Recursive Partitioning and Summarization: A Practical Framework for Differentially Private Data Publishing

Wahbeh Qardaji, Ninghui Li
Purdue University
305 N. University Street,
West Lafayette, IN 47907, USA
{wqardaji, ninghui}@cs.purdue.edu

ABSTRACT

In this paper we investigate Recursive Partitioning and Summarization (RPS), a practical framework for data publishing that satisfies differential privacy. While there have been several negative results concerning non-interactive differentially private data release, we show that such results do not necessarily mean that such release is impossible. To that end, we propose a data release framework that leverages current advances in differentially private query answering to synthesize an anonymized dataset. We show that since each query only affects a sub linear number of tuples, we are able to guarantee differential privacy. To evaluate the efficacy and general applicability of our approach, we experimentally evaluate the utility of our framework in three domains and several real and synthetic datasets. All our results indicate the applicability of our framework to general data release.

Categories and Subject Descriptors

H.2.8 [**DATABASE MANAGEMENT**]: Database Administration-Security, integrity, and protection; K.4.1 [**COMPUTERS AND SOCIETY**]: Privacy

General Terms

Security, Algorithms

Keywords

Differential Privacy, Anonymization, Data Privacy

1. INTRODUCTION

In this paper we consider the problem of differentially private data publishing. In particular, we consider the scenario in which a trusted curator gathers sensitive information from a large number of respondents, creates a relational dataset where each tuple corresponds to one entity, such as an individual, a household, or an organization, and then publishes a privacy-preserving (i.e., *sanitized* or *anonymized*) version of the dataset. This has been referred to as the "non-interactive" mode of private data analysis, as opposed to the "interactive" mode, where the data curator provides an interface through which users may pose queries about the data, and get (possibly noisy) answers.

Publishing microdata is gradually becoming more and more common. The Census bureau has a mandate to publish census data. In addition, medical researchers argue for the need to support clinical research in electronic health records systems. Furthermore, many organizations have the need to publish transactional data for research and other purposes. Microdata publishing, however, has also resulted in well-publicized privacy breach incidents. Examples include the identification of the medical record of the governor of Massachusetts from the GIC data [29]; the identification of the search history of an AOL user from the AOL query log data [3]; the identification of Netflix subscribers from the Netflix Prize dataset [25]; and the de-anonyimzation of social networks [26].

We aim at developing practical techniques that can publish datasets in a way that satisfies the strong privacy guarantee of differential privacy [7, 9]. We observe that there are two main approaches to private data analysis. For lack of better names, we call them the *experimental approach* and the *theoretical approach*. Our approach bridges the techniques developed in both approaches.

The experimental approach has been the dominant approach in the database community. This approach focuses mostly on the non-interactive mode, as this is the primary way data is shared in practice. The emphasis is on developing algorithms and tools that can be applied to large datasets. For privacy, many earlier approaches use syntactic privacy notions such as k-anonymity [29], ℓ-diversity [22], t-closeness [21], and so on. Recently, it has been increasingly recognized that these syntactic privacy notions do not provide sufficient privacy protection, and differential privacy is increasingly being adopted, though so far only to very specialized kinds of datasets, such as to publishing frequencies of web search items [19, 15], or node degree sequences of graphs [16], one-dimensional histograms [17, 31]. For utility, the primary method is experimental evaluation. One experimentally measures to what degree the dataset has been changed and/or to what degree the accuracy of certain data mining tasks is affected.

The theoretical approach has been the dominant approach in the theory/cryptography community. This approach focuses on the interactive mode. The privacy notion developed here is differential privacy [7, 9]. The emphasis has been on

ASIACCS '12, May 2–4, 2012, Seoul, Korea.

proving theorems that identify the boundaries of privacy and utility, and developing methods that can answer particular kinds of queries while satisfying differential privacy. For utility, one chooses some classes of queries and analyzes how accurately these queries can be answered.

We combine the concepts and techniques from both the experimental approach and the theoretical approach. For privacy, we adopt the notion of differential privacy. For utility, we use experimental methods, applying our algorithm on different kinds of datasets and measuring the utility of the resultant datasets. While our goal is to publish microdata in the non-interactive setting, we employ techniques for differentially private query answering developed for the interactive mode. Our approach generates a sequence of queries on the dataset and uses the query results to reconstruct a sanitized version of the dataset. This is possible despite the negative results in [6, 13], which show that answering a linear (in the database size) number of queries relatively accurately leads to disclosure of the dataset in some cases. One key insight resolving the apparent contradiction is that while a linear number of queries is required in our approach, a single tuple in the dataset affects only a sub-linear, $O(\log N)$, number of the queries, making satisfying differential privacy while obtaining reasonably accurate answers possible.

More specifically, we investigate a framework that we call Recursive Partitioning and Summarization (RPS), which exploits the above insight. In the RPS framework, we view tuples as points in a multi-dimensional space. Given a dataset and the multi-dimensional region that tuples in the dataset are in, an RPS algorithm recursively partitions the region into smaller regions, then summarizes each region independently, and finally combines the summarized outputs. To instantiate this framework into a concrete algorithm, one specifies three sub-routines: how to partition a region, how to determine when to stop further partitioning, and how to summarize data items in a region once partitioning stops. An RPS algorithm satisfies differential privacy when each of the three sub-routines satisfy differential privacy, as each node affects only the query regarding the partitions that the node is in, it affects $O(\log N)$ queries, where N is the number of tuples in the dataset. If each sub-routine is performed in time linear in the number of tuples in the region, then the algorithm runs in time $O(N \log N)$.

While the idea of partitioning and summarization has been used implicitly or explicitly before, a key challenge is how to instantiate it. We make a key observation to make the RPS framework feasible for multi-dimensional relational data is that the exponential mechanism for differential privacy [24] provides an effective solution to the critical partitioning step, yielding a close-to-balanced partition in a differentially private way.

1.1 Our Contributions

- We investigate the Recursive Partitioning and Summarization framework and identify using the exponential mechanism to achieve balanced partitioning as a key enabling step for this framework. RPS instantiated with balanced partitioning provides a practical and general method to differentially privately publish multi-dimensional relational datasets. We point out that many data sharing scenarios involve such datasets, including census data, medical records, etc.
- We experimentally evaluate the effectiveness of our approach using both synthetic datasets and the Adult dataset [1]. Our results on the synthetic datasets show that the sanitized datasets preserve useful features of the input datasets, and the performance depends on choices of algorithmic parameters such as maximum depth and stopping conditions. Our result on the Adult dataset show that the sanitized datasets enable accurate classification. In particular, our method outperforms the Mondrian algorithm [20], which satisfies the weaker k-anonymity.
- Finally, to demonstrate the versatility of the RPS approach, we also apply the RPS algorithm to sanitize the node degree sequence of social networks. Such sanitized sequence can then be used to generate a new graph with properties similar to the original graph. This problem is challenging because removing one node may affect the degrees of many other nodes. We show that the RPS approach can be instantiated in a way that satisfies differential privacy relative to adding or removing a node. It provides a practical way for privately releasing node-degree sequences for graph data, outperforming the current state of the art.

The rest of this paper is organized as follows. We discuss background on differential privacy in Section 2. We present the RPS framework in Section 3, and the evaluation results in Sections 4 and 5. We discuss related work in Section 6 and conclude with Section 7.

2. BACKGROUND ON DIFFERENTIAL PRIVACY

We first briefly review the notion of differential privacy, which was developed in a series of papers [6, 11, 4, 9, 7]. The intuition behind this privacy notion is as follows: if a disclosure occurs when an individual participates in the database, then the same disclosure also occurs with similar probability (within a small multiplicative factor) even when the individual does not participate.

Definition 1 (ϵ-Differential Privacy [7, 9]). *A randomized algorithm $\mathcal{A}$ gives ϵ-differential privacy if for any pair of neighboring datasets D and D', and any $S \in Range(\mathcal{A})$,*

$$\Pr[\mathcal{A}(D) = S] \leq e^{\epsilon} \Pr[\mathcal{A}(D') = S] \quad (1)$$

Differential privacy has the following composition property: if two algorithms satisfy differential privacy for ϵ_1 and ϵ_2, then releasing the results of both algorithms satisfy differential privacy for $\epsilon_1 + \epsilon_2$. The parameter ϵ measures the degree of privacy. The larger ϵ is, the lower the privacy. When ϵ is too large, the privacy guarantee becomes meaningless. Hence one needs to have a limit for ϵ, which is know as a *privacy budget*. Answering each query consumes a portion of privacy budget. When the privacy budget is used up after answering a number of queries, no new queries can be answered.

The Feasibility of Non-interactive Differential Privacy There have been a series of negative results concerning differential privacy in the non-interactive mode [6, 13, 9, 10], and these results have been interpreted "to mean that one cannot answer a linear, in the database size, number

of queries with small noise while preserving privacy" and motivates "an interactive approach to private data analysis where the number of queries is limited to be small — sub-linear in the size n of the dataset" [10]. This impossibility result views the database as a vector, where the order of items is important. Furthermore, it depends on answering a linear number of queries in the database size.

These negative results not withstanding, it is shown in several recent papers [5, 10, 12] that it is possible to privately publish a dataset that is capable of answering queries in certain classes relatively accurately. One key insight to account for the this apparent contradiction is the observation that it may still be possible to answer linear number of queries without compromising privacy, provided that each tuple affects answer only to a sub-linear number of them.

Satisfying Differential Privacy There are three methods that can be used to satisfy differential privacy: the common Laplacian Mechanism to add noise, the Smooth Sensitivity method of adding noise. The third alternative is the Exponential Mechanism, which we describe below.

McSherry and Talwar [24] proposed an alternative to the approach of adding noises to the query result. This method is based on the idea that any anonymization method maps, possibly randomly, a set of n inputs each from a domain $\mathcal{D}$ to some output in range $\mathcal{R}$. The mechanism relies on an input query function $q : \mathcal{D}^n \times \mathcal{R} \to \mathbb{R}$ that assigns a real valued score to any pair (d, r) from $\mathcal{D}^n \times \mathcal{R}$. This can be viewed as a *quality* function that assigns higher scores to more desirable outputs.

The goal of the mechanism is to take $d \in \mathcal{D}^n$ and return $r \in \mathcal{R}$ such that $q(d, r)$ is maximized while guaranteeing differential privacy.

DEFINITION 2. *For any quality function $q : (\mathcal{D}^n \times \mathcal{R}) \to \mathbb{R}$, and a privacy parameter ϵ, an exponential mechanism $\mathcal{M}_q^\epsilon : \mathcal{D}^n \to \mathcal{R}$ is a randomized mechanism which, given an input $d \in \mathcal{D}^n$ returns a valid output $r \in \mathcal{R}$ with probability proportional to*

$$\exp\left(\epsilon q(d, r)\right)$$

We define Δq to be the largest possible difference in the quality function when applied to two neighboring datasets. We can thus claim that the exponential privacy mechanism, $\mathcal{M}_q^\epsilon$, with quality function q, gives $(2\epsilon\Delta q)$-differential privacy [24].

3. RECURSIVE PARTITIONING AND SUMMARIZATION

Many privacy-preserving data publishing mechanisms can be modeled as instantiating the following meta-algorithm: Initially, one views tuples as points in a multi-dimensional space. The meta-algorithm first partitions the space into smaller regions, and then returns information about each region, which typically includes the number of tuples in each partition. This can then be used to generate sanitized or synthesized dataset. In our view, there is no fundamental difference between sanitized or synthesized, as both are generated from the original dataset.

To make this process satisfy differential privacy, one needs to make both steps (partitioning and summarization) satisfy DP. The summarization step can be answered easily using a differentially private count query. The basic approach is to add Laplace noise to the count of each region. Multiple improvements exist to increase the accuracy of the answer. This includes wavelet transforms [31] and hierarchical constrained inference [17].

A natural way to make the partitioning process easier to analyze with regard to differential privacy is to perform recursive binary partitioning. That is, each region is first partitioned into two sub-regions, and then these sub-regions are further partitioned. One needs only to ensure that each binary partition decision satisfies differential privacy. There are two general methods of performing this step with acceptable accuracy: fixed partitioning and balanced partitioning.

Fixed Partitioning vs. Balanced Partitioning Fixed partitioning works by deterministically choosing a fixed pivot on which to partition the region. This is independent of the data, and therefore intuitively private and does not consume an differential privacy budget. A prominent example is the universal histogram method in [17]. This approach works by dividing the data domain into equal sized bins. One would then ask for interval counts at different levels of granularity. Conceptually, one can arrange all queried intervals into a tree, where the unit-length intervals are the leaves. Each node in the tree corresponds to an interval, and each node has at least 2 children, corresponding to equally sized subintervals.

Fixed partitioning has two problems. As a result of fixed partitioning, one would get regions of equal size, but conceivably varying densities. Hence, one problem is that high density regions would not get sufficient division. Thus, when partitioning to arbitrary data granularity is impossible, fixed partitioning can provide no accuracy guarantee for all queries.

The problem with deep partitioning, however, is that it would result in many empty regions. Noise will be added to such regions, potentially creating a lot of noisy data. This problem has been observed by several methods in the literature including the Wavelet method [31] for contingency table release. This method works by partitioning to data granularity (i.e. where each region contains identical data points). While this method indeed improves query accuracy by using wavelet transforms on attributes, it suffers from degraded query accuracy when the data is sparse. When each region contains zero, or very few, data points, the relative noise due to summarization is very high; therefore, the quality of the data decreases. While [31] provides a method to remedy the effect of such noise, this method leverages the assumption that adjacent regions have similar counts. This assumption, however, fails to hold as data dimensionality increases and when the nature of the data changes.

One the other hand, balanced, or even-region, partitioning, works by partitioning the region evenly, thus creating equal density partitions. One way to accomplish this is to choose a median datapoint as a pivot. This circumvents the disadvantages of fixed region partitioning at the expense of consuming some privacy budget. We, therefore, examine a differentially private recursive balanced partitioning framework and analyze different methods instantiating it.

3.1 Recursive Partitioning and Summarization Meta-Algorithm

The meta-algorithm for Recursive Partitioning and Summarization (RPS) is given in Algorithm 1. Given a dataset D and the multi-dimensional region R_0 which includes the

tuples in D, the RPS meta-algorithm recursively partitions the region R_0 into smaller regions, and then finally summarizes each region independently and combines the summarized outputs.

Algorithm 1 Recursive Partitioning and Summarization

RPS_sanitize(D, R)
  return post_process(RPS_recursion(D, R))

RPS_recursion(D, R)
  $(Stop, Info) \leftarrow$ stopping_condition(D, R)
  if *Stop* **then**
    return summarize(D, R)
  end if
  $(R_1, R_2) \leftarrow$ balanced_partition(D, R)
  return $(Info,$ RPS_recursion$(D, R_1),$ RPS_recursion$(D, R_2))$

stopping_condition(D, R)
  return (whether to stop partitioning the region R, auxiliary information about R)

summarize(D, R)
  return a multiset of tuples that summarize the tuples in D that are in region R

balanced_partition(D, R)
  return the results of partitioning R into two subregions

post_process(*Tree*)
  return dataset based on the tree resulted from RPS

Running the algorithm on a dataset D and an initial range R_0 results in a series of partitions, which form a partition tree. The root of the tree is R_0. Each internal node has two child nodes which represent two non-overlapping subregions that together form the region represented by the parent node. Running this algorithm also results in a series of queries about D. More specifically, on each internal node corresponding to a region R, two queries are executed: stopping_condition(D, R) and partition(D, R), and on each leaf node corresponding to a region R, two queries are executed: stopping_condition(D, R) and summarize(D, R).

Because changing one tuple in a dataset affects only queries along one path in the tree from a leaf node to the root,an RPS algorithm satisfies ϵ-differential privacy if for any partition tree generated by the algorithm, all queries satisfy differential privacy, and the total privacy budgets consumed along any path is bounded by ϵ. This follows from the sequential and parallel composition properties of differential privacy. [23]

An RPS can be very efficient. If all the three subroutines run in time linear in the number of tuples in the region, then the algorithm takes time $O(N \log N)$.

3.2 Instantiating the RPS framework

We now study how to instantiate this RPS framework into concrete algorithms. One needs to provide the implementation of three functions: stopping_condition(D, R), partition(D, R), and summarize(D, R). We want these functions to satisfy differential privacy relative to D. One further needs to provide a way to allocate a privacy budget to answer these queries.

Partitioning. Partitioning is the most important step. One approach is to first choose an attribute (one dimension in the space), then query the median value among tuples in the region, and finally partition the region at the median point. To answer the medium query, one can use the approach based on smooth sensitivity. This method, however, has a number of drawbacks. First, it is unclear how to best choose the attribute on which to partition. Second, it is unclear how to deal with attributes with categorical values, where the concept of median does not apply.

We propose using the exponential mechanism to partition the dataset. This method can be adapted to both continuous and discrete attributes. For continues attributes, we choose a granularity, and consider all multiples of the granularity. For instance, given a domain range $[0, 100]$ and granularity 1, we consider partitions along $1, 2, \ldots$ to 100. For discrete attributes, we can either consider all possible partitions of the attribute domain if the domain is sufficiently small. Alternatively, we can consider an order (possibly random) over the domain and partition linearly along this order. Such decisions can be made based on the database schema.

Another advantage of using the exponential method is that, with one query we can consider all possible ways of partitioning along different dimensions at once instead of choosing one dimension a priori. For example, suppose that we have only two attributes: education and salary. Here we would consider all possible ways to partition along education and all possible ways to partition along salary. By considering the union of such partitions as the output of the partition mechanism, we can use the exponential mechanism to choose the best partition along both dimensions.

An interesting challenge is what quality function to use. In general, we choose a quality function that would give higher preference to an *even split*. When even split is the only objective, the quality of the partition $(R_1, R_2) \in \mathcal{R}$ is defined as follows:

$$q(d, (r_1, r_2)) = \frac{n - |r_1 - r_2|}{4} \tag{2}$$

where r_1 and r_2 are the number of elements in R_1 and R_2 respectively, and $n = r_1 + r_2$.

Removing one record will change the size of one of the partitions by 1; hence the sensitivity of q is

$$\Delta q \leq \frac{n - |r_1 - r_2|}{4} - \frac{(n-1) - (|r_1 - r_2| + 1)}{4} = \frac{1}{2}$$

We have found that different application domains may benefit from different choices of the quality function. We discuss such choices when discussing different application domains in Sections 4 and 5.

Furthermore, the running time is proportional to the size of the attribute domain, which can be limited to a constant, and is linear to the size of the dataset in the region.

Choosing a stopping condition. To determine whether one should stop further partitioning a region R or not, several methods can be used. The first is to stop when a certain depth is reached. This has two benefits. One is that one can allocate the privacy budget to be used in each level using the knowledge of the maximum depth. The other is that this requires no query on D and consumes no privacy budget; hence one can conserve the privacy budget for other

queries.

Alternatively, we can stop when the region contains too few tuples. To determine whether this is the case, the algorithm would issue a count query to determine how many tuples D has in the region R and decide to stop if the result is lower than a threshold. Because count queries have low sensitivities, the standard method of adding Laplace noise is accurate enough for this purpose.

We can also stop when the size of the region is small enough. For example, if the region already contains a single possible value, i.e., all tuples are already the same, then it makes no sense to perform additional partitioning. Even when some attributes are continuous, perhaps one would consider a region to be small enough so that consuming additional privacy budget in further partitioning is no longer beneficial. Using this stopping condition also consumes no privacy budget.

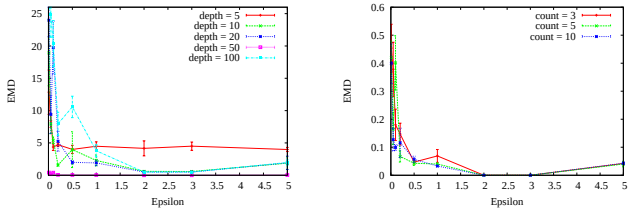
In our implementation, we choose to use a combination of the above methods. A maximum depth limit is used. In addition, the algorithm also stops when the region is *narrow* enough, and when the noisy count returns a number smaller than a threshold (we experimented with 1, 3, 5, 10). In our experiments, we evaluate the choice of the threshold on the utility of the resulting dataset.

Summarizing the resulting partitions. When the stopping condition is reached, we need to summarize the tuples in the region R. To do this, we need to determine how many tuples to generate in the region and then generate these tuples based on the region R.

There are two alternatives to determine how many tuples to generate in the region. The first is to generate based on the depth and the size of the dataset. For example, if the current depth is d, and the dataset size is N, we generate $N/2^d$ tuples in R. This approach consumes no privacy budget. The other approach is to issue another count query to get the number c of tuples that are in R, which can be answered using all remaining privacy budget, as this is the last query along this path.

To generate a tuple in R, a natural approach is to generate each attribute independently from other attributes. For an ordered attribute, several alternatives exist. One could sample a random value in the region, choose the mid point (mean) between the two boundary values, and in some cases (such as when dealing with node degree sequences in Section 5) choosing one end of the region turns out to be the best approach. For a categorical attribute, we simple use the set of all values in the region. Alternatively, for each attribute, one could randomly choose a value in the range.

Allocating the privacy budget. Many strategies can be applied in allocating the privacy budget among different queries, depending on whether partitioning, checking stopping condition, and summarizing need to consume the privacy budget, and the relative importance of different subroutines for an application. In our experiments, we choose to leave half the privacy budget for the final query needed to summarize a leaf region, in order to get a reasonably accurate count, and the other half for partitioning and checking stopping condition. Other alternatives are possible. Another intriguing idea is how to split the privacy budget for each level between partitioning and checking stopping condition. It may be beneficial that on shallower levels, one does not check for stopping conditions and use all budget for partitioning.



(a) Depth (b) Noisy Stopping Count

Figure 1: Utility of a single attribute under different instantiations of RPS that differ on ϵ, depth, and stopping count. The graphs plot ϵ vs Earth Mover's Distance to the original distribution. Smaller values are better.

The relative effectiveness of different instantiations of RPS algorithms likely depends on the input dataset and the privacy budget at hand. In a data publishing scenario, the data curator may want to experiment with different settings and choose the most desirable output to publish.

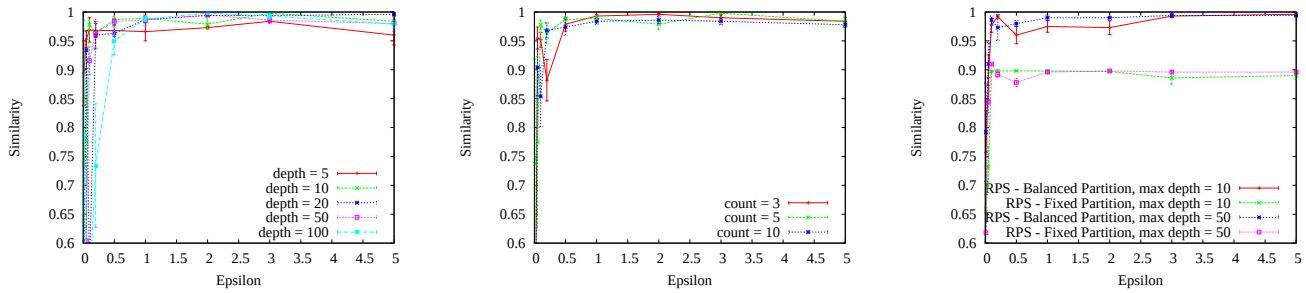
4. ANONYMIZING MULTIDIMENSIONAL DATA

We evaluate RPS on multidimensional data. A multidimensional dataset, D, is a set of records. Each record consists of a list of values corresponding to a set of attributes, or dimensions. The applications of this for such types of data are wide and varied and include Census data, medical records, and many others. In this section, we first show the general applicability of our approach by anonymizing a synthetic dataset under different instantiations of the RPS algorithm. We then apply the framework to a general census dataset: the UCI Machine Learning Adult dataset [1], which has been used in several studies on privacy preserving data publishing. We study the efficacy of our approach by examining characteristics of the anonymized dataset as compared to the original and by measuring accuracy of data mining workloads. We repeat each experiment 5 times and report the mean with the standard error.

4.1 Results on Synthetic Data

To synthesize data, we sampled points from a predefined space. We first set to evaluate the ability of the algorithm to preserve the distribution of each attribute (dimension) in the data. Hence, we sampled a set of 10,000 points in $\mathbb{R}$ from a normal distribution with $\mu = 50$ and $\sigma = 25$. This is similar to the experimental approach used to evaluate Mondrian [20]. To evaluate how well the distribution is preserved, we measured the Earth Mover's Distance between the original dataset and the anonymized counterpart.

The Earth Mover's Distance (EMD) is also referred to as the 1st Mallows distance or 1st Wasserstein distance. Given two empirical probability distributions, the EMD measures the minimum cost of turning one distribution into another. Informally, we can view one distribution as a mass of dirt over the space, and the other as a collection of holes in the same space. EMD measures the least amount of work needed to fill the holes with dirt; where work refers to the amount of dirt times the distance by which it is moved. EMD can be calculated by solving the transportation problem. More formally, for two sorted numerical datasets X and Y in $\mathbb{R}$,



(a) Depth (b) Noisy Stopping Condition (c) Partition Method

Figure 2: Similarity of tuples within a region under different instantiations of RPS that differ on ϵ, depth, stopping count, and partition method. Larger values are better

the EMD is calculated as follows:

$$\mathsf{EMD}(X,Y) = \sqrt{\frac{1}{n}\sum_i (X_i - Y_i)^2}$$

where X_i maps to Y_i such that the overall amount of work is minimized. The EMD was used by Li et al. [21] in evaluating the utility of anonymized datasets.

The results for this evaluation are shown in Figure 1. We evaluate the efficacy of RPS by changing different parameters: the stopping condition including maximum allowable partition depth and a noisy stopping condition, and the privacy parameter (ϵ). In Figure 1(a), we fix the stopping condition to a noisy count ≤ 5, and we vary the maximum allowable depth from 5 to 100. We test this under values of epsilon ranging from 0.01 to 5. In Figure 1(b), we fix the maximum depth to 50 and vary the stopping count.

We can conclude the following from the results. First, the choice of depth matters. A depth which is too shallow (ex: 5) gives bad results. This is because the algorithm terminates while the partitions are coarse grained. Similarly, a depth which is too large (ex: 100) also gives bad results under small ϵ. This is because the privacy budget allocated to individual queries would be very small. For the dataset at hand, a depth of 50 gives the best results achieving an EMD close to 0. A depth of 10 or 20 gives slightly worst results. Similar reasoning can be applied to the stopping condition, although the differences in the results are less noticeable. A small noisy count (3) and a large noisy count (10) give slightly worst results for smaller values of ϵ. Finally, an interesting thing to note here is that changes in parameters are most noticeable under small values of ϵ. As ϵ gets larger, the accuracy stabilizes under the different changes and the variance in the results decreases.

The next thing we evaluate is the ability of the RPS algorithm to respect clustering inherent in a multidimensional dataset. The reasoning behind this evaluation is as follows: RPS generates a set of regions, where each region would contain a set of tuples. Ideally, we want tuples in the same cluster to be in the same region. Hence, we created a 5-dimensional dataset in which tuples are generated by three different processes (i.e. there are three clusters of tuples). Each tuple was sampled from one of three different multivariate Gaussian distributions. These distributions were chosen to have different means and different covariance matrices. We then applied different instantiations of the RPS framework and assessed the similarity of the tuples in each region. We labelled each region with the generator that created the majority of the tuples in that region. We then calculated the percentage such tuples hold over all the tuples in the region and aggregated the results over all regions.

We present the results in Figure 2. Again, we instantiated the RPS framework with different parameters as we did above. Figure 2(a) varies the maximum depth of the partitions while fixing the noisy stopping count to 5. It again shows that too small or too large of a maximum depth hinders utility. Figure 2(b) fixes maximum depth at 10 and varies the noisy stopping count. As the value of ϵ increases, the difference in this condition is negligible. Having a larger count gives slightly worst utility. A smaller noisy count also gives worst utility at smaller values of ϵ. In general, setting the noisy count to 5 works well. Figure 2(c) fixes the noisy count at 5 and varies the partition method under two different settings of maximum depth. We compare a method of partitioning that uses the median (via the exponential method) and a method that just takes the midpoint of the boundary interval as the splitting value. The results favor the partition method that uses the median. This is because it more accurately reflects how data is clustered, while taking the mean introduces some errors into how the final regions are defined.

4.2 Results on the Adult Dataset

We also evaluate the effectiveness of our algorithm on the UCI Machine Learning Adult dataset [1]. The Adult dataset is a general census dataset that is commonly used to predict the income of an individual (either above 50K or below 50K) by taking into account several features such as level of education, sector of industry, country, age, etc. The dataset has 30,162 records that do not contain missing values. We consider 11 attributes for each record: age, workclass, education, marital-status, occupation, relationship, race, sex, hours-per-week, native-country, and salary (<=50K, >50K).

We evaluate the dataset by applying data mining classification algorithms: the Naive Bayes algorithm and the C4.5 decision tree algorithm. This approach has several advantages. First, it allows us to evaluate a dataset for the purpose it is intended to serve. Hence, we can evaluate utility objectively. Second, it allows us to evaluate the efficacy of our algorithm in maintaining the relationships between attributes. Classification algorithms take into account the relationships between attributes in building the classification model. The more these relationships are intact, the

better the results compared to the original data. To evaluate the accuracy of our results, we performed ($k = 5$)-fold cross validation. This is a standard technique in data mining evaluation which partitions the dataset into k folds, then repeats the algorithm k times, each time keeping one fold for testing and using the others as the learning dataset.

To partition the dataset in RPS, we consider integer granularity for continuous attributes. For each discrete attribute domain, we consider an order over the values in the domain. We use orders that are already available when applicable (ex: for education) and generate a random order otherwise (ex: for native country). We furthermore tune the algorithm to the specific utility guarantees needed; i.e. the classification task with a binary class attribute. In this case, we want the partitions over the median to separate the different values of the class attributes in the partitions. We modify the quality function to support that. Suppose that the dataset being anonymized has a class attribute that can take two values: $+$ and $-$. Further suppose that in a partition $(R_1, R_2) \in \mathcal{R}$, the number of $+$ and $-$ in R_1 are x_1 and x_2 and that the numbers in R_2 are y_1 and y_2. We can specify the quality function as:

$$q(d, (r_1, r_2)) = \frac{n - |r_1 - r_2| + \max(x_1 + y_2, y_1 + x_2)}{2} \quad (3)$$

When summarizing the resulting partitions, we can choose a class attribute for each partitions by taking a noisy majority vote.

The results of the anonymization are in Figure 3. In these experiments, the attribute on which to partition is chosen randomly at each step. We vary ϵ and measure the classification accuracy under two stopping conditions: a noisy count of 5 and continuing to a maximum depth of 50 (i.e. stop at a count of 0). To evaluate our results we use three benchmarks. The first is the accuracy of the classification algorithm on the original data. We want to achieve an accuracy closest to this. The second is the baseline accuracy for the dataset. This is the accuracy if we take a simple majority vote for the class label. For the Adult dataset, this is 0.75. Finally, we compare the result with the original Mondrian algorithm.

In Figure 3(a) we can see that the RPS algorithm is able to achieve an accuracy close to that of the Naive Bayes algorithm. Since this algorithm relies on conditional probabilities given the class labels, we can assert that our algorithm largely maintains the correlation between attributes and the class label. Furthermore, we can see that RPS at a noisy count of 5 performs better than continually recursing to the maximum depth. This is primarily because the latter might result in empty or sparse regions which would add noise to the summarization. In addition, our algorithm surprisingly outperformed the Mondrian algorithm for k-anonymity. Our intuition is that this is because of the way Mondrian performs the partitioning. Mondrian gets the actual median without considering the class label. Our algorithms favor an even split while keeping the class label in mind.

In Figure 3(b), we run the C4.5 classifier on the datasets. For this classifier, our algorithm performed slightly worse compared to the previous result. This may be because the C4.5 classifier is more sensitive to changes in attribute distributions than the pervious algorithm.

5. ANONYMIZING GRAPH PROPERTIES

We extend our approach to show how to efficiently anonymize other types of data. The problem of anonymizing graph data has been motivated by Backstorm, Dwork et al. [2, 8]. In this section, we focus on anonymizing properties of the graph that can be used to regenerate an anonymized version of the graph itself. In particular, we look at the problem of anonymizing the degree sequence of the graph while protecting individual nodes in the graph.

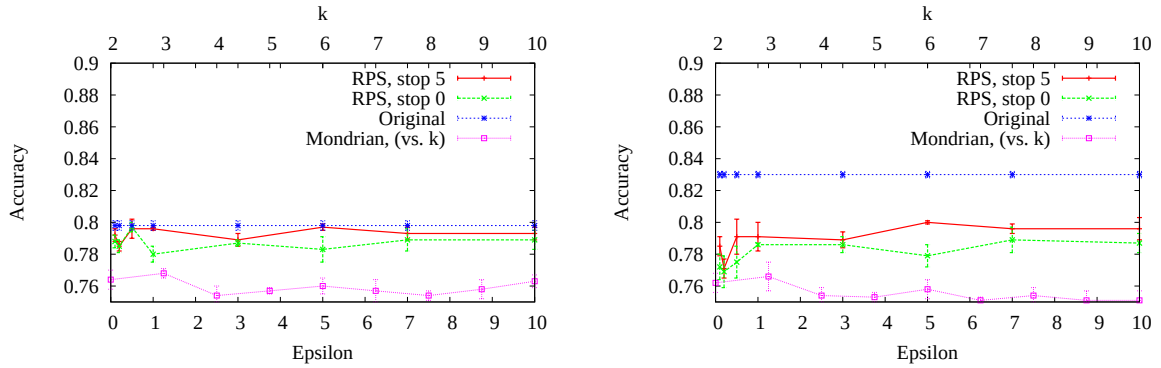
The problem of releasing the degree sequence has been analyzed by Hay et al. [16]. The main contribution of Hay et al.'s method relies on viewing the degree sequence query as a series of queries returning the ith largest degree in the graph. Since the query returns the degree distribution in sorted order, the sensitivity of the query as a whole due to the removal of one edge is 2. Analogously, the sensitivity due to removing up to Λ edges is 2Λ. Hence, noise can be added to each degree using the Laplacian method proportional to the sensitivity. In addition, Hay et al. leverage the fact that the unanonymized degrees are sorted in order to improve on the accuracy of the anonymized result using a constrained inference algorithm. This is currently the only known method of releasing the degree sequence while satisfying differential privacy.

This method, however, has focused on protecting the individual edges in the graph rather than nodes themselves since the latter requires the addition of an unacceptably large amount of noise using the Laplacian mechanism. Edge differential privacy, however, is insufficient. Ideally, one needs to prevent the re-identification of nodes, rather than whether there is a relationship between two nodes. Here, we show how to answer the degree sequence query efficiently and accurately while satisfying *node*-differential privacy. We experimentally compare our approach to this current state of the art.

5.1 Anonymizing the Degree Sequence

More formally, the anonymization problem we consider can be defined as follows. Given a graph G^n_Λ with n nodes such that the maximum degree of any node is Λ, we release a sorted sequence of size n which corresponds to the degree sequence of the graph. Generally, for a graph of size n, $\Lambda = n - 1$. However, in some publishing scenarios is it reasonable to assume that the maximum degree of the graph is known. For instance, the data publisher may a priori sample the graph or prune nodes that exceed a particular degree. Alternatively, one can obtain a differentially private estimate of the maximum degree and use that in the anonymization. In essence, if $\Lambda = n - 1$ is much larger than the maximum degree of the graph, then the anonymized degree distribution is likely to be biased at the tail since the highest degree nodes will be over generalized. Since this is generally the case, we might want to exploit different methods of summarizing the resulting graph partitions. Furthermore, the choice of Λ will have the greatest effect on only one partition in the sequence; this might result in an overestimation of the degrees in that partition. We can slightly account for this bias by deterministically choosing the lower boundary of the domain for each region. Our experimental evaluation that follows, show that graph datasets can benefit from this type of summarization.

Partitioning Method To partition, we have to adjust



(a) Naïve Bayes Classifier (b) C4.5 Classifier

Figure 3: Classification Accuracy for the Adult Dataset - Accuracy is expressed as a percentage. The larger the value the better

the quality function to account for the increased sensitivity of graph queries. We consider the case where we need to protect individual nodes in the graph. Removing one node will change up to $\Lambda + 1$ entries in the degree sequence: the degree of the node itself will be set to 0, and the degrees of all its neighbors will decrease by 1. Thus far, this has been a hurdle preventing development of accurate dissemination algorithms for graph queries. The higher sensitivity of the degree sequence query still manifests itself in the RPS algorithm; however, the specific mechanism by which the algorithm partitions the sequence and summarizes the results helps minimize the effect of the increased sensitivity. That is, since we do not add noise to the individual degrees, the negative effects of the increased sensitivity would not be substantial. We would only need to sacrifice some accuracy in choosing the partitions.

We account for this by changing the quality function that determines an even split to decrease its sensitivity. We therefore define the quality of the partition $(R_1, R_2) \in \mathcal{R}$ in as follows:

$$q(d, (r_1, r_2)) = \frac{r_1 + r_2 - |r_1 - r_2|}{4n + 2} \tag{4}$$

where r_1 and r_2 are the sizes of R_1 and R_2 respectively and n is the total number of nodes in the graph.

Removing one node will change the degrees of up to $\Lambda + 1$ nodes in the sequence. This can cause up to $\Lambda + 1$ nodes to shift in the partitions (i.e. move from R_1 to R_2, or vice-versa). Hence Δq for the degree sequence query is:

$$\Delta q \leq \frac{r_1 + r_2 - |r_1 - r_2|}{4n + 2} - \frac{r_1 + r_2 - 1 - (|r_1 - r_2| - 2(\Lambda + 1))}{4n + 2}$$
$$= \frac{2(\Lambda + 1) + 1}{4n + 2} \leq \frac{2n + 1}{4n + 2} = \frac{1}{2}$$

Stopping condition We also need to consider the sensitivity of the degree sequence query when taking the counts in each region. Note that the sensitivity of the count query here is no longer 1, therefore we cannot use the normal count query. We therefore allow the algorithm to continue until a maximum depth is reached. We use a depth of 15 in our experiments.

Summarizing the resulting regions When summarizing each region at the end, we calculate the smooth sensitivity of the count query. The reasoning behind this approach is that removing one node will most likely only shift some of the node degrees in a region by 1 and remove, at most, one node completely. Hence, in addition to the removed node, only the nodes with degrees at the region boundaries will affect the count. By calculating the smooth sensitivity based on these counts (and adding noise sampled from a fat tailed distribution, ex: Cauchy), we can release accurate counts for each region while satisfying differential privacy. Finally, we deterministically choose the lower boundary of the domain for each region to represent the degree of nodes in the region.

	nodes	edges	α	$\mathsf{max_d}$
Caltech	769	16656	1.14	248
Georgetown	9414	425638	1.04	1235
UNC	18163	766800	1.05	3795
Enron (Undirected)	36692	183831	1.56	1383

Table 1: Graph Statistics - α is the power law coefficient of the degree distribution, and $\mathsf{max_d}$ is the maximum degree of nodes in the graph

5.2 Experimental Evaluation

The previous anonymization scheme was implemented and tested on 4 graph datasets with varying sizes and properties. The properties for the datasets are summarized in Table 1. The datasets Caltech, Georgetown and UNC are Facebook college networks captured in 2005 [30]. These datasets show the full intra-school links as they were in September 2005. Each node represents an individual and each edge represents a friendship link. Each of these datasets contains a different number of nodes and a different density of edges; and hence show the efficacy of our approach on graph with a few hundred nodes to graphs of tens of thousands of nodes. The last dataset we use is the Enron email network [18]. Each node in the dataset represents an individual, and each undirected edge indicates that at least one email was exchange between the two parties. This graph is sparser compared to the Facebook networks.

Utility Measures To evaluate our results, we employ two statistical distance measures commonly used in the literature for evaluating the similarity of two distributions [16,

21]. The first is the Kolmogorov-Smirnoff statistic, also known as the KS-distance. Given two empirical distribution functions F_X and F_Y, where $F_X(x) = \frac{1}{n}\sum_{i=1}^{n} \mathbf{I}_{X_i \leq x}$, the KS-Distance between two such distributions is defined as the maximum distance between the two distributions at any point: $\mathsf{KS}(X, Y) = \max_x |F_X(x) - F_Y(x)|$. It is sensitive to differences in both location and shape of the empirical cumulative distribution functions of the two samples. Smaller values of KS-distance indicate closer distributions and better utility. The KS-distance is widely used as as a general non-parametric distance measure. It was also employed by Hay et al. to evaluate the distance between a degree sequence and its anonymized counterpart [16]. The KS-distance, however, is less sensitive to the tail of the distributions. We thus use another distance measure to account for such discrepancies.

The second measure we use is the Earth Mover's Distance (EMD), as described in the previous section. Hay et al. employed the Mallow's distance at p=2 (equivalent to the EMD) for their evaluation of similarity of degree sequences [16].

Results Figure 4 and Figure 5 (in Appendix A) show the results for the KS-distance and the Earth Mover's Distance respectively. Each anonymization was repeated 5 times for each value of ϵ ranging from 0.01 to 10.0. The graphs show the results along with the standard error. We also repeat the experiments under different assumptions of maximum degree. In Figure 4(a) and Figure 5(a), we assume a priori knowledge of the maximum degree and hence set Λ to be the actual maximum degree shown in Table 1. In Figure 4(b) and Figure 5(b), we set Λ to its maximum possible value given a graph of size n ($\Lambda = n - 1$). In this case, we summarize the partitions by choosing the lower bound for the partition domain.

We compare the results of our anonymization method to the work of Hay et al. [16], which presents the only other method of satisfying differential privacy for graph degree sequence queries. The experimental results look very promising. While considering the KS-distance (Figure 4), the RPS algorithm can achieve fairly high accuracy. Furthermore, the KS-Distance for the both settings of Λ was small. This is even more the case with larger graphs. We note, however, that this is because larger values of Λ only affect higher degree nodes and hence only the tail of the distribution to which the KS-distance is less sensitive. The results imply that accurate degrees are reported for *most* of the nodes in the graph and this is sufficient in several analysis scenarios. In addition, the RPS algorithm performed much better than the Laplacian and constrained inference alternative. This remained to be the case for all graphs under all values of epsilon. The deviation is most accentuated for smaller values of epsilon, which imply a large amount of noise added via the Laplacian mechanism.

We also analyzed the accuracy of the anonymized degrees under the Earth Mover's distance (Figure 5). This measure allocates equal sensitivity to all nodes in the graph and is thus as sensitive at the tail as it is at the median. The results are also very good for both settings of Λ. As expected, $\Lambda = \mathsf{max_d}$ results in better accuracy; however the deviation between both settings of Λ is not large. Furthermore, the EMD for both is considered good. For $\epsilon \geq 1$, the result is close to 10 and is at least 100 units smaller than the result for the constrained inference method. We attribute this to the benefits of the flexibility RPS gives in choosing how to summarize the partitions. The results for all graphs consistently outperformed the constrained inference alternative.

6. RELATED WORK

Work on privacy is largely motivated by a number of privacy breach incidents [29, 3, 25, 26]. Most work done, however, has focused on syntactic methods of achieving privacy, which are deemed insufficient for privacy protection [29, 22, 21]. In this paper we utilize a stronger privacy guarantee: differential privacy. Differential privacy was presented in a series of papers [6, 11, 4, 9, 7] and methods of satisfying it are presented in [7, 27, 24].

The Mondrian algorithm [20] performs a similar partitioning and summarization of the dataset, however it aims to satisfy the weaker privacy notion of k-anonymity. In addition, Blum et al. [5] proposed an approach that employs non-recursive partitioning, but their results are mostly theoretical and lack general practical applicability. [14] gives an approach for data mining for differential privacy that uses the exponential mechanism; however, their approach is not concerned with publishing an entire dataset. Privacy Integrated Queries (PINQ) [23] is a platform which provides methods, including count and median, to implement the RPS algorithm. PINQ, however, only provides mechanisms for differentially private database access, rather than actual data sanitization methods.

Work has also been done on improving the accuracy of interactive data release [28]. This classifies queries as "easy" or "hard", according to whether or not the majority of databases consistent with previous answers to hard queries would give an accurate answer to it. A "hard" query is answered using the Laplacian mechanism. An "easy" query simply returns the corresponding median value. Our approach, however, deals with the non-interactive case.

Anonymizing graph properties has been motivated by [2, 8]. In addition, [16] present a method to release the degree sequence while protecting edges. We show how our framework can be used to release the degree sequence while protecting nodes.

7. CONCLUSION

In this paper we tackle the problem of differentially private data release. We first consider the non-interactive mode of differentially private data release. We examine current negative results and comment on how they are inapplicable to general differentially private data publication.

We then propose a general and practical anonymization framework called Recursive Partitioning and Summarization (RPS). RPS works by issuing a set of differentially private queries to recursively divide the dataset into several regions. We then generate a synthetic dataset by summarizing each region in a differentially private manner. We experimentally evaluate the utility of our framework in three domains. First, we benchmark our method using synthetically generated datasets and show that RPS can indeed preserve the characteristics of such datasets. Next, we apply RPS to the Adult census dataset. We run common data mining algorithms and evaluate the effect of anonymization. Finally, we show how our framework can be applied to graph datasets by anonymizing the degree sequences of four real-world graph datasets with different properties. All our results indicate the applicability of our framework to general data release.

Acknowledgements

This paper is based upon work supported by the United States National Science Foundation under Grant No. 1116991, and by the United States AFOSR under grant titled "A Framework for Managing the Assured Information Sharing Lifecycle".

8. REFERENCES

[1] A. Asuncion and D. Newman. UCI machine learning repository, 2010.

[2] L. Backstrom, C. Dwork, and J. Kleinberg. Wherefore art thou r3579x?: anonymized social networks, hidden patterns, and structural steganography. In *WWW*, pages 181–190, 2007.

[3] M. Barbaro and J. Tom Zeller. A face is exposed for aol searcher no. 4417749. New York Times, Aug 2006.

[4] A. Blum, C. Dwork, F. McSherry, and K. Nissim. Practical privacy: the sulq framework. In *PODS '05: Proceedings of the twenty-fourth ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems*, pages 128–138, New York, NY, USA, 2005. ACM.

[5] A. Blum, K. Ligett, and A. Roth. A learning theory approach to non-interactive database privacy. In *STOC*, pages 609–618, 2008.

[6] I. Dinur and K. Nissim. Revealing information while preserving privacy. In *PODS*, pages 202–210, 2003.

[7] C. Dwork. Differential privacy. In *ICALP*, pages 1–12, 2006.

[8] C. Dwork. Differential privacy: A survey of results. In *TAMC*, pages 1–19, 2008.

[9] C. Dwork, F. McSherry, K. Nissim, and A. Smith. Calibrating noise to sensitivity in private data analysis. In *TCC*, pages 265–284, 2006.

[10] C. Dwork, M. Naor, O. Reingold, G. Rothblum, and S. Vadhan. On the complexity of differentially private data release: efficient algorithms and hardness results. *Proceedings of the 41st annual ACM symposium on Theory of computing*, pages 381–390, 2009.

[11] C. Dwork and K. Nissim. Privacy-preserving datamining on vertically partitioned databases. In *CRYPTO*, pages 528–544, 2004.

[12] C. Dwork, G. Rothblum, and S. Vadhan. Boosting and differential privacy. *Foundations of Computer Science (FOCS), 2010 51st Annual IEEE Symposium on*, pages 51 – 60, 2010.

[13] C. Dwork and S. Yekhanin. New efficient attacks on statistical disclosure control mechanisms. *Advances in Cryptology–CRYPTO 2008*, pages 469–480, 2008.

[14] A. Friedman and A. Schuster. Data mining with differential privacy. In *KDD '10: Proceedings of the 16th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 493–502, New York, NY, USA, 2010. ACM.

[15] M. Götz, A. Machanavajjhala, G. Wang, X. Xiao, and J. Gehrke. Privacy in search logs. *CoRR*, abs/0904.0682, 2009.

[16] M. Hay, C. Li, G. Miklau, and D. Jensen. Accurate Estimation of the Degree Distribution of Private Networks. In *ICDM*, pages 169–178, 2009.

[17] M. Hay, V. Rastogi, G. Miklau, and D. Suciu. Boosting the accuracy of differentially private histograms through consistency. *Proc. VLDB Endow.*, 3:1021–1032, September 2010.

[18] B. Klimt and Y. Yang. Introducing the enron corpus. In *CEAS*, 2004.

[19] A. Korolova, K. Kenthapadi, N. Mishra, and A. Ntoulas. Releasing search queries and clicks privately. In *WWW*, pages 171–180, 2009.

[20] K. LeFevre, D. DeWitt, and R. Ramakrishnan. Mondrian multidimensional k-anonymity. In *ICDE*, page 25, 2006.

[21] N. Li, T. Li, and S. Venkatasubramanian. t-closeness: Privacy beyond k-anonymity and l-diversity. In *ICDE*, pages 106–115, 2007.

[22] A. Machanavajjhala, J. Gehrke, D. Kifer, and M. Venkitasubramaniam. ℓ-diversity: Privacy beyond k-anonymity. In *ICDE*, page 24, 2006.

[23] F. McSherry. Privacy integrated queries: an extensible platform for privacy-preserving data analysis. In *SIGMOD*, pages 19–30, 2009.

[24] F. McSherry and K. Talwar. Mechanism design via differential privacy. In *FOCS*, pages 94–103, 2007.

[25] A. Narayanan and V. Shmatikov. Robust de-anonymization of large sparse datasets. In *S&P*, pages 111–125, 2008.

[26] A. Narayanan and V. Shmatikov. De-anonymizing social networks. In *IEEE Symposium on Security and Privacy*, pages 173–187. IEEE Computer Society, 2009.

[27] K. Nissim, S. Raskhodnikova, and A. Smith. Smooth sensitivity and sampling in private data analysis. In *STOC*, pages 75–84, 2007.

[28] A. Roth and T. Roughgarden. Interactive privacy via the median mechanism. In *Proceedings of the 42nd ACM symposium on Theory of computing*, STOC '10, pages 765–774, New York, NY, USA, 2010. ACM.

[29] L. Sweeney. k-anonymity: A model for protecting privacy. *Int. J. Uncertain. Fuzziness Knowl.-Based Syst.*, 10(5):557–570, 2002.

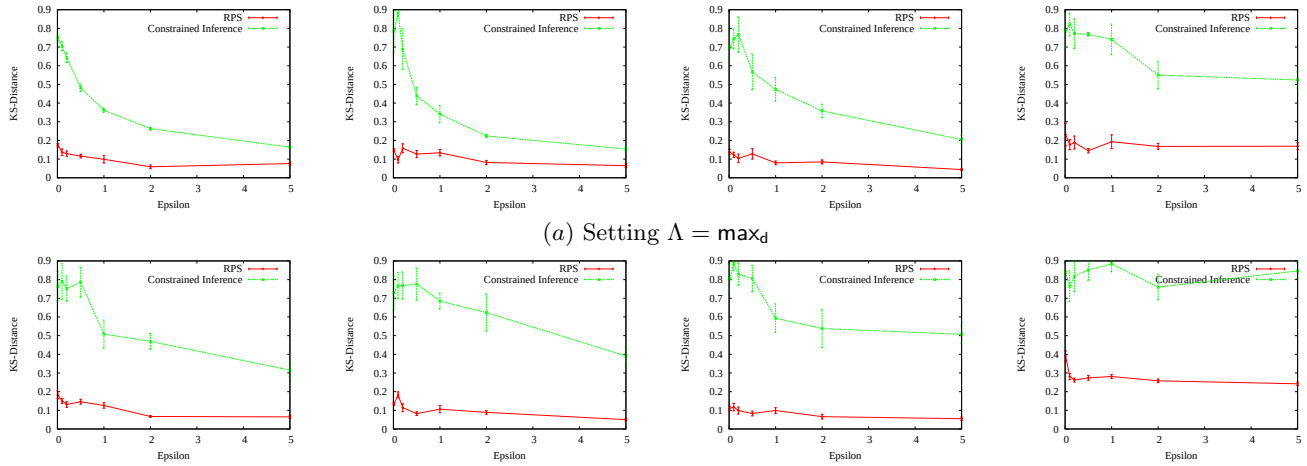
[30] A. L. Traud, E. D. Kelsic, P. J. Mucha, and M. A. Porter. Community structure in online collegiate social networks. arXiv:0809.0960, 2008.

[31] X. Xiao, G. Wang, and J. Gehrke. Differential privacy via wavelet transforms. *IEEE Transactions on Knowledge and Data Engineering*, 23:1200–1214, 2011.

APPENDIX

A. UTILITY OF DEGREE SEQUENCE ANONYMIZATION

To demonstrate the utility of anonymizing the degree sequence query, we provide the results for the experiments described in Section 5 in Figures 4 and 5. Figure 4 uses the KS-Distance of the anonymized sequences to the original.

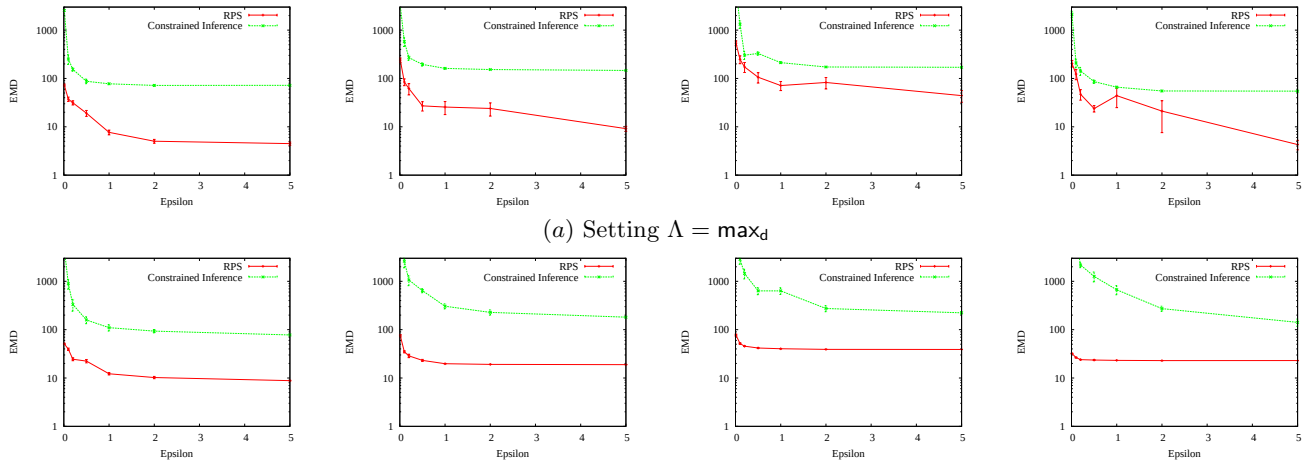


(a) Setting $\Lambda = \mathsf{max_d}$

(b) Setting $\Lambda = n - 1$

Caltech Georgetown UNC Enron

Figure 4: Degree Distribution Accuracy under Node Differential Privacy - the plots show ϵ vs. accuracy (KS-Distance). The smaller the values, the better.



(a) Setting $\Lambda = \mathsf{max_d}$

(b) Setting $\Lambda = n - 1$

Caltech Georgetown UNC Enron

Figure 5: Degree Distribution Accuracy under Node Differential Privacy - the plots show ϵ vs. accuracy (Earth Mover's Distance). The smaller the values, the better. The y-axis is given in log scale