasets such **Tree** and **Mesh** seeing large gains and **Enron** seeing relatively small gains. Sometimes utility degrades (a normalized score exceeding one). Generally this is when the original GraphGen algorithm is a very accurate approximation of the original graph (e.g., distortion on **Mesh**), so the denominator of the normalized measure is small. Table 4 shows that utility improves with GraphGen( $\mathcal{H}_2$ ), but the improvement is much less than with GraphGen( $\mathcal{H}_1$ ).

*Comparison between GraphGen( $\mathcal{H}_1$ ) and LT* While the utility of LT was compared against BCKS and GraphGen in Section 7.3, these algorithms are not directly comparable in terms of their privacy guarantees because LT places restrictions on the adversary’s knowledge. However, we can directly compare LT with GraphGen( $\mathcal{H}_1$ ) because they both provide equal privacy protection.

Table 5 compares LT and GraphGen( $\mathcal{H}_1$ ) using a measure which is normalized to LT. Thus a score less than 1 indicates that GraphGen( $\mathcal{H}_1$ ) more accurately approximates the original graph, and a score exceeding 1 indicates that LT is more accurate. (A dash indicates that LT matches the original, so the normalized score is undefined; and a 0\* indicates that both LT and GraphGen( $\mathcal{H}_1$ ) perfectly match the original.) The results suggest that the approaches perform somewhat comparably. There is only one measure (distortion) in which one algorithm is consistently more accurate across the datasets, and there is no dataset where one algorithm is consistently more accurate.

#### 7.5 Assessing edge disclosure in generalized graphs

Recall our assessment (Section 3.2) of edge disclosure under naive anonymization, which showed that it is possible for a knowledgeable adversary to accurately determine whether two nodes are connected. We revisit edge disclosure here, measuring the extent to which graph generalization reduces the risk of edge disclosure.

**Table 3** A comparison of utility of GraphGen( $\mathcal{H}_1$ ) and GraphGen at  $k = 10$ . Numbers are normalized scores where less than 1 indicates GraphGen( $\mathcal{H}_1$ ) is more accurate than GraphGen.

Statistic	HepTh	Enron	NetTrace	HOT	Power-Law	Tree	Mesh
Size of giant component	0.014	0.348	0.19	0.695	0	0.333	0.086
Avg. path lengths	1.369	6.174	1.016	0.919	0.002	0.431	0.046
Distortion	0.648	0.973	0.972	0.624	0.665	0.006	0.483
Mallows distance	0.54	0.959	0.946	0.653	0.256	0.005	0.009
Max. degree	1.023	0.982	1.001	0.994	0.997	0.349	0.398
Degree diversity	0.584	1.015	1	0.95	0.911	0.006	0.034
Degree correlation	0.712	0.972	1.039	1.013	0.403	0.065	0.015
Clustering coefficient	0.508	0.869	0.223	0.318	0.954	0.002	0.023

**Table 4** A comparison of utility of GraphGen( $\mathcal{H}_2$ ) and GraphGen at  $k = 10$ . Numbers are normalized scores where a number less than 1 indicates GraphGen( $\mathcal{H}_2$ ) is more accurate than GraphGen.

Statistic	HepTh	Enron	NetTrace	HOT	Power-Law	Tree	Mesh
Size of giant component	0.921	1.03	0.926	0.923	1.079	0.772	0.086
Avg. path lengths	0.947	0.793	1.031	0.818	1.098	0.671	0.046
Distortion	0.964	1.31	1.02	0.822	1.24	0.018	0.483
Mallows distance	0.996	1.005	0.996	0.841	0.998	0.014	0.009
Max. degree	1.018	0.998	0.999	0.983	1.004	0.481	0.398
Degree diversity	0.997	0.999	1	0.973	1.002	0.004	0.034
Degree correlation	1.004	1.018	1.002	1.013	0.97	0.075	0.015
Clustering coefficient	0.995	1.009	0.93	0.643	0.926	0.006	0.023

**Table 5** A comparison of utility of GraphGen( $\mathcal{H}_1$ ) and LT at  $k = 10$ . Numbers are normalized scores where a number less than 1 indicates GraphGen( $\mathcal{H}_1$ ) is more accurate than LT. (A dash indicates that LT perfectly matched the original, so the normalized score is undefined. A 0\* indicates that both LT and GraphGen( $\mathcal{H}_1$ ) perfectly matched the original.)

Statistic	HepTh	Enron	NetTrace	HOT	Power-Law	Tree	Mesh
Size of giant component	0.003	-	0.007	1.195	0*	0.423	0*
Avg. path lengths	0.242	0.151	1.484	0.296	0.016	3.438	0.032
Distortion	0.473	0.55	1.425	0.661	0.84	0.002	0.039
Mallows distance	9.555	0.872	0.958	0.934	1.323	0.643	3.465
Max. degree	1.48	0.92	0.937	0.775	0.807	-	-
Degree diversity	46.66	8.311	0.928	1.016	1.482	0.273	8.151
Degree correlation	2.483	12.204	0.71	0.112	0.803	0.051	0.871
Clustering coefficient	0.492	1.025	0.002	1.684	0.579	0.138	0.556

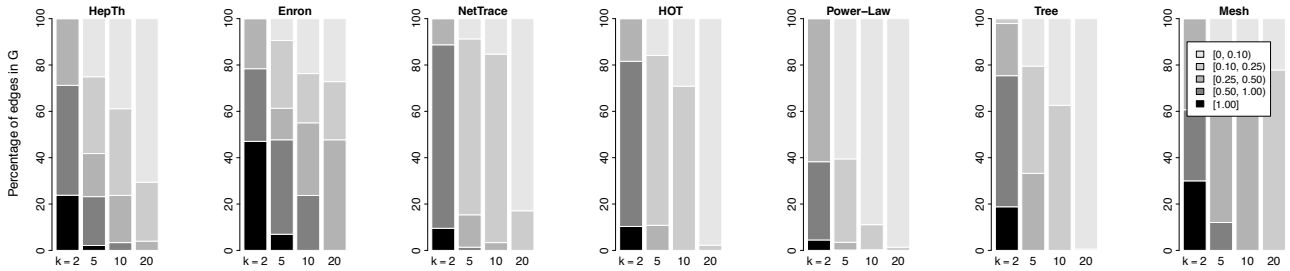
While graph generalization prevents re-identification (Section 6.2), edge disclosure may still be possible. For example, if an adversary can determine which supernode contains Alice and which supernode contains Bob he can estimate the likelihood of an edge between Alice and Bob based on the weight of the superedge between their respective supernodes. The weight reveals the number of edges in the original graph between the nodes in Alice’s supernode and the nodes in Bob’s supernode. A higher weight increases the likelihood they are connected.

To assess the risk of edge disclosure, we conservatively assume that the adversary can successfully identify the supernode of each target node. In practice, we expect that this will be difficult for an adversary with limited knowledge, so our results may exaggerate the risk. Given two target nodes  $u$  and  $v$  in  $G$ , the adversary computes the likelihood of edge between  $u$  and  $v$  by first identifying their supernodes in  $\mathcal{G}$ , denoted  $X$  and  $Y$  respectively, and then observing the superedge weight,  $d(X, Y)$ . The

likelihood of edge  $(u, v)$  is  $d(X, Y)/|X||Y|$  or, in the case when  $X = Y$ —i.e., the targets share a supernode—the edge likelihood is  $2d(X, X)/|X|(|X| - 1)$ .

Our experiment is as follows. Given a graph  $G$  and a setting of  $k$ , we produce a generalized graph  $\mathcal{G}_k$  and measure its edge disclosure risk. For each edge in the original graph  $G$ , we measure its likelihood given  $\mathcal{G}_k$ . Each edge likelihood  $\ell$  is a number in  $[0, 1]$  which we discretize into five categories from “low” ( $\ell \in [0, 0.10)$ ) to “high” ( $\ell = 1.0$ ). We report the percentage of edges in each category. This is similar to the experiments in Section 3.2 except rather than vary adversary knowledge, we assume a powerful adversary who knows the mapping of nodes to supernodes.

Figure 9 shows the results across several input graphs and settings of  $k$ . (Note the grayscale used here differs from the one used in Figure 4.) The results show that when  $k = 2$ , some edges are disclosed in all datasets. This is not surprising because at  $k = 2$ , whenever two neigh-



**Fig. 9** Risk of edge disclosure in generalized graphs across different datasets and settings of  $k$ .

bors are placed into the same supernode, the edge between them is disclosed—the weight of the self-supernode is either 1 (if they are connected) or 0 (if they are not).

At  $k = 5$ , a small portion of edges is disclosed in two graphs, **HepTh** (2.1%) and **Enron** (6.9%), but for the other graphs no edges are disclosed. Overall, edge disclosure diminishes rapidly with increasing  $k$ . By  $k = 20$ , edge likelihoods are less than half across all graphs.

The experiments show that for reasonable settings of  $k$ , the process of graph generalization greatly reduces the threat of edge disclosure. Our assessment is conservative and may overstate the threat. To prevent disclosure even at small  $k$ , one must explicitly place neighboring nodes in separate supernodes. This is done in the BCKS algorithm, which uses a safety condition to ensure that supernode weights are bounded by  $1/k$ . However, this additional safety condition has considerable cost in utility as shown in Section 7.3.

## 8 Related work

### 8.1 Attacks

Backstrom et al. [4] were the first to propose an attack on anonymized networks, demonstrating that naive anonymization does not ensure privacy. Their main result concerns an *active* attack, where the adversary is capable of adding nodes and edges *prior* to anonymization. The attack re-identifies an arbitrary set of targets by inserting a random subgraph that will be unique with high probability (independent of the input graph) and then connecting the subgraph to the targets.

Passive attacks—where the adversary attacks an already published network—have been more extensively studied. We first introduced the passive attack based on  $\mathcal{H}_i$  degree signatures in Hay et al. [19,20]. We also studied adversaries with knowledge of partial subgraph patterns around a target, and knowledge of connections to hubs in the network. Narayanan and Shmatikov [34] propose a passive attack in which the adversary exploits access to an auxiliary network whose membership overlaps with the anonymized network. Such an attack can lead to breaches of privacy if for instance the anonymized

network includes sensitive attributes or additional edges absent from the auxiliary network.

Singh and Zhan [43] measure the vulnerability to attack as a function of well known topological properties of the graph, and Wang et al. [46] propose a measure of anonymity based on description logic.

### 8.2 Network anonymization algorithms

In [20], we proposed an anonymization technique for graphs, a technique based on random edge deletions and insertions, which resisted attacks of an  $\mathcal{H}_1$  adversary, but at a significant cost in graph utility [20]. Edge randomization is further explored by Ying and Wu [50] who propose randomization strategy that biases the randomization to preserve key spectral properties of the graph. This improves utility, but they do not evaluate the impact that biasing the randomization has on privacy.

Liu and Terzi [30] propose several algorithms for anonymizing a graph through the insertion and removal of edges, altering the graph so that nodes cannot be distinguished by degree. We compare against their SimultaneousSwap algorithm in Section 7.

Zhou and Pei [52] present an anonymization algorithm for graphs that allows for labels on the nodes. They consider an adversary who knows the local neighborhood of a target (the induced subgraph of the target and its neighbors) and anonymize the graph by generalizing node labels and inserting edges until each neighborhood is isomorphic to at least  $k - 1$  others. Zou et al. [53] consider a similar approach, except require each node to be *automorphically equivalent* with  $k - 1$  others.

In their initial work on graph anonymization, Cormode et al. [9] consider bipartite graph data—representing, for example, associations between people and products they purchase—and propose an anonymization algorithm that breaks the association between identifying attributes and nodes in the graph. The main threat considered is an adversary with knowledge of node attributes, and so the anonymization leaves the structure of the graph intact. This approach is extended in [8] to handle a richer class of data, such as social networks with multiple edge types and attributes on nodes and edges. They also consider an approach which protects against an adversary with

knowledge of graph structure. They propose a partitioning based approach that we compare against in Section 7.

Zheleva et al. [51] consider graphs with labeled edges and an adversary with a predictive model for edges and knowledge of constraints on connections in the graph; the goal of anonymization is to prevent accurate prediction of a class of sensitive edges. The data model, threats considered, and adversary capabilities differ significantly from those treated here.

Rastogi et al. [39] present a mechanism for tables that has a natural interpretation for graphs. They randomly remove a fraction of original edges and randomly add a fraction of new edges. The resulting table is released in its entirety. They show that the parameters of the random process can be chosen to ensure strong protection against edge disclosure while allowing a class of counting queries to be estimated accurately. Unfortunately it does not address queries that require joins on the edge table, which are crucial to network analysis.

### 8.3 Query answering approaches

Dwork et al. [13] introduce a strong notion of privacy called differential privacy and present an interactive algorithm – where users pose queries and the data owner returns randomly perturbed answers – that achieves strong privacy guarantees and can guarantee high utility for some queries. Differential privacy has been an area of active research (for a survey, see Dwork [12]). The privacy definition extends naturally to data publication, but most work considers interactive approaches.

Some work has considered how differential privacy can be applied to network data [13, 18, 36, 38]. The proper interpretation of differential privacy for networks is not immediate, as the sensitive entity could be considered an edge, a set of edges, or a node along with its edges [18].

Under node-differential privacy, a node and all of its incident edges are protected; however, with such a rigorous privacy standard, basic network properties such as degree distribution cannot be accurately estimated. Under edge-differential privacy, which prevents the disclosure of individual edges, the degree distribution [18, 21] and other network analyses have low sensitivity and can be answered accurately [13].

However, even under edge-differential privacy, some common network analyses have high sensitivity and cannot be answered accurately. For example, measures of transitivity have  $O(n)$  sensitivity, rendering noisy estimates useless. Nissim et al. [36] propose a relaxation of differential privacy and provide algorithms for query answering where the amount of noise depends on the particular input database. Among other application, this technique can be used to accurately estimate transitivity on some graphs. Rastogi et al. [38] propose an alternative weakening of differential privacy for queries involving joins on a relationship table, allowing transitivity to be

estimated accurately along with a more general class of subgraph counting queries.

### 8.4 Other graph privacy work

The anonymization of existing networks is not the only privacy problem that arises with network data. Frikken and Golle [16] designed a protocol for privately assembling a graph that is distributed among a large number of parties. The output of the protocol is a naively-anonymized graph. Korolova et al. [25] consider an adversary who tries to re-assemble the graph from a set of views of local neighborhoods (obtained, for example, by breaking into user accounts of an online social network).

## 9 Conclusion

We have focused on what we believe to be one of the most basic and distinctive challenges for protecting privacy in network datasets—understanding the extent to which graph structure acts as an identifier. We have formalized adversary knowledge and evaluated their impact on real and synthetic networks as well as models of random graphs. We proposed anonymizing a graph by generalizing it: partitioning the nodes and summarizing the graph at the partition level. We show that a wide range of important graph analyses can be performed accurately on the generalized graphs published. An important area for future investigation is to develop bounds on the distortion introduced by anonymization. Analytical bounds could be developed through analysis of the generalized graphs, or empirical bounds could be inferred through careful sampling of the possible worlds implied by the generalized graphs. We also hope to investigate techniques that will safely permit the analyst to sample higher quality representatives from the set of possible worlds, for example, by biasing sampling towards the true graph.

## 10 Acknowledgments

Hay and Jensen were supported by the Air Force Research Laboratory and the Intelligence Advanced Research Projects Activity, under agreement number FA8750-07-2-0158. Hay, Miklau, Li, and Towsley were supported by NSF CNS 0627642. The U.S. Government is authorized to reproduce and distribute reprints for Governmental purposes notwithstanding any copyright notation thereon. The views and conclusion contained herein are those of the authors and should not be interpreted as necessarily representing the official policies or endorsements, either expressed or implied, of the Air Force Research Laboratory and the Intelligence Advanced Research Projects Activity, or the U.S. Government.

## References

1. W. Aiello, F. R. K. Chung, and L. Lu. A random graph model for massive graphs. In *Proceedings of the Thirty-second Annual ACM Symposium on Theory of Computing*, pages 171–180, Portland, OR, May 2000. ACM Press, New York, NY.
2. D. L. Alderson and L. Li. Diversity of graphs with highly variable connectivity. *Physical Review E*, 75(4):046102, Apr 2007.
3. L. Babai and L. Kucera. Canonical labelling of graphs in linear average time. In *Proceedings of the Twentieth Annual Symposium on Foundations of Computer Science*, pages 39–46, San Juan, Puerto Rico, October 1979. IEEE Computer Society.
4. L. Backstrom, C. Dwork, and J. M. Kleinberg. Wherefore art thou R3579X? Anonymized social networks, hidden patterns and structural steganography. In *Proceedings of the Sixteenth International World Wide Web Conference*, pages 181–190, Banff, AB, Canada, 2007. ACM Press, New York, NY.
5. A.-L. Barabási and R. Albert. Emergence of scaling in random networks. *Science*, 286(5439):509–512, October 1999.
6. J. Blitzstein and P. Diaconis. A sequential importance sampling algorithm for generating random graphs with prescribed degrees. Unpublished, 2006.
7. W. W. Cohen. Enron email dataset. <http://www.cs.cmu.edu/~enron/>, 2005.
8. G. Cormode, D. Srivastava, S. Bhagat, and B. Krishnamurthy. Class-based graph anonymization for social network data. *Proceedings of the VLDB Endowment*, 2(1):766–777, 2009.
9. G. Cormode, D. Srivastava, T. Yu, and Q. Zhang. Anonymizing bipartite graph data using safe groupings. *Proceedings of the VLDB Endowment*, 1(1):833–844, 2008.
10. D. G. Corneil and C. C. Gotlieb. An efficient algorithm for graph isomorphism. *Journal of the ACM*, 17:51–64, January 1970.
11. L. Costa, F. Rodrigues, G. Traverso, and P. Boas. Characterization of complex networks: A survey of measurements. *Advances in Physics*, 56(1):167–242, 2007.
12. C. Dwork. Differential privacy: A survey of results. In *Proceedings of the Theory and Applications of Models of Computation Fifth International Conference*, number 4978 in Lecture Notes in Computer Science, pages 1–19, Xi’an, China, April 25–29, 2008. Springer.
13. C. Dwork, F. McSherry, K. Nissim, and A. Smith. Calibrating noise to sensitivity in private data analysis. In *Proceedings of the Third Theory of Cryptography Conference*, number 3876 in Lecture Notes in Computer Science, pages 265–284, New York, NY, March 4–7, 2006. Springer.
14. P. Erdős and A. Rényi. On the evolution of random graphs. *Bulletin of the Institute of International Statistics*, 38:343–347, 1961.
15. N. Friedkin. Horizons of observability and limits of informal control in organizations. *Social Forces*, 62(1):54–77, 1983.
16. K. B. Frikken and P. Golle. Private social network analysis: How to assemble pieces of a graph privately. In *Proceedings of the ACM Workshop on Privacy in the Electronic Society*, pages 89–98, Alexandria, VA, October 30 2006. ACM Press.
17. S. R. Ganta, S. P. Kasiviswanathan, and A. Smith. Composition attacks and auxiliary information in data privacy. In *Proceedings of the Fourteenth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 265–273, New York, NY, USA, 2008. ACM.
18. M. Hay, C. Li, G. Miklau, and D. Jensen. Accurate estimation of the degree distribution of private networks. In *Proceedings of the IEEE International Conference on Data Mining*, pages 169–178, Los Alamitos, CA, USA, 2009. IEEE Computer Society.
19. M. Hay, G. Miklau, D. D. Jensen, D. F. Towsley, and P. Weis. Resisting structural re-identification in anonymized social networks. *Proceedings of the VLDB Endowment*, 1(1):102–114, August 2008.
20. M. Hay, G. Miklau, D. D. Jensen, P. Weis, and S. Srivastava. Anonymizing social networks. Technical Report UM-CS-2007-19, Department of Computer Science, University of Massachusetts, Amherst, MA, 2007.
21. M. Hay, V. Rastogi, G. Miklau, and D. Suciu. Boosting the accuracy of differentially-private histograms through consistency. *Proceedings of the VLDB Endowment*, 3(1):1021–1032, September 2010.
22. P. Holme and B. J. Kim. Growing scale-free networks with tunable clustering. *Physical Review E*, 65(2):026107, Jan 2002.
23. D. Kifer. Attacks on privacy and de Finetti’s theorem. In *Proceedings of the Thirty-fifth SIGMOD International Conference on Management of Data*, pages 127–138, New York, NY, USA, 2009. ACM.
24. J. M. Kleinberg. Navigation in a small world. *Nature*, 406(6798):845, August 2000.
25. A. Korolova, R. Motwani, S. U. Nabar, and Y. Xu. Link privacy in social networks. In *Proceedings of the Seventeenth ACM Conference on Information and Knowledge Management*, pages 289–298, Napa Valley, CA, October 26–30, 2008. ACM Press.
26. E. Levina and P. Bickel. The earth mover’s distance is the mallows distance: some insights from statistics. In *Proceedings of the Eighth IEEE International Conference on Computer Vision*, volume 2, pages 251–256, 2001.
27. L. Li, D. Alderson, J. Doyle, and W. Willinger. Towards a theory of scale-free graphs: Definition, properties, and implications. *Internet Mathematics*, 2(4):431–523, 2005.
28. L. Li, D. Alderson, W. Willinger, and J. Doyle. A first-principles approach to understanding the internet’s router-level topology. In *Proceedings of the 2004 Conference on Applications, technologies, architectures, and protocols for computer communications*, pages 3–14, New York, NY, USA, 2004. ACM.
29. N. Li, T. Li, and S. Venkatasubramanian.  $t$ -closeness: Privacy beyond  $k$ -anonymity and  $\ell$ -diversity. In *Proceedings of the Twenty-third International Conference on Data Engineering*, pages 106–115, Istanbul, Turkey, April 2007. IEEE Computer Society.
30. K. Liu and E. Terzi. Towards identity anonymization on graphs. In *Proceedings of the ACM SIGMOD International Conference on Management of Data*, pages 93–106, Vancouver, BC, June 10–12, 2008. ACM Press, New York, NY.
31. A. Machanavajjhala, J. Gehrke, D. Kifer, and M. Venkatasubramanian.  $\ell$ -diversity: Privacy beyond  $k$ -anonymity. In *Proceedings of the Twenty-second International Conference on Data Engineering*, page 24, Atlanta, GA, April 2006. IEEE Computer Society.
32. D. J. Martin, D. Kifer, A. Machanavajjhala, J. Gehrke, and J. Y. Halpern. Worst-case background knowledge for privacy-preserving data publishing. In *Proceedings of the Twenty-third International Conference on Data Engineering*, pages 126–135, Istanbul, Turkey, April 2007. IEEE Computer Society.

33. M. Mitzenmacher and E. Upfal. *Probability and computing: Randomized algorithms and probabilistic analysis*. Cambridge University Press, 2005.
34. A. Narayanan and V. Shmatikov. De-anonymizing social networks. In *Proceedings of the IEEE Symposium on Security and Privacy*, pages 173–187, Oakland, CA, May 17–20 2009. IEEE Computer Society.
35. M. E. J. Newman. The structure and function of complex networks. *SIAM Review*, 45(2):167–256, 2003.
36. K. Nissim, S. Raskhodnikova, and A. Smith. Smooth sensitivity and sampling in private data analysis. In *Proceedings of the Thirty-ninth Annual ACM Symposium on Theory of Computing*, pages 75–84, San Diego, CA, June 11–13, 2007. ACM Press, New York, NY.
37. J. J. Potterat, L. Phillips-Plummer, S. Q. Muth, R. B. Rothenberg, D. E. Woodhouse, T. S. Maldonado-Long, H. P. Zimmerman, and J. B. Muth. Risk network structure in the early epidemic phase of HIV transmission in Colorado Springs. *Sexually Transmitted Infections*, 78(Suppl. 1):i159–i163, 2002.
38. V. Rastogi, M. Hay, G. Miklau, and D. Suciu. Relationship privacy: Output perturbation for queries with joins. In *Proceedings of the Twenty-Eighth ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems*, pages 107–116, Providence, RI, June 29–July 2, 2009. ACM Press.
39. V. Rastogi, S. Hong, and D. Suciu. The boundary between privacy and utility in data publishing. In *Proceedings of the Thirty-third International Conference on Very Large Data Bases*, pages 531–542, Vienna, Austria, September 2007. ACM Press, New York, NY.
40. S. J. Russell and P. Norvig. *Artificial Intelligence: A Modern Approach*. Prentice Hall, Upper Saddle River, NJ, 2nd edition, 2003.
41. P. Samarati. Protecting respondents’ privacy in micro-data release. *IEEE Transactions on Knowledge and Data Engineering*, 13(6):1010–1027, November 2001.
42. P. Samarati and L. Sweeney. Protecting privacy when disclosing information: k-anonymity and its enforcement through generalization and suppression. Technical Report SRI-CSL-98-04, Computer Science Laboratory, SRI International, 1998.
43. L. Singh and J. Zhan. Measuring topological anonymity in social networks. In *Proceedings of the IEEE International Conference on Granular Computing*, pages 770–770, November 2007.
44. L. Sweeney. K-anonymity: A model for protecting privacy. *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems*, 10(5):557–570, 2002.
45. H. Tangmunarunkit, R. Govindan, S. Jamin, S. Shenker, and W. Willinger. Network topology generators: degree-based vs. structural. In *Proceedings of the 2002 conference on Applications, technologies, architectures, and protocols for computer communications*, pages 147–159, New York, NY, USA, 2002. ACM.
46. D.-W. Wang, C.-J. Liao, and T. Sheng Hsu. Privacy protection in social network data disclosure based on granular computing. In *Proceedings of the IEEE International Conference on Fuzzy Systems*, pages 997–1003, 2006.
47. D. J. Watts and S. H. Strogatz. Collective dynamics of ‘small-world’ networks. *Nature*, 393(6684):409–410, June 1998.
48. R. C.-W. Wong, A. W.-C. Fu, K. Wang, and J. Pei. Minimality attack in privacy preserving data publishing. In *Proceedings of the 33rd International Conference on Very Large Data Bases*, pages 543–554. VLDB Endowment, 2007.
49. X. Xiao and Y. Tao. M-invariance: towards privacy preserving re-publication of dynamic datasets. In *Proceedings of the 2007 ACM SIGMOD International Conference on Management of Data*, pages 689–700, New York, NY, USA, 2007. ACM.
50. X. Ying and X. Wu. Randomizing social networks: A spectrum preserving approach. In *Proceedings of the SIAM International Conference on Data Mining*, pages 739–750, Atlanta, GA, April 24–26 2008. Society for Industrial and Applied Mathematics.
51. E. Zheleva and L. Getoor. Preserving the privacy of sensitive relationships in graph data. In *First ACM SIGKDD International Workshop on Privacy, Security, and Trust in KDD, Revised Selected Papers*, number 4890 in Lecture Notes in Computer Science, pages 153–171, San Jose, CA, August 2007. Springer.
52. B. Zhou and J. Pei. Preserving privacy in social networks against neighborhood attacks. In *Proceedings of the Twenty-fourth International Conference on Data Engineering*, pages 506–515, Cancun, Mexico, April 2008. IEEE Computer Society.
53. L. Zou, L. Chen, and M. T. Özsu. K-Automorphism: A general framework for privacy preserving network publication. *Proceedings of the VLDB Endowment*, 2(1):946–957, 2009.