Accurate Estimation of the Degree Distribution of Private Networks

Michael Hay, Chao Li, Gerome Miklau, and David Jensen
Department of Computer Science
University of Massachusetts Amherst
{*mhay, chaoli, miklau, jensen*}*@cs.umass.edu*

***Abstract*—We describe an efficient algorithm for releasing a provably private estimate of the degree distribution of a network. The algorithm satisfies a rigorous property of *differential privacy*, and is also extremely efficient, running on networks of 100 million nodes in a few seconds. Theoretical analysis shows that the error scales linearly with the number of unique degrees, whereas the error of conventional techniques scales linearly with the number of nodes. We complement the theoretical analysis with a thorough empirical analysis on real and synthetic graphs, showing that the algorithm's variance and bias is low, that the error diminishes as the size of the input graph increases, and that common analyses like fitting a power-law can be carried out very accurately.**

***Keywords*-privacy; social networks; privacy-preserving data mining; differential privacy.**

I. Introduction

The analysis of social networks is crucial to addressing a diverse set of societally important issues including disease transmission, fraud detection, efficiency of communication networks, among many others. Although technological advances have allowed the collection of these networks (often massive in scale), privacy concerns have severely restricted the ability of social scientists and others to study these networks. Valuable network data remains locked away in institutions, far from the scientists who are best equipped to exploit it, because the data is too sensitive to share.

The challenges of analyzing sensitive graph-structured data have recently received increased attention. It is now well-known that removing identifiers from a social network and releasing a "naively anonymized" isomorphic network can leave participants open to a range of attacks [1], [7], [17]. In response, a number of more sophisticated anonymization techniques have been proposed [3], [7], [13], [22], [23]. These techniques transform the graph—through the addition/removal of edges or clustering of nodes into groups—into a new graph which is then published. The analyst uses the transformed graph to derive estimates of various properties of the original.

The conventional goals of such algorithms are *privacy* and *utility*, although neither is satisfactorily achieved. Existing approaches provide anonymity, typically through transformations that make each node indistinguishable from others, but they lack rigorous guarantees of privacy. They rely on the assumption that the adversary's knowledge is limited [3], [13], [22] and/or fail to prove that the guarantee of anonymity ensures that private information is not disclosed [7], [13], [22], [23]. In terms of utility, while the transformed graph necessarily differs from the original, the hope is that it retains important structural properties of interest to the analyst. A drawback of existing techniques is that they lack formal bounds on the magnitude of the error introduced by the transformation. A common strategy for assessing utility is to measure familiar graph properties and compare these measures numerically to the original data. Empirically, it appears that with increasing anonymity, the graph is rapidly distorted and some metrics are systematically biased [3], [7], [13], [22], [23].

A final limitation of existing techniques is that few scale to the massive graphs that are now collected and studied. Most existing techniques have been evaluated on graphs of about 5,000-50,000 nodes, and may be difficult to scale much larger [7], [13], [22], [23].

In this work, we focus on a specific utility goal—estimating the degree distribution of the graph—and develop an algorithm that provides provable utility, strong privacy, and excellent scalability. The algorithm returns the degree distribution of the graph after applying a complex two-phase process of random perturbation. The error due to random noise is provably low, yet sufficient to prevent even powerful adversaries from extracting private information. The algorithm can scale to graphs with hundreds of millions of nodes. The techniques we propose here do not result in a published graph. Instead we release only an estimate of the true degree distribution to analysts.

We choose to focus on the degree distribution because it is one of the most widely studied properties of a graph. It influences the structure of a graph and processes that operate on a graph, and a diverse line of research has studied properties of random ensembles of graphs consistent with a known degree distribution [12], [14], [18].

The simple strategy of releasing the exact degree distribution fails to provide adequate privacy protection. Some graphs have unique degree distributions (i.e., all graphs matching this degree distribution are isomorphic) making the release of the degree distribution no safer than naive anonymization. In general, it is unclear how to determine what the degree distribution reveals about the structure of the graph. The problem is compounded when either the adversary has partial knowledge of graph structure, or the degree distribution is only one of several statistics published. Our goal is to design an approach that provides robust privacy protection against powerful adversaries and is compatible with releasing multiple statistics.

The algorithms proposed here satisfy a rigorous privacy standard called *differential privacy* [4]. It protects against any adversary, even one with nearly complete knowledge of the private data. It also composes well: one can release multiple statistics under differential privacy, so long as the algorithm for each statistic satisfies differential privacy. Thus while we focus on the degree distribution, additional statistics can be incorporated into the privacy framework.

While an existing differentially private algorithm [5] can be easily adapted to obtain noisy answers to queries about the degree distribution, the added noise introduces considerable error. In this work, we capitalize on a recent innovation in differentially private algorithms that has been shown to boost accuracy without sacrificing privacy [8]. The technique performs a post-processing step on the noisy answers, using the fact that the queries impose constraints on the space of possible answers to infer a more accurate result. We apply this technique to obtain an accurate estimate of the degree distribution of a graph.

Our contributions include the following.

- (Privacy) We adapt the definition of differential privacy to graph-structured data. We present several alternative formulations and describe the implications for the privacy-utility tradeoff inherent in them.
- (Scalability) We provide an implementation of the inference step of Hay et al. [8] that is linear, rather than quadratic, in the number of nodes. As a result of this algorithmic improvement the technique can scale to very large graphs, processing a 200 million node graph in 6 seconds. We also extend the inference technique to include additional constraints of integrality and non-negativity.
- (Utility) We assess utility of the resulting degree distribution estimate through comprehensive experiments on real and synthetic data. We show that: (i) estimates are extremely accurate under strong privacy parameters, exhibiting low bias and variance; (ii) for power-law graphs, the relative boost in accuracy from inference increases with graph size; (iii) an analyst can use the differentially private output to accurately assess whether the degree distribution follows a power-law.

These contributions are some of the first positive results in the private analysis of social network data, showing that a fundamental network analysis task can be performed accurately and efficiently, with rigorous guarantees of privacy.

Admittedly, the degree distribution is just one property of a graph, and there is evidence that a number of other properties are not constrained by the degree distribution alone [12], [14]. Nevertheless, it is hard to imagine a useful technique that distorts the degree distribution greatly. Thus it is important to know how accurately it can be estimated, independently of other properties. A long-term goal is to develop a differentially private algorithm for publishing a synthetic graph offering good utility for a range of analyses. Because of the degree distribution's profound influence on the structure of the graph, we believe that accurate estimation of it is a critical step towards that long-term goal.

II. Differential privacy for graphs

We first review the definition of differential privacy, and then propose how it can be adapted to graph data.

A. *Differential privacy*

To define differential privacy, we consider an algorithm A that operates on a private database I. The algorithm is randomized and the database is modeled as a set of records, each describing an individual's private information. Differential privacy formally limits how much a single individual record in the input can influence the output of the algorithm. More precisely, if I' is a *neighboring* database—i.e., one that differs from I by exactly one record—then an algorithm is differentially private if it is likely to produce the same output whether the input is I or I'.

The following is a formal definition of differential privacy, due to Dwork [4]. Let $nbrs(I)$ denote the set of neighbors of I—i.e., $I' \in nbrs(I)$ if and only if $|I \oplus I'| = 1$ where $\oplus$ denotes symmetric difference.[1]

Definition II.1 (ϵ-differential privacy)**.** *An algorithm A is ϵ-differentially private if for all instances I, any $I' \in nbrs(I)$, and any subset of outputs $S \subseteq Range(A)$, the following holds:*

$$Pr[A(I) \in S] \leq \exp(\epsilon) \times Pr[A(I') \in S]$$

where probability Pr is over the randomness of A.

The parameter ϵ measures the disclosure and is typically also an input to the algorithm. For example, the techniques used in this paper add random noise to their outputs, where the noise is a function of ϵ. The choice of ϵ is a matter of policy, but typically ϵ is "small," say at most 1, making the probability "almost the same" whether the input is I or I'.

An example illustrates why this protects privacy. Suppose a hospital wants to analyze the medical records of their patients and publish some statistics about the patient population. A patient may wish to have his record omitted from the study, out of a concern that the published results will reveal something about him personally and thus violate his privacy. The above definition assuages this concern because whether the individual opts-in or opts-out of the study, the probability of a particular output is almost the same. Clearly,

[1]Differential privacy has been defined inconsistently in the literature. The original definition (called indistinguishability) defines neighboring databases in terms of Hamming distance [5]. Note that ϵ-differential privacy (as defined above) implies 2ϵ-indistinguishability.

any observed output cannot reveal much about his particular record if that output is (almost) as likely to occur even when the record is excluded from the database.

B. Differential privacy for graphs

In the above definition, the database is a table whereas in the present work, the database is a graph. Below we adapt the definition of differential privacy to graphs.

The semantic interpretation of differential privacy rests on the definition of neighboring databases. Since differential privacy guarantees that the output of the algorithm cannot be used to distinguish between neighboring databases, what is being protected is precisely the difference between neighboring databases. In the above definition, a neighboring database is defined as the addition or removal of a single record. With the hospital example, the patient's private information is encapsulated within a single record. So differential privacy ensures that the output of the algorithm does not disclose the patient's medical history.

With network data, which is primarily about relationships among individuals, the correspondence between private data and database records is less clear. To adapt differential privacy to graphs, we must choose a definition for neighboring graphs and understand the privacy semantics of that choice. We propose three alternatives offering varying degrees of privacy protection.

We model the input as a graph, $G = (V, E)$, where V is a set of n entities and E is a set of edges. Edges are undirected pairs (u, v) such that u and v are members of V. (Results are easily extended to handle directed edges.) While the meaning of an edge depends on the domain—it could connote friendship, email exchange, sexual relations, etc.—we assume that it represents a sensitive relationship that should be kept private. The focus of the present work is concerned with graph structure, so the inclusion of attributes on nodes or edges is left for future work.

The first adaptation of differential privacy to graphs is mathematically similar to the definition for tables. Neighboring graphs are defined as graphs that differ by one "record." Given a graph G, one can produce a neighboring graph G' by either adding/removing an edge in E, or by adding/removing an *isolated* node in V. Restricting to isolated nodes ensures that the change to V does not require additional changes to E to make it consistent with V. Formally, G and G' are neighbors if $|V \oplus V'| + |E \oplus E'| = 1$. Because this adaptation allows neighboring graphs to differ by at most one edge, we call it **edge-differential privacy**.

An edge-differentially private algorithm protects individual edges from being disclosed. For some applications, edge-differential privacy seems to be a reasonable privacy standard. For example, consider the study of Kossinets and Watts [10], in which they analyze a graph derived from the email communication among students and faculty of a large university. What makes this dataset sensitive is that it reveals who emails whom; edge-differential privacy protects email relationships from being disclosed.

However, in some applications, it may be desirable to extend the protection beyond individual edges. For example, Klovdahl et al. [9] analyze the social network structure of "a population of prostitutes, injecting drug users and their personal associates." In this graph, an edge represents a sexual interaction or the use of a shared needle. Edges are clearly private information, but so too are other properties like node degree (the number of sexual/drug partners) and even membership in the network.

A second adaptation of differential privacy to graphs provides much stronger privacy protection. In **node-differential privacy**, two graphs are neighbors if they differ by at most one node and *all* of its incident edges. Formally, G and G' are neighbors if $|V \oplus V'| = 1$ and $E \oplus E' = \{(u, v) | u \in (V \oplus V') \text{ or } v \in (V \oplus V')\}$.

Node-differential privacy mirrors the "opt-in/opt-out" notion of privacy from the hospital example. It assuages any privacy concerns, as a node-differentially private algorithm behaves almost as if the participant did not appear in at all.

While node-differential privacy is a desirable privacy objective, it may be infeasible to design algorithms that are both node-differentially private and enable accurate network analysis. A differentially private algorithm must hide even the worst case difference between neighboring graphs, and this difference can be large under node-differential privacy. For instance the empty graph (n isolated nodes) is a neighbor of the star graph (a hub node connected to n nodes). We show in Sec III-A that estimates about node degrees are highly inaccurate under node-differential privacy.

To span the spectrum of privacy between edge- and node-differential privacy, we introduce an extension to edge-differential privacy that allows neighboring graphs to differ by more than a single edge. In **k-edge-differential privacy**, graphs G and G' are neighbors if $|V \oplus V'| + |E \oplus E'| \leq k$.

A larger setting of k leads to greater privacy protection. If $k = 1$, then k-edge-differential privacy is equivalent to edge-differential privacy. If $k = |V|$, then k-edge-differential privacy is even stronger than node-differential privacy, as the set of neighboring graphs under k-edge-differential privacy is a superset of the neighbors under node-differential privacy. If $1 < k < |V|$, then k-edge-differential privacy prevents the disclosure of aggregate properties of any subset of k edges. Notice that for those nodes whose degree is less than k, it provides essentially equivalent protection as node-differential privacy. Nodes whose degree is k or larger face more exposure. However, nodes with large degree also have greater influence on the structure of the graph. If our goal is to also allow analysts to accurately measure the graph structure, then it may be necessary to expose high degree nodes to greater privacy risk.

For the remainder of the paper, we will use k-edge-differential privacy as our privacy standard.

III. Estimating the degree distribution under differential privacy

In this section, we review the two techniques that form the basis of our approach. The first is a technique by Dwork et al. [5] for answering queries under differential privacy. The second is a recent technique [8] that post-processes the noisy output of the Dwork et al. algorithm to improve accuracy. We use these techniques to obtain a noisy estimate of the degree distribution of the graph. In the next section, we present a fast and scalable implementation of the latter technique.

A. Differentially-private query answering

Dwork et al. [5] give a general technique that allows an analyst to pose an arbitrary set of queries and receive noisy answers. The input to the algorithm is a sequence of queries $\mathbf{Q}$ where the answer to each query is a number in $\mathbb{R}$. The algorithm computes the true answer $\mathbf{Q}(I)$ to the queries on the private data and then adds random noise to the answers. The noise depends on the query sequence's *sensitivity*.

Definition III.1 (Sensitivity)**.** *The sensitivity of* $\mathbf{Q}$*, denoted* $S_{\mathbf{Q}}$*, is defined as*

$$S_{\mathbf{Q}} = \max_{I,I' \in nbrs(I)} \|\mathbf{Q}(I) - \mathbf{Q}(I')\|_1 .$$

The sensitivity of a query depends on how neighboring databases are defined. Intuitively, queries are more sensitive under node-differential privacy than edge-differential privacy, because the difference between neighboring graphs is larger under node-differential privacy.

However, regardless of how neighbors are defined, the following proposition holds. Let $\langle \text{Lap}(\sigma) \rangle^d$ denote a d-length vector of independent random samples from a Laplace distribution with mean zero and scale σ.

Proposition 1 ([5])**.** *Let* $\tilde{\mathbf{Q}}$ *denote the randomized algorithm that takes as input a database* I*, a query* $\mathbf{Q}$ *of length* d*, and some* $\epsilon > 0$*, and outputs*

$$\tilde{\mathbf{Q}}(I) = \mathbf{Q}(I) + \langle Lap(\mathbf{S_Q}/\epsilon) \rangle^d$$

Algorithm $\tilde{\mathbf{Q}}$ *satisfies* ϵ*-differential privacy.*

While this proposition holds for any of the adaptations of differential privacy, the accuracy of the answer depends on the magnitude of $S_{\mathbf{Q}}$, which differs across the adaptations. Using an example query, we illustrate the accuracy trade-offs between k-edge- and node-differential privacy. Let $\mathbf{D}_u$ denote the query that returns the degree of node u if $u \in V$ and otherwise returns -1.

Since the addition of Laplace noise introduces error of $\pm S_{\mathbf{Q}}/\epsilon$ in expectation, the accuracy of the answer depends on ϵ and the sensitivity of $\mathbf{D}_u$. Under k-edge-differential privacy, the sensitivity is k—in the worst case, neighboring graphs differ by k edges that are all adjacent to u, making u's degree differ by k. Thus we expect an accurate answer to $\mathbf{D}_u$ when k/ϵ is small relative to u's degree.

For node-differential privacy, however, the sensitivity is unbounded, unless we impose some restriction on the size of the input graph. If graphs are restricted to contain at most $n^\star$ nodes, then the sensitivity of $\mathbf{D}_u$ is $n^\star$—the worst case is a pair of neighboring graphs where u is connected to the other $n^\star - 1$ nodes in one graph and absent in the other. Since the magnitude of the error is the same as the range of $\mathbf{D}_u$, the answer is useless. This example suggests it is infeasible to accurately estimate node degrees under node-differential privacy because the difference in node degrees between neighboring graphs is too large.

Finally, we comment on the relationship between k and ϵ. As observed previously, an algorithm that provides ϵ-differential privacy for neighboring databases that differ by a single record also provides $k\epsilon$-differential privacy for neighboring databases that differ by at most k records [4]. To give a concrete example, suppose we run algorithm $\tilde{\mathbf{Q}}$ with $\epsilon = 0.01$ and compute $S_{\mathbf{Q}}$ assuming edge-differential privacy. Then as configured, $\tilde{\mathbf{Q}}$ satisfies k-edge ϵ-differential privacy for $k = 1$ and $\epsilon = 0.01$; it also satisfies k-edge ϵ-differential privacy for, say, $k = 10$ and $\epsilon = 0.1$, or even $k = 100$ and $\epsilon = 1.0$. In the next section and in the experiments, we assume that $k = 1$; however, the results hold for $k > 1$ provided that ϵ is appropriately scaled as in these examples.

B. Constrained inference

Hay et al. [8] introduce a post-processing technique that operates on the output of algorithm $\tilde{\mathbf{Q}}$. It can be seen as an improvement on the basic algorithm of Dwork et al. [5] that boosts accuracy without sacrificing privacy. The main idea behind the approach is to use the semantics of a query to impose constraints on the answer. While the true answer $\mathbf{Q}(I)$ always satisfies the constraints, the noisy answer that is output by $\tilde{\mathbf{Q}}$ may violate them. Let $\tilde{q}$ denote the output of $\tilde{\mathbf{Q}}$. The constrained inference process takes $\tilde{q}$ and finds the answer that is "closest" to $\tilde{q}$ and also satisfies the constraints of the query. Here "closest" is measured in L_2 distance and the consistent output is called the *minimum* L_2 *solution*.

Definition III.2 (Minimum L_2 solution)**.** *Let* $\mathbf{Q}$ *be a query sequence with a set of constraints denoted* $\gamma_{\mathbf{Q}}$*. A minimum* L_2 *solution is a vector* $\overline{q}$ *that satisfies the constraints* $\gamma_{\mathbf{Q}}$ *and minimizes* $||\tilde{q} - \overline{q}||_2$.

As discussed in [8], the technique has no impact on privacy since it requires no access to the private database, only $\tilde{q}$, the output of the differentially private algorithm.

This technique can be used to obtain an accurate estimate of the degree distribution of the graph. Our approach is to ask a query for the graph's degree *sequence*, a sequence of non-decreasing numbers corresponding to the degrees of the graph's vertices. Of course, a degree sequence can be converted to a degree *distribution* by simply counting the frequency of each degree. The advantage of the degree sequence query is that it is constrained, as explained below.

We now define $\mathbf{S}$, the degree sequence query.[2] Let $deg(i)$ return the i^{th} smallest degree in G. Then, the degree sequence of the graph is the sequence $\mathbf{S} = \langle deg(1) \dots deg(n) \rangle$. Under 1-edge-differential privacy, the sensitivity of $\mathbf{S}$ is 2: suppose a neighboring graph has an additional edge between two nodes of degree d, d', then two values in $\mathbf{S}$ are affected, the largest i such that $\mathbf{S}[i] = d$ becomes $d+1$, similarly for d'. Let $\tilde{\mathbf{S}}$ denote the application of the algorithm described in Proposition 1 to the $\mathbf{S}$ query. A random output of $\tilde{\mathbf{S}}$ is denoted $\tilde{s}$.

Query $\mathbf{S}$ is constrained because the degrees are positioned in sorted order. The constraint set for $\mathbf{S}$ is denoted $\gamma_{\mathbf{S}}$, and contains the inequalities $\mathbf{S}[i] < \mathbf{S}[i+1]$ for $1 \leq i < n$. The following theorem gives the minimum L_2 solution for $\tilde{s}$.

Theorem 1 ([8])**.** *Given $\tilde{s}$, let $M[i,j]$ be the average of the subsequence $\tilde{s}[i,j]$: $M[i,j] = \sum_{k=i}^{j} \tilde{s}[k]/(j-i+1)$. The minimum L_2 solution $\overline{s}$ is unique and is defined as $\overline{s}[k] = \max_{1 \leq i \leq k} \min_{i \leq j \leq n} M[i,j]$.*

We use $\overline{\mathbf{S}}$ to refer to the algorithm that first computes $\tilde{\mathbf{S}}$ and then applies constrained inference to obtain the above minimum L_2 solution. The following example provides an intuition for how $\overline{\mathbf{S}}$ uses sort constraints to reduce the error.

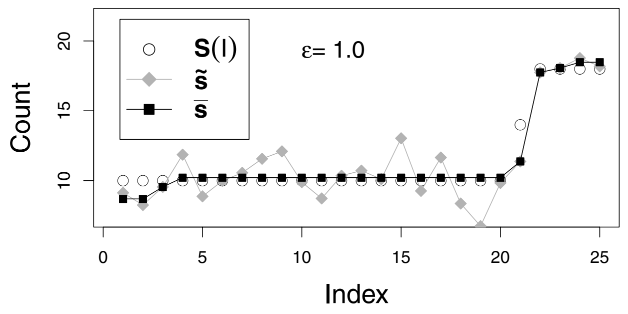


Figure 1. Example sequence $\mathbf{S}(I)$, noisy estimate $\tilde{s} = \tilde{\mathbf{S}}(I)$, and constrained inference estimate $\overline{s} = \overline{\mathbf{S}}(I)$.

Example 1. *Figure 1 shows a degree sequence $\mathbf{S}(I)$ for a 25 node graph, along with a sampled $\tilde{s}$ and inferred $\overline{s}$. While the values in $\tilde{s}$ deviate considerably from $\mathbf{S}(I)$, $\overline{s}$ lies very close to the true answer. In particular, for subsequence $[1, 20]$, the true sequence $\mathbf{S}(I)$ is uniform and the constrained inference process effectively averages out the noise of $\tilde{s}$. The twenty-first position is a unique degree in $\mathbf{S}(I)$ and constrained inference does not refine the noisy answer, i.e., $\overline{s}[21] = \tilde{s}[21]$.*

As suggested by the example, the error of $\overline{\mathbf{S}}$ can be lower than that of $\tilde{\mathbf{S}}$, particularly when the degree sequence contains subsets of nodes with the same degree. Hay et al. [8] theoretically analyze the error in terms of mean squared error. For a query $\tilde{\mathbf{Q}}$, the mean square error is $MSE(\tilde{\mathbf{Q}}) = \mathbb{E}[||\tilde{\mathbf{Q}} - \mathbf{Q}(I)||_2] = \sum_i \mathbb{E}[(\tilde{\mathbf{Q}}[i] - \mathbf{Q}[i])^2]$, where the expectation is over the randomness of $\tilde{\mathbf{Q}}$.

Theorem 2 ([8])**.** *Let d be the number of unique degrees in $\mathbf{S}(I)$. Then $MSE(\overline{\mathbf{S}}) = O(d \log^3 n/\epsilon^2)$. In comparison, $MSE(\tilde{\mathbf{S}}) = \Theta(n/\epsilon^2)$.*

This result shows that error scales linearly with the number of unique degrees, rather than the number of nodes. While this is a promising result, it is not clear what lower MSE in the degree *sequence* means for an analyst interested in studying the degree *distribution*. Furthermore, it is unclear whether the closed form solution described in Theorem 1 can be computed efficiently. These issues are addressed next.

IV. Algorithm for computing $\overline{\mathbf{S}}$

We now describe an efficient algorithm for applying constrained inference to the noisy sequence $\tilde{s}$. A straight-forward approach for computing $\overline{s}$ is to construct a dynamic program based the definition of $\overline{s}$ from Theorem 1. However, it requires linear time to compute each $\overline{s}[k]$, making the total runtime quadratic, infeasible for many large graphs. We present a novel algorithm that reduces the complexity to linear time. The algorithm is a dynamic program that works backwards from the end of the sequence, constructing a partial solution for a subsequence of $\overline{s}$. By working backwards, we can reuse computations from previous steps so updating the partial solution requires only (amortized) constant time rather than linear time.

Before describing the algorithm, we introduce some notation and restate the minimum L_2 solution of Theorem 1 using this notation. Let the minimum cumulative average at k be denoted as $\mathbf{M}_k = \min_{1 \leq j \leq k} M[k,j]$. Then we can rewrite the solution at $\overline{s}[k]$ as follows:

$$\overline{s}[k] = \max_{1 \leq i \leq k} \min_{i \leq j \leq n} M[i,j] = \max_{1 \leq i \leq k} \mathbf{M}_i \tag{1}$$

The basic idea behind the linear time algorithm is to construct $\overline{s}$ incrementally, starting at the *end* of the sequence and working backwards toward the beginning. At each step ℓ, the algorithm maintains a partial solution for the *subsequence* $\tilde{s}[\ell, \dots, n]$—meaning that the sort constraints are only enforced on this subsequence and the rest of $\tilde{s}$ is ignored. At each step, the subsequence is extended to include another element of $\tilde{s}$ and the partial solution is updated accordingly.

We denote the partial solution as $\bar{r}^{\ell}$, and from Equation 1, the value of $\bar{r}^{\ell}$ at position k is equal to

$$\bar{r}^{\ell}[k] = \max_{\ell \leq i \leq k} \min_{i \leq j \leq n} M[i,j] = \max_{\ell \leq i \leq k} \mathbf{M}_i \tag{2}$$

Observe that partial solution $\bar{r}^1$ is equal to the desired $\overline{s}$.

[2]We simplify the presentation, as well as later experiments, by assuming that n is known. In practice, n is unlikely to be sensitive, especially for large networks. Alternatively, one could derive a very accurate estimate of n using Proposition 1 and then adjust $\mathbf{S}$ accordingly. This would result in a small amount of additional error in the estimate for the number of low degree nodes.

Given a partial solution $\bar{r}^{\ell}$, we can extend the solution to $\ell-1$ by extending the subsequence to include the observation $\tilde{s}[\ell-1]$ and updating the partial $\bar{r}^{\ell}$ to obtain $\bar{r}^{\ell-1}$. There are two components of the update procedure. First, we determine the value for the new observation at position $\ell-1$. From Equation 2 the solution is simply the minimum cumulative average starting at $\ell-1$; i.e., $\bar{r}^{\ell-1}[\ell-1] = \mathbf{M}_{\ell-1}$.

The second step is to check whether including the $(\ell-1)^{th}$ element requires updating the existing solution for positions $\ell, \ldots, n$. From Equation 2, we can see that we must update any $k = \ell, \ldots, n$ where the current solution $\bar{r}^{\ell}[k]$ is smaller than the new value at $\ell-1$ position, $\bar{r}^{\ell-1}[\ell-1]$:

$$\begin{aligned} \bar{r}^{\ell-1}[k] &= \max_{\ell-1 \leq i \leq k} \mathbf{M}_i \\ &= \max\left(\mathbf{M}_{\ell-1}, \max_{\ell \leq i \leq k} \mathbf{M}_i\right) \\ &= \max\left(\bar{r}^{\ell-1}[\ell-1], \bar{r}^{\ell}[k]\right) \end{aligned}$$

Thus, each step requires first computing $\mathbf{M}_{\ell-1}$ and then updating the existing solution for positions $\ell, \ldots, n$. We show next that we can use the partial solution $\bar{r}^{\ell}$ to simplify the cost of finding $\mathbf{M}_{\ell-1}$. While computing an individual $\mathbf{M}_{\ell-1}$ can take linear time in the worst-case, the *amortized* cost is only $O(1)$. Furthermore, we store partial solution $\bar{r}^{\ell}$ in such a way that once $\mathbf{M}_{\ell-1}$ is found, no additional work is required to update the solution.

Given a partial solution $\bar{r}^{\ell}$, break it into subsequences such that each subsequence is uniform and has maximal length. Let J^{ℓ} be the set of indexes marking the ends of the uniform subsequences. E.g., if all of the elements in $\bar{r}^{\ell}$ are distinct, then $J^{\ell} = \{\ell, \ldots, n\}$; if the values of $\bar{r}^{\ell}$ are all the same, then $J^{\ell} = \{n\}$.

The following theorem shows how we can use $\bar{r}^{\ell}$ to compute the minimum cumulative average $\mathbf{M}_{\ell-1}$. Recall that $\mathbf{M}_{\ell-1} = \min_{\ell-1 \leq k \leq n} M[\ell-1, j]$. Let j^* denote the end index of this minimum average, i.e., $j^* = \arg\min_{\ell-1 \leq j \leq n} M[\ell-1, j]$.

Theorem 3. *Given $\bar{r}^{\ell}$ and a corresponding J^{ℓ}, one of the following conditions holds. Either, $j^* = \ell-1$ or $j^* \in J^{\ell}$. Furthermore, the set $J^{\ell-1} = \{j^*\} \cup \{j | j \in J^{\ell}$ and $j > j^*\}$.*

Theorem 3 shows that the set J^{ℓ} can be used to find the minimum cumulative average for $\ell-1$. The algorithm proceeds by considering the indexes in J^{ℓ} in ascending order. The initial cumulative average is set to $M[\ell-1, \ell-1]$, and the average is extended to the next endpoint in J^{ℓ} so long as it reduces the average. When the average increases, the algorithm terminates the search. The following Lemma implies that this will be j^*.

Lemma 1. *Let j^* be defined as above, then $\max_{\ell \leq j \leq j^*} \mathbf{M}_j \leq \mathbf{M}_{\ell-1} \leq \mathbf{M}_{j^*+1}$.*

During the computation, we need only store the set J^{ℓ} instead of the entire sequence of $\bar{r}^{\ell}$. Updating J^{ℓ} is much faster than updating $\bar{r}^{\ell}$ and from J^{ℓ} we can easily reconstruct the solution $\bar{r}^{\ell}$. The details are shown in Algorithm 1, which computes J^{ℓ} (lines 1-8) for $\ell = n, \ldots, 1$. Then, it constructs $\bar{s}$ using J^1 (lines 10-16).

Algorithm 1 An algorithm for computing $\bar{s}$ given $\tilde{s}$

```
1:  $J \leftarrow \emptyset$, $J.push(n)$
2:  for $k$ from $n$ to 1 do
3:      $j^* \leftarrow k$, $j \leftarrow J.top()$
4:      while $J \neq \emptyset$ and $M[j^*+1, j] \leq M[k, j^*]$ do
5:          $j^* \leftarrow j$, $J.pop()$, $j \leftarrow J.top()$
6:      end while
7:      $J.push(j^*)$
8:  end for
9:  $b \leftarrow 1$
10: while $J \neq \emptyset$ do
11:     $j^* \leftarrow J.pop()$
12:     for $k$ from $b$ to $j^*$ do
13:         $\bar{s}[k] \leftarrow M[b, j^*]$
14:     end for
15:     $b \leftarrow j^* + 1$
16: end while
17: return $\bar{s}$
```

Example 2. *The following table shows a sample input $\tilde{s}$ along with the computations used to compute $\bar{s}$.*

k	$\tilde{s}[k]$	$\bar{s}[k]$	$\arg\min_{k \leq j \leq n} M[k, j]$
1	1	1	$M[1, 1]$
2	9	5	$M[2, 5]$
3	4	5	$M[3, 4]$
4	3	5	$M[4, 4]$
5	4	5	$M[5, 5]$

The algorithm begins with $J^5 = \{5\}$. Stack J^4 becomes $\{4, 5\}$ after $\tilde{s}[4]$ is considered since $\tilde{s}[4] < \mathbf{M}_5$. When it comes to $\tilde{s}[3]$, the stack J^3 is $\{4, 5\}$ since $\mathbf{M}_3 = M[3, 4]$. When $\tilde{s}[2]$ is added since $M[2, 5] < M[2, 4]$, we know $\mathbf{M}_2 = M[2, 5]$ and $J^2 = \{5\}$. Then $\tilde{s}[1]$ arrives and makes J^1 equal to $\{1, 5\}$. The algorithm rebuilds $\bar{s}$ as $1, 5, 5, 5, 5$.

The time complexity of Algorithm 1 is O(n). First, once the stack J is completed, reconstructing $\bar{s}$ (lines 10-16) clearly takes $O(n)$. Second, the complexity of computing stack J is also linear: it can seen by considering the number of times that line 5 executes. Since each execution of line 5 reduces the size of stack J by 1 and there are only $O(n)$ push operations on stack J, we know line 5 executes at most $O(n)$ times. In the worst-case the stack J can require $O(n)$ space. However, only the top of the stack is accessed during computations and the rest can be written to disk as needed.

Incorporating additional constraints: The output of the algorithm, $\bar{s}$, may be non-integral and include negative numbers, when in fact the values of the true degree sequence are constrained to lie in $\{0, \ldots, n-1\}$. We would like a solution that respects these constraints since they are required in many applications. Theorem 4 shows that such a solution is computed from $\overline{s}$ by simply rounding.

Theorem 4. *Let $\gamma'_{\mathbf{S}}$ be the constraint set $\gamma_{\mathbf{S}}$ augmented with the additional constraint that each count be an integer in the set $\{0, \ldots, n-1\}$. Given $\overline{s}$, the minimum L_2 solution for constraint set $\gamma_{\mathbf{S}}$, let $\overline{s}'$ denote the sequence derived from $\overline{s}$ in which each element $\overline{s}[k]$ is rounded to the nearest value in $\{0, \ldots, n-1\}$. Then $\overline{s}'$ is a* minimum L_2 solution *that satisfies the constraint set $\gamma'_{\mathbf{S}}$.*

V. Experiments

The goals of our experiments are two-fold. First, we assess the scalability of the constrained inference algorithm introduced in Section IV. Second, we want to understand the tradeoff between privacy and utility. To do so, we first characterize how the noise introduced for privacy distorts the degree distribution. Then, using several metrics to compare distributions, we assess how accurately the distributions derived from $\overline{\mathbf{S}}$ and $\tilde{\mathbf{S}}$ approximate the true degree distribution. The accuracy depends on ϵ, which governs the amount of privacy. We also assess how the privacy-utility tradeoff changes with the size of the graph (does a bigger graph allow for better utility at a fixed level of privacy?). Finally, we consider one of the most common tasks that analysts perform on a degree distribution: assessing whether it follows a power-law. We measure how the added noise affects the fit of a power-law model.

We experiment on both synthetic and real datasets. The real datasets are derived from crawls of four online social networking sites: **Flickr** ($\approx$1.8M nodes), **LiveJournal** ($\approx$5.3M), **Orkut** ($\approx$3.1M), and **YouTube** ($\approx$1.1M) [16]. To the best of our knowledge, these are the largest publicly available social network datasets. The synthetic datasets include **Random**, a classical random graph, which has a Poisson degree distribution ($\lambda = 10$), and **Power**, a random graph with a power-law degree distribution ($\alpha = 1.5$).

A. Scalability

Figure 2 shows that the runtime of Algorithm 1 scales linearly and is extremely fast. The left figure shows the runtime on the real datasets and the right figure shows the runtime on even larger synthetic datasets of up to 200M nodes. In addition to **Random** and **Power**, we include two non-random synthetic distributions, corresponding to the best- and worst-case inputs for the runtime of the algorithm. The best-case is **Regular**, a uniform degree distribution (all nodes have degree 10), the worst-case is **Natural**, a distribution having one occurrence of each degree in $\{0, \ldots, n-1\}$.

The small variation in runtime across datasets shows that it is not particularly sensitive to the type of degree distribution. Furthermore, it is extremely fast: processing a 200 million node graph takes less than 6 seconds. In contrast, the algorithm of Hay et al. [8] takes 20 minutes for a 1 million node graph and over an hour to process a 2 million node graph. The efficiency of the improved algorithm makes the constrained inference approach practical for large graphs.

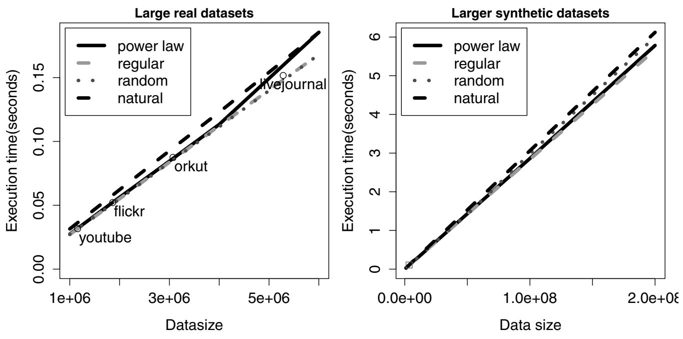


Figure 2. Runtime of Algorithm 1 on real (left) and larger synthetic datasets (right).

B. Utility

We use two measures we use to assess accuracy. First, we use the Kolmogorov-Smirnoff (KS) statistic, a measure used to test whether two samples are drawn from the same distribution. Let the empirical cumulative distribution function (CDF) of sample $X = X_1, \ldots, X_n$ be defined as $F_X(x) = \frac{1}{n}\sum_{i=1}^{n} \mathbf{I}[X_i \le x]$. Then the KS statistic between X and Y is $KS(X,Y) = \max_x |F_X(x) - F_Y(x)|$.

The KS statistic is insensitive to differences in the tails of the two distributions, so we also use the Mallows distance (aka Earth Mover's distance) to capture deviations in the tail. Given samples X and Y each of size n, with $X_{(i)}$ denoting the i^{th} largest sample in X, the Mallows p-distance is

$$\text{Mallows}_p(X,Y) = \left(\frac{1}{n}\sum_{i=1}^{n} \left|X_{(i)} - Y_{(i)}\right|^p\right)^{1/p}$$

An example shows how Mallows distance is more sensitive than the KS statistic to the tail of the distribution. Consider three graphs A, B, and C in which all nodes have degree 1, except in B one node has degree 2 and in C one node has degree $n-1$. The KS statistic between A and either B or C is $O(n^{-1})$. The Mallows distance ($p = 1$) between A and B is $O(n^{-1})$, but between A and C, the Mallows distance is $O(1)$, capturing the difference between their largest degrees.

A visual comparison of distributions: Figure 3(a) shows the true degree distribution along with the differentially private approximations, revealing that $\overline{\mathbf{S}}$ produces a very accurate approximation while $\tilde{\mathbf{S}}$ does not. The distributions are represented using the complementary CDF (CCDF), denoted CF and defined as $CF_X(x) = 1 - F_X(x)$. Thus, each line shows what fraction of nodes have a degree greater than the given value on the x-axis. Abusing notation, we use $\mathbf{S}(I)$, $\tilde{s}$, and $\overline{s}$, which are all degree sequences, to refer to their corresponding degree distributions. Thus, the line labeled $\mathbf{S}(I)$ refers to the true degree distribution and the lines labeled $\tilde{s}$ and $\overline{s}$ refer to the degree distributions derived from differentially private sequences $\tilde{s}$ and $\overline{s}$ (here $\epsilon = 0.01$).

Figure 3(a) shows that noise added to produce $\tilde{s}$ substantially distorts the degree distribution. In contrast, $\overline{s}$ is a much more accurate approximation of $\mathbf{S}(I)$. While $\overline{s}$ exhibits some deviations from the true distribution, the deviations appear to oscillate around the true distribution. This demonstrates that, by exploiting the sort constraints, constrained inference can filter out much of the noise in $\tilde{s}$.

Bias & variance analysis: In addition to showing individual samples $\tilde{s}$ and $\overline{s}$, we also analyze the bias and variance of randomized algorithms $\tilde{\mathbf{S}}$ and $\overline{\mathbf{S}}$. More precisely, we measure bias of $\overline{\mathbf{S}}$ as the expected difference between the CCDFs of $\overline{\mathbf{S}}$ and $\mathbf{S}(I)$ for each degree—i.e., $bias_{\overline{\mathbf{S}}}(x) = \mathbb{E}[CF_{\overline{\mathbf{S}}}(x) - CF_{\mathbf{S}(I)}(x)]$ where the expectation is over the randomness in $\overline{\mathbf{S}}$. The variance of $\overline{\mathbf{S}}$ is $var_{\overline{\mathbf{S}}}(x) = \mathbb{E}[(CF_{\overline{\mathbf{S}}}(x) - \mathbb{E}[CF_{\overline{\mathbf{S}}}(x)])^2]$. We focus on $\overline{\mathbf{S}}$ because it is evident from Figure 3(a) that $\tilde{\mathbf{S}}$ exhibits substantial bias.

We evaluate the bias/variance of $\overline{\mathbf{S}}$ empirically thru repeated sampling. The results are shown in the bottom panel of Figure 3(a). The y-axis is the difference in cumulative probability between $\mathbf{S}$ and $\overline{\mathbf{S}}$, $CF_{\overline{\mathbf{S}}}(x) - CF_{\mathbf{S}(I)}(x)$. The line shows the average difference (bias) and the error bars depict the standard deviation from the average (square root of variance). The line remains near 0, suggesting that $\overline{\mathbf{S}}$ may be an unbiased or nearly unbiased estimator of $\mathbf{S}(I)$. The variance peaks wherever the CCDF exhibits steepest change.

Accuracy vs. privacy: Figures 3(b) and 3(c) show the relationship between privacy and the accuracy for two measures of accuracy—KS in 3(b), Mallows in 3(c). We report the average accuracy over 10 trials (random samplings of $\tilde{s}$). The amount of privacy is controlled by the parameter ϵ (horizontal axis)—smaller ϵ corresponds to stronger privacy.

The results show that $\overline{\mathbf{S}}$ is uniformly more accurate than $\tilde{\mathbf{S}}$, across all datasets, settings of ϵ, and both measures of accuracy. Furthermore, for low settings of ϵ (stronger privacy), the difference in accuracy is greater, suggesting that the benefit of constrained inference increases with privacy.

Also shown in the figure is the accuracy of an estimate based on random sampling (10% of the degrees are sampled uniformly at random). While sampling does not provide differential privacy, it can serve as a useful reference point. Sampling has very low KS distance (as expected), but higher Mallows distance because random sampling is unlikely to select the high degree nodes in the tail. In fact, sampling has higher Mallows distance than $\overline{\mathbf{S}}$ (except on **Random**, which is a distribution without long tails). Since analysts often cannot obtain complete graphs and must rely on samples, this result suggests that the additional error due to privacy can be small compared to the sampling error.

Accuracy vs. size: Figure 3(d) shows how the privacy-utility tradeoff of $\overline{\mathbf{S}}$ improves as the graph increases in size. The figure reports accuracy on **Power** graphs of varying size, from 10K to 5M nodes. The results show a clear separation between $\tilde{\mathbf{S}}$ and $\overline{\mathbf{S}}$: as the size of the graph increases, the accuracy of $\tilde{\mathbf{S}}$ remains constant whereas the accuracy of $\overline{\mathbf{S}}$ improves. Thus, with $\overline{\mathbf{S}}$, larger datasets yield either more privacy (given a fixed accuracy target, we can lower ϵ) or better utility (higher accuracy for fixed ϵ).

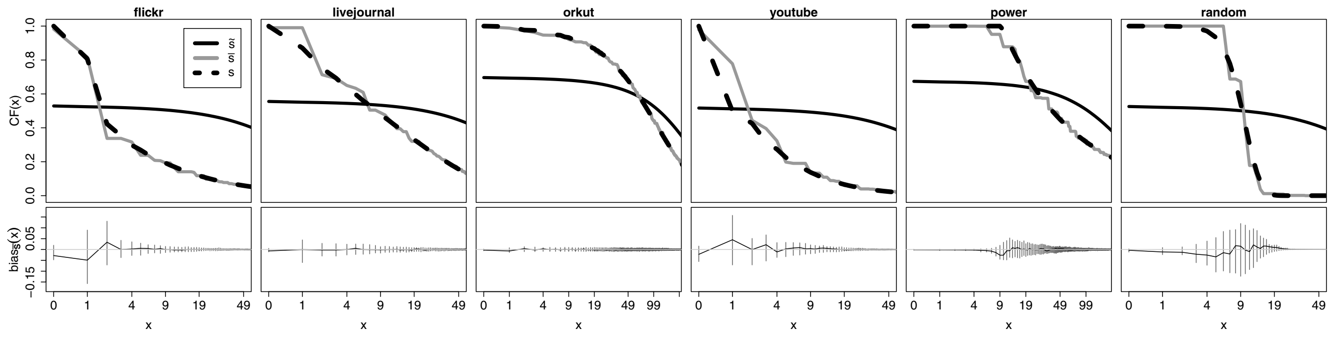
The accuracy of $\tilde{\mathbf{S}}$ does not improve with graph size because random noise is added to each degree, thus the average error per degree does not change with the size of the graph. However, as Example 1 showed, $\overline{\mathbf{S}}$ can be very accurate when the degree sequence contains long subsequences of uniform degrees. As the graph size increases, accuracy improves because the subsequences of uniform degree grow longer (in a power-law graph, the expected proportion of nodes with a given degree is a constant independent of n).

In this experiment, the parameters k and ϵ of the privacy condition remain fixed as n increases. If node degrees were to increase with graph size, then holding ϵ fixed would mean that while the absolute disclosure remains fixed, the *relative* disclosure about a node's neighborhood would increase with n. When evaluating graph models where node degrees increase with size (e.g., forest-fire graphs [11]), it may be appropriate to decrease ϵ as n increases.

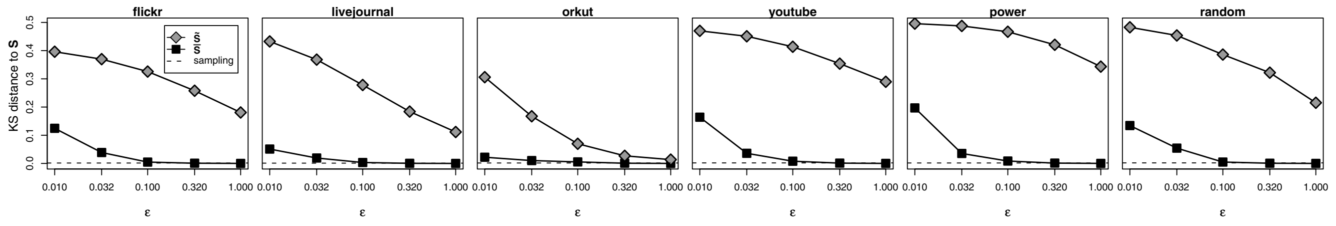
Modeling power-law distributions: Our final experiment assesses how accurately the analyst can estimate the parameters of a power-law model using $\tilde{\mathbf{S}}$ or $\overline{\mathbf{S}}$. The experiment is designed as follows. First, we sample a **Power** graph with parameters $\theta = (\alpha = 1.5, x_{min} = 10)$. We fix this as the true degree distribution. Then we sample $\tilde{s}$ and $\overline{s}$ and derive corresponding distributions. To each of these three degree distributions, we fit a power-law model using maximum likelihood [2]. The result is three different estimates for the parameters θ, which we denote $\hat{\theta}$, $\tilde{\theta}$, and $\bar{\theta}$ respectively. We are interested in comparing the model fit to the true degree distribution, $\hat{\theta}$, to the models fit under differential privacy, $\tilde{\theta}$ and $\bar{\theta}$.

The individual parameter estimates are shown in the middle and right plot of Figure 3(e), but the leftmost plot provides a holistic assessment of the model fit. It assesses model fit using the D statistic of Clauset et al. [2] which measures the KS statistic on the power-law tail of the distribution. We consider two variants of this measure: in one, the tail is defined by the estimate of x_{min} under $\tilde{s}$ or $\overline{s}$; in the other, x_{min} is based on the true x_{min}.

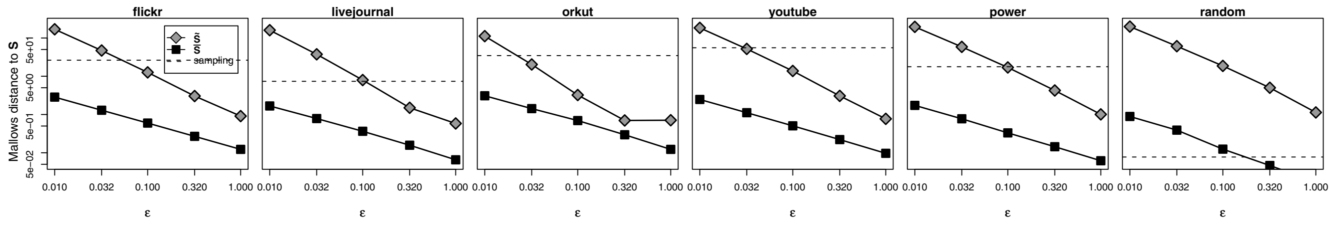
The plots reveal that using either $\tilde{\mathbf{S}}$ or $\overline{\mathbf{S}}$, the analyst will estimate a model that has a close fit to the tail of the original (power-law) distribution, when the tail is defined by the x_{min} *estimated on the noisy distribution*. However, it also shows that the size of the tail is under-estimated (the power-law behavior becomes apparent only for large degrees). If we compare the models based on how well they fit the true tail of the power-law distribution (solid lines of leftmost plot), we see that $\tilde{\mathbf{S}}$ has considerable distortion (note the log-scale) while $\overline{\mathbf{S}}$ is reasonably accurate even at small ϵ.



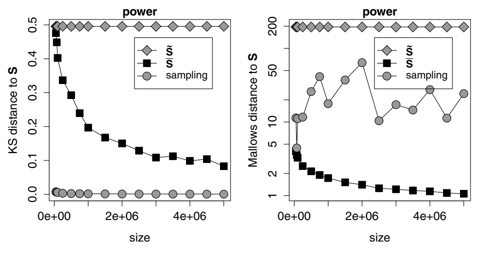
(a) Complementary CDFs of $\mathbf{S}(I)$, $\tilde{s}$ and $\bar{s}$ (top). Bias of $\overline{\mathbf{S}}$ (bottom)



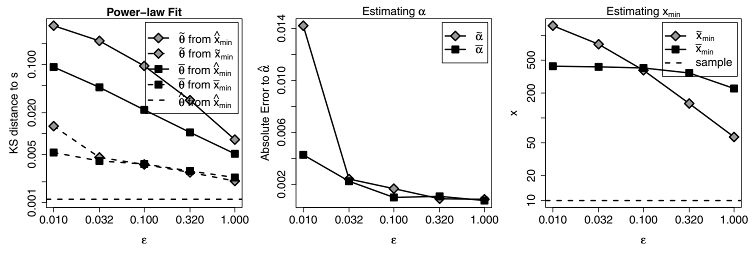
(b) Privacy (ϵ) vs. Accuracy (KS distance)



(c) Privacy (ϵ) vs. Accuracy (Mallows distance with $p = 2$)



(d) Size vs. Accuracy for fixed $\epsilon = 0.01$



(e) Accuracy of estimating power-law model using $\tilde{\mathbf{S}}$, $\overline{\mathbf{S}}$.

Figure 3. The privacy-utility tradeoff of two differentially private estimators $\tilde{\mathbf{S}}$ and $\overline{\mathbf{S}}$.

VI. Related work

The constrained inference technique that underlies this work was originally proposed in [8]. That work focuses on using constraints to improve accuracy for a variety of counting queries. While applications to degree estimation were recognized by the authors, a number of issues necessary for practical network analysis were left open. The present paper shows that inference only requires linear time and that the framework can be extended to include the additional constraints of integrality and non-negativity constraints. Further, we resolve open questions about the practical utility of the algorithm—showing that it scales to large graphs and produces accurate, nearly unbiased estimates of the degree distribution—and provide a more complete characterization of the privacy-utility tradeoffs.

Most prior work focuses on protecting privacy through *anonymization*, transforming the graph so that nodes cannot be re-identified [3], [7], [13], [22], [23]. The output is a published graph which the analyst can study in place of the original. While publishing a graph allows a broader range of analysis, anonymization is a much weaker notion of privacy and is vulnerable to attack (e.g., [6], [21]).

Furthermore, for the analyst interested in studying the degree distribution, these techniques may not scale to large graphs and can introduce considerable distortion. For example, the technique of Liu & Terzi [13], which is the most scalable approach, appears to considerably bias power-law degree distributions, reducing the power-law coefficient by 0.5 for reasonable settings of the privacy parameters. Our estimates have much smaller error (e.g., 0.004 at $\epsilon = 0.01$) and satisfy a much stronger privacy condition.

Differential privacy has been an active area of research (Dwork [4] provides a survey). Enabling accurate analysis of social networks is an often mentioned goal, but we are aware of only a few concrete results: techniques for computing properties such as clustering coefficient that have high sensitivity [19], [20] and a forthcoming approach that estimates the parameters of a random graph model [15].

VII. Conclusion

For the task of approximating the degree distribution of a private social network, we present an algorithm that protects privacy, scales to large graphs, and produces extremely accurate approximations. Our approach satisfies differential privacy, which means that, unlike approaches based on anonymization, it provides extremely robust protection, even against powerful adversaries. Finally, given the importance of the degree distribution to the structure of a graph, we believe that our techniques are a critical first step towards the ultimate goal of publishing synthetic graphs that are both accurate and ensure differential privacy.

Acknowledgments

Hay and Jensen were supported by the Air Force Research Laboratory and the Intelligence Advanced Research Projects Activity (IARPA), under agreement number FA8750-07-2-0158. Hay, Miklau, and Li 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 (IARPA), or the U.S. Government.

References

[1] L. Backstrom, C. Dwork, and J. Kleinberg. Wherefore art thou R3579X? . In *WWW*, 2007.

[2] A. Clauset, C. R. Shalizi, and M. Newman. Power-law distributions in empirical data. *SIAM Review*, 2009.

[3] G. Cormode, D. Srivastava, S. Bhagat, and B. Krishnamurthy. Class-based graph anonymization for social network data. In *VLDB*, 2009.

[4] C. Dwork. Differential privacy: A survey of results. In *TAMC*, 2008.

[5] C. Dwork, F. McSherry, K. Nissim, and A. Smith. Calibrating noise to sensitivity in private data analysis. In *TCC*, 2006.

[6] S. Ganta, S. Kasiviswanathan, and A. Smith. Composition attacks and auxiliary information in data privacy. In *KDD*, 2008.

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

[8] M. Hay, V. Rastogi, G. Miklau, and D. Suciu. Boosting the accuracy of differentially-private histograms through consistency. arXiv:0904.0942, 2009.

[9] A. Klovdahl, J. Potterat, D. Woodhouse, J. Muth, S. Muth, and W. Darrow. Social networks and infectious disease: the Colorado Springs study. *Soc Sci Med*, 1994.

[10] G. Kossinets and D. Watts. Empirical Analysis of an Evolving Social Network. *Science*, 311(5757):88–90, 2006.

[11] J. Leskovec, J. Kleinberg, and C. Faloutsos. Graphs over time: Densification laws, shrinking diameters and possible explanations. In *KDD*, 2005.

[12] L. Li, D. Alderson, J. Doyle, and W. Willinger. Towards a theory of scale-free graphs: Definition, properties, and implications. *Internet Mathematics*, 2005.

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

[14] P. Mahadevan, D. Kiroukov, K. Fall, and A. Vahdat. Systematic topology analysis and generation using degree correlations. In *SIGCOMM*, 2006.

[15] D. Mir and R. Wright. A differentially private graph estimator. In *International Workshop on Privacy Aspects of Data Mining*, 2009.

[16] A. Mislove, M. Marcon, K. P. Gummadi, P. Druschel, and B. Bhattacharjee. Measurement and analysis of online social networks. In *IMC*, 2007.

[17] A. Narayanan and V. Shmatikov. De-anonymizing social networks. In *IEEE Symposium on Security and Privacy (Oakland)*, 2009.

[18] M. Newman, S. Strogatz, and D. Watts. Random graphs with arbitrary degree distributions and their applications. *Physical Review E*, 64(2), 2001.

[19] K. Nissim, S. Raskhodnikova, and A. Smith. Smooth sensitivity and sampling in private data analysis. In *STOC*, 2007.

[20] V. Rastogi, M. Hay, G. Miklau, and D. Suciu. Relationship privacy: Output perturbation for queries with joins. In *PODS*, 2009.

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

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

[23] L. Zou, L. Chen, and T. Ozsu. K-Automorphism: A general framework for privacy preserving network publication. In *VLDB*, 2009.