Publishing Set-Valued Data via Differential Privacy

Rui Chen
Concordia University
Montreal, Canada
ru_che@encs.concordia.ca

Noman Mohammed
Concordia University
Montreal, Canada
no_moham@encs.concordia.ca

Benjamin C. M. Fung
Concordia University
Montreal, Canada
fung@ciise.concordia.ca

Bipin C. Desai
Concordia University
Montreal, Canada
bcdesai@cs.concordia.ca

Li Xiong
Emory University
Atlanta, USA
lxiong@mathcs.emory.edu

ABSTRACT

Set-valued data provides enormous opportunities for various data mining tasks. In this paper, we study the problem of publishing set-valued data for data mining tasks under the rigorous differential privacy model. All existing data publishing methods for set-valued data are based on partition-based privacy models, for example k-anonymity, which are vulnerable to privacy attacks based on background knowledge. In contrast, differential privacy provides strong privacy guarantees independent of an adversary's background knowledge, computational power or subsequent behavior. Existing data publishing approaches for differential privacy, however, are not adequate in terms of both utility and scalability in the context of set-valued data due to its high dimensionality.

We demonstrate that set-valued data could be efficiently released under differential privacy with guaranteed utility with the help of context-free taxonomy trees. We propose a probabilistic top-down partitioning algorithm to generate a differentially private release, which scales linearly with the input data size. We also discuss the applicability of our idea to the context of relational data. We prove that our result is (ϵ, δ)-useful for the class of counting queries, the foundation of many data mining tasks. We show that our approach maintains high utility for counting queries and frequent itemset mining and scales to large datasets through extensive experiments on real-life set-valued datasets.

1. INTRODUCTION

Set-valued data, such as transaction data, web search queries, and click streams, refers to the data in which each record owner is associated with a set of items drawn from a universe of items [19, 29, 30]. Sharing set-valued data provides enormous opportunities for various data mining tasks in different application domains such as marketing, advertising, and infrastructure management. However, such data often contains sensitive information that could violate individual privacy. Therefore, set-valued data needs to be sanitized before it can be released to the public. In this paper, we consider the problem of publishing set-valued data that simultaneously protects individual privacy under the framework of *differential privacy* [8] and provides guaranteed utility to data miners.

There has been some existing research [5, 16, 19, 29, 30, 36, 37] on publishing set-valued data based on *partition-based privacy models* [15], for example k-anonymity [28] (or its relaxation, k^m-anonymity [29, 30]) and/or confidence bounding [5, 32]. However, due to both their vulnerability to adversaries' background knowledge and their deterministic nature, many types of privacy attacks [20, 26, 33] have been identified on these approaches derived using these models, leading to privacy compromise. In contrast, *differential privacy* [8], a relatively new privacy model stemming from the field of statistical disclosure control, provides strong privacy guarantees independent of an adversary's background knowledge, computational power or subsequent behavior. Differential privacy, in general, requires that the outcome of any analysis should not overly depend on a single data record. It follows that even if a user had opted in the database, there would not be a significant change in any computation based on the database. Therefore, this assures every record owner that any privacy breach will not be a result of participating in a database.

There are two natural settings of data sanitization under differential privacy: *interactive* and *non-interactive*. In the interactive setting, a sanitization mechanism sits between the users and the database. Queries posed by the users and/or their responses must be evaluated and may be modified by the mechanism in order to protect privacy; in the non-interactive setting, a data publisher computes and releases a sanitized version of a database, possibly a synthetic database, to the public for future analysis. There have been some lower bound results [6, 8, 9] of differential privacy, indicating that only a limited number of queries could be answered; otherwise, an adversary would be able to precisely reconstruct almost the entire original database, resulting in a serious compromise of privacy. Consequently, most recent works have concentrated on designing various interactive mechanisms that answer only a sublinear number, in the size n of the underlying database, of queries *in total*, regardless of the number of users. Once this limit is reached, either the database has to be shut down, or any further query

Proceedings of the VLDB Endowment, Vol. 4, No. 11

would be rejected. This limitation has greatly hindered their applicability, especially in the scenario where a database is made available to many users who legitimately need to pose a large number of queries. Naturally, one would favor a non-interactive release, which could be used to answer an arbitrary large number of queries or for various data analysis tasks. Blum et al. [4] point out that the aforementioned lower bounds could be circumvented in the non-interactive setting at the cost of preserving usefulness for *only* restricted classes of queries. However, they did not provide an efficient algorithm.

Dwork et al. [10] further propose a more efficient non-interactive sanitization mechanism with a *synthetic* dataset output. However, the progress is not sufficient to solve the problem of publishing set-valued data for *data mining tasks* for two reasons. First, the approach in [10] is of runtime complexity, $poly(|\mathcal{C}|, |I|)$, where $|\mathcal{C}|$ is the size of a concept class and $|I|$ is the size of the item universe. A set-valued dataset could be reconstructed by *counting queries* (see Section 3.3 for a formal definition). This implies a complexity of $poly(2^{|I|} - 1, |I|)$, which is not desirable for real-life set-valued data, where $|I|$ is typically over a thousand. Second, for data mining tasks the published data needs to be "semantically interpretable"; therefore, synthetic data does not fully meet the publisher's goal [37]. Similarly, the approaches of two very recent papers [34, 35], which are designed for publishing *relational data* by first enumerating all possible combinations of all different values of different attributes, also suffer from the scalability problem in the context of set-valued data. We argue that a more efficient solution could be achieved by taking into consideration the underlying dataset. The solution also has a positive impact on the resulting utility as there is no need to add noise to every possible combination. The main technical challenge is how to make use of a specific dataset while satisfying differential privacy.

In this paper, we demonstrate that in the presence of a *context-free taxonomy tree* we can efficiently generate a sanitized release of set-valued data in a differentially private manner with guaranteed utility for counting queries and many other data mining tasks. Unlike the use of taxonomy trees in the *generalization* mechanism for partition-based privacy models, where the taxonomy trees are highly specific to a particular application, the taxonomy tree required in our solution does not necessarily need to reflect the underlying semantics and, therefore, is context-free. This feature makes our approach flexible for applying to various kinds of set-valued datasets.

Contribution. We summarize our contributions as follows.

First, this is the first study of publishing set-valued data via differential privacy. The previous anonymization techniques [5, 16, 19, 29, 30, 36, 37] developed for publishing set-valued data are dedicated to partition-based privacy models. Due to their deterministic nature, they cannot be used for achieving differential privacy. In this paper, we propose a *probabilistic* top-down partitioning algorithm that provides provable utility under differential privacy, one of the strongest privacy models.

Second, this is the first paper that proposes an efficient *non-interactive* approach scalable to high-dimensional set-valued data with guaranteed utility under differential privacy. We stress that our goal is to publish the *data*, not data mining results. Publishing data provides much greater flexibilities for data miners than publishing data mining results. We show that a more efficient and effective solution could be achieved by making use of the underlying dataset, instead of explicitly considering *all* possible outputs as used in the existing works [4, 10, 34, 35]. For a set-valued dataset, it could be done by a top-down partitioning process based on a context-free taxonomy tree. The use of a context-free taxonomy tree makes our approach applicable to all kinds of set-valued datasets. We prove that the result of our approach is (ϵ, δ)-useful for counting queries, which guarantees the usefulness for data mining tasks based on counts, e.g., mining frequent patterns and association rules [17]. We argue that the general idea has a wider application, for example, to relational data in which each attribute is associated with a taxonomy tree. This implies that some traditional data publishing methods, such as TDS [14] and Mondrian [22], could be adapted to satisfy differential privacy.

2. RELATED WORK

Set-Valued Data Publishing. Due to the nature of *high dimensionality* in set-valued data, the extensive research on privacy-preserving data publishing (PPDP) for relational data does not fit well with set-valued data [13]. Some recent papers have started to address the problem of sanitizing set-valued data for the purpose of data mining [5, 11, 16, 19, 29, 30, 36, 37].

Ghinita et al. [16] and Xu et al. [36, 37] divide all items into either *sensitive* or *non-sensitive*, and assume that an adversary's background knowledge is strictly confined to non-sensitive items. Ghinita et al. [16] propose a bucketization-based approach that limits the probability of inferring a sensitive item to a specified threshold, while preserving correlations among items for frequent pattern mining. Xu et al. [37] bound the background knowledge of an adversary to at most p non-sensitive items, and employ global suppression to preserve as many item instances as possible. Xu et al. [36] improve the technique in [37] by preserving frequent itemsets and presenting a border representation. Cao et al. [5] further assume that an adversary may possess background knowledge on sensitive items and propose a privacy notion ρ-uncertainty, which bounds the confidence of inferring a sensitive item from any itemset to ρ.

Terrovitis et al. [29, 30] and He and Naughton [19] eliminate the distinction between sensitive and non-sensitive. Similar to the idea of [36] and [37], Terrovitis et al. [29] propose to bound the background knowledge of an adversary by the maximum number m of items and propose a new privacy model k^m-anonymity, a relaxation of k-anonymity. They achieve k^m-anonymity by a bottom-up global generalization solution. To improve the utility, recently Terrovitis et al. [30] provide a local recoding method for achieving k^m-anonymity. He and Naughton [19] point out that k^m-anonymity provides a weaker privacy protection than k-anonymity and propose a top-down local generalization solution under k-anonymity. We argue that even k-anonymity provides insufficient privacy protection for set-valued data. Evfimievski et al. [11] propose a series of randomization operators to limit the confidence of inferring an item's presence in a dataset with the goal of association rule mining.

Differential Privacy. In the last few years, differential privacy has been gaining considerable attention in various applications. Most of the research on differential privacy concentrates on the interactive setting with the goal of ei-

Table 1: A sample set-valued dataset

TID	Items
t_1	$\{I_1, I_2, I_3, I_4\}$
t_2	$\{I_2, I_4\}$
t_3	$\{I_2\}$
t_4	$\{I_1, I_2\}$
t_5	$\{I_2\}$
t_6	$\{I_1\}$
t_7	$\{I_1, I_2, I_3, I_4\}$
t_8	$\{I_2, I_3, I_4\}$

ther reducing the magnitude of added noise [18, 27] or releasing certain data mining results [2, 3, 12, 21, 23]. Refer to [7] for an overview of recent works on differential privacy. Lately, several works [4, 10, 34, 35] have started to address the use of differential privacy in the non-interactive setting as a substitute for partition-based privacy models. Blum et al. [4] demonstrate that it is possible to circumvent the lower bound results to release *synthetic* private databases that are useful for all queries over a discretized domain from a concept class with polynomial VC-dimension [31]. However, their mechanism is not efficient, taking runtime complexity of $superpoly(|\mathcal{C}|, |I|)$, where $|\mathcal{C}|$ is the size of a concept class and $|I|$ the size of the item universe. This fact makes their mechanism impossible for practical applications. To improve the efficiency, Dwork et al. [10] propose a recursive algorithm of generating a *synthetic* database with runtime complexity of $poly(|\mathcal{C}|, |I|)$. As mentioned earlier, this improvement, however, is still insufficient to handle real-life set-valued datasets. In this paper, we propose an algorithm that is scalable to large real-life set-valued datasets.

Xiao et al. [35] propose a two-step algorithm for *relational data*. It first issues queries for *every* possible combination of attribute values to the PINQ interface [24], and then produces a generalized output using the perturbed dataset returned by PINQ. Apparently, this approach is computationally expensive in the context of set-valued data due to the high dimensionality, which requires issuing a total of $2^{|I|} - 1$ queries. All these works [4, 10, 35] are based on the query model. In contrast, Xiao et al. [34] assume that their algorithm has direct and unconditional access to the underlying *relational data*. They propose a wavelet-transformation based approach that lowers the magnitude of noise than adding independent Laplace noise. Similarly, the algorithm needs to process all possible entries in the entire output domain, which causes a scalability problem for set-valued data.

3. PRELIMINARIES

Let $I = \{I_1, I_2, ..., I_{|I|}\}$ be the universe of items, where $|I|$ is the size of the universe. The multiset $D = \{t_1, t_2, ..., t_{|D|}\}$ denotes a set-valued dataset, where each record $t_i \in D$ is a non-empty subset of I. Table 1 presents an example of set-valued datasets with the item universe $I = \{I_1, I_2, I_3, I_4\}$. An overview of notational conventions is provided in Appendix A.

3.1 Context-Free Taxonomy Tree

A set-valued dataset could be associated with a single *taxonomy tree*. In the classic generalization mechanism, the taxonomy tree required is highly specific to a particular application. This constraint has been considered a major limitation of applying generalization [1]. The reason of requiring an application-specific taxonomy tree is that the release

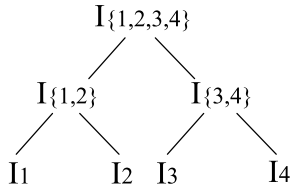


Figure 1: A context-free taxonomy tree of the sample data

contains generalized items that need to be *semantically* consistent with the original items. In our approach, we publish only original items; therefore, the taxonomy tree could be *context free*.

Definition 3.1 (Context-Free Taxonomy Tree). A context-free taxonomy tree is a taxonomy tree, whose internal nodes are a set of their leaves, not necessarily the semantic generalization of the leaves.

For example, Figure 1 presents a context-free taxonomy tree for Table 1, and one of its internal nodes $I_{\{1,2,3,4\}} = \{I_1, I_2, I_3, I_4\}$. We say that an item can be *generalized* to a taxonomy tree node if it is in the node's set. For example, I_1 can be generalized to $I_{\{1,2\}}$ because $I_1 \in \{I_1, I_2\}$.

3.2 Differential Privacy

Differential privacy requires that the removal or addition of a single database record does not significantly affect the outcome of any analysis. It ensures a data record owner that any privacy breach will not be a result of participating in the database since anything that is learnable from the database with his record is also learnable from the one without his record. Formally, differential privacy in the *non-interactive* setting [4] is defined as follow. Here the parameter, α, specifies the degree of privacy offered.

Definition 3.2 (α-differential privacy). A privacy mechanism $\mathcal{A}$ gives α-differential privacy if for any dataset D_1 and D_2 differing on at most one record, and for any possible sanitized dataset $\widetilde{D} \in Range(\mathcal{A})$,

$$Pr[\mathcal{A}(D_1) = \widetilde{D}] \leq e^{\alpha} \times Pr[\mathcal{A}(D_2) = \widetilde{D}] \quad (1)$$

where the probability is taken over the randomness of $\mathcal{A}$.

Two principal techniques for achieving differential privacy have appeared in the literature, one for real-valued outputs [8] and the other for outputs of arbitrary types [25]. A fundamental concept of both techniques is the *global sensitivity* of a function [8] mapping underlying datasets to (vectors of) reals.

Definition 3.3 (Global Sensitivity). For any function $f : D \rightarrow \mathbb{R}^d$, the sensitivity of f is

$$\Delta f = \max_{D_1, D_2} ||f(D_1) - f(D_2)||_1 \quad (2)$$

for all D_1, D_2 differing in at most one record.

Roughly speaking, functions with lower sensitivity are more tolerant towards changes of a dataset and, therefore, allow more accurate differentially private mechanisms.

Laplace Mechanism. For the analysis whose outputs are real, a standard mechanism to achieve differential privacy is to add Laplace noise to the true output of a function. Dwork et al. [8] propose the Laplace mechanism which takes

as inputs a dataset D, a function f, and the privacy parameter α. The magnitude of the noise added conforms to a Laplace distribution with the probability density function $p(x|\lambda) = \frac{1}{2\lambda}e^{-|x|/\lambda}$, where λ is determined by both the global sensitivity of f and the desired privacy level α.

THEOREM 3.1. [8] *For any function $f : D \rightarrow \mathbb{R}^d$ over an arbitrary domain D, the mechanism $\mathcal{A}$*

$$\mathcal{A}(D) = f(D) + Laplace(\Delta f/\alpha) \quad (3)$$

gives α-differential privacy.

For example, for a single counting query Q over a dataset D, returning $Q(D) + Laplace(1/\alpha)$ maintains α-differential privacy because a counting query has a sensitivity 1.

Exponential Mechanism. For the analysis whose outputs are not real or make no sense after adding noise, McSherry and Talwar [25] propose the exponential mechanism that selects an output from the output domain, $r \in \mathcal{R}$, by taking into consideration its score of a given utility function q in a differentially private manner. The exponential mechanism assigns exponentially greater probabilities of being selected to outputs of higher scores so that the final output would be close to the optimum with respect to q. The chosen utility function q should be insensitive to changes in any particular record, that is, has a low sensitivity. Let the sensitivity of q be $\Delta q = \max_{\forall r, D_1, D_2} ||q(D_1, r) - q(D_2, r)||_1$.

THEOREM 3.2. [25] *Given a utility function $q : (D \times \mathcal{R}) \rightarrow \mathbb{R}$ for a dataset D, the mechanism $\mathcal{A}$,*

$$\mathcal{A}(D, q) = \left\{ return\ r\ with\ probability \propto exp(\frac{\alpha q(D, r)}{2\Delta q}) \right\} \quad (4)$$

gives α-differential privacy.

For a sequence of differentially-private computations, its privacy guarantee is provided by the composition properties of differential privacy, namely *sequential composition* and *parallel composition*, which are summarized in Appendix B.

3.3 Utility Metrics

Due to the lower bound results [6, 8, 9], we can *only* guarantee the utility of restricted classes of queries [4] in the non-interactive setting. In this paper, we aim to develop a solution for publishing set-valued data that is useful for *counting queries*.

DEFINITION 3.4 (COUNTING QUERY). For a given itemset $I' \subseteq I$, a counting query Q over a dataset D is defined to be $Q(D) = |\{t \in D : I' \subseteq t\}|$.

We choose counting queries because they are crucial to several key data mining tasks over set-valued data, for example, mining frequent patterns and association rules [17]. In this paper, we employ (ϵ, δ)-*usefulness* [4] to theoretically measure the utility of sanitized data for counting queries.

DEFINITION 3.5 ((ϵ, δ)-USEFULNESS). A privacy mechanism $\mathcal{A}$ is (ϵ, δ)-useful for queries in class $\mathcal{C}$ if with probability $1 - \delta$, for every $Q \in \mathcal{C}$ and every dataset D, for $\widetilde{D} = \mathcal{A}(D)$, $|Q(\widetilde{D}) - Q(D)| \leq \epsilon$.

(ϵ, δ)-usefulness is effective to give an overall estimation of utility, but fails to provide intuitive experimental results. Therefore, in Section 5.1, we experimentally measure the utility of sanitized data for counting queries by *relative error* (see Section 5.1 for more details.).

4. SANITIZATION ALGORITHM

We present a *Diff*erentially-private sanitization algorithm that recursively *Part*itions a given set-valued dataset based on a context-free taxonomy tree (*DiffPart*).

4.1 Partitioning Algorithm

Intuitively, a differentially private release of a set-valued dataset could be generated by adding Laplace noise to a set of counting queries. A simple yet infeasible approach can be achieved by employing Dwork et al.'s method [8]: first generate all distinct itemsets from the item universe; then for each itemset issue a counting query and add Laplace noise to the answer. This approach suffers from two main drawbacks in the context of set-valued data. First, it requires a total of $\sum_{k=1}^{|I|} \binom{|I|}{k} = 2^{|I|} - 1$ queries, where k is the number of items in a query, giving rise to a scalability problem. Second, the noise added to the itemsets that never appear in the original dataset accumulates exponentially, rendering the release useless for data analysis tasks. In fact, these are also the main limitations of other non-interactive approaches [4, 10, 34, 35] when applied to set-valued data. We argue that an efficient solution could be achieved by taking into consideration the underlying dataset. However, attentions must be paid because identifying the set of counting queries based on the input dataset may leak its sensitive information and, therefore, violates differential privacy.

We first provide an overview of *DiffPart*. It starts by creating the context-free taxonomy tree. It then generalizes all records to a single partition with a common representation. We call the common representation the *hierarchy cut*, consisting of a set of taxonomy tree nodes. It recursively distributes the records into *disjoint* sub-partitions with more specific representations in a top-down manner based on the taxonomy tree. For each sub-partition, we determine if it is empty *in a noisy way* and further split the sub-partitions considered "non-empty". Our approach stops when no further partitioning is possible in any sub-partition. We call a partition a *leaf partition* if every node in its hierarchy cut is a leaf of the taxonomy tree. Finally, for each leaf partition, the algorithm asks for its noisy size (the noisy number of records in the partition) to construct the release. Our use of a top-down partitioning process is inspired by its use in [19], but with substantial differences. Their approach is used to generate a *generalized* release satisfying k-anonymity while ours is to identify the set of counting queries used to publish differentially private data.

Algorithm 1 presents our approach in more detail. It takes as inputs the raw set-valued dataset D, the fan-out f used to construct the taxonomy tree, and also the total privacy budget B specified by the data publisher, and returns a sanitized dataset $\widetilde{D}$ satisfying B-differential privacy.

Top-Down Partitioning. The algorithm first constructs the context-free taxonomy tree H by iteratively grouping f nodes from one level to an upper level until a single root is created. If the size of the item universe is not divided by f, smaller groups can be created.

The initial partition p is created by generalizing all records in D under a hierarchy cut of a single taxonomy tree node, namely the root of H. A record can be *generalized* to a hierarchy cut if *every* item in the record can be generalized to a node in the cut and *every* node in the cut generalizes some items in the record. For example, the record $\{I_3, I_4\}$ can be generalized to the hierarchy cuts $\{I_{\{3,4\}}\}$ and $\{I_{\{1,2,3,4\}}\}$,

Algorithm 1 DiffPart

Input: Raw set-valued dataset D; fan-out f; privacy budget B
Output: Sanitized dataset $\widetilde{D}$

$\widetilde{D} \leftarrow \emptyset$;
Construct a taxonomy tree H with fan-out f;
Partition $p \leftarrow$ all records in D;
$p.cut \leftarrow$ the root of H;
$p.\widetilde{B} = B/2$; $p.\alpha = p.\widetilde{B}/|InternalNodes(p.cut)|$;
Add p to an initially empty queue $\mathbb{Q}$;
while $\mathbb{Q} \neq \emptyset$ **do**
    Dequeue p' from $\mathbb{Q}$;
    Sub-partitions $P \leftarrow SubPart\_Gen(p', H)$;
    for each sub-partition $p_i \in P$ **do**
        if p_i is a leaf partition **then**
            $N_{p_i} = NoisyCount(|p_i|, B/2 + p_i.\widetilde{B})$;
            if $N_{p_i} \geq \sqrt{2}C_1/(B/2 + p_i.\widetilde{B})$**then**
                Add N_{p_i} copies of $p_i.cut$ to $\widetilde{D}$;
        else
            Add p_i to $\mathbb{Q}$;
return $\widetilde{D}$;

Procedure 1 SubPart_Gen Procedure

Input: Partition p; taxonomy tree H
Output: Noisy non-empty sub-partitions V of p

Initialize a vector V;
Select a node u from $p.cut$ to partition;
Generate all non-empty sub-partitions to S;
Allocate records in p to S;
for each sub-partition $s_i \in S$ **do**
    $N_{s_i} = NoisyCount(|s_i|, p.\alpha)$;
    if $N_{s_i} \geq \sqrt{2}C_2 \times height(p.cut)/p.\alpha$ **then**
        $s_i.\widetilde{B} = p.\widetilde{B} - p.\alpha$;
        $s_i.\alpha = s_i.\widetilde{B}/|InternalNodes(s_i.cut)|$;
        Add s_i to V;
$j = 1$; $l =$ number of u's children;
while $j \leq 2^l - |S|$ **do**
    $N_j = NoisyCount(0, p.\alpha)$;
    if $N_j \geq \sqrt{2}C_2 \times height(p.cut)/p.\alpha$ **then**
        Randomly generate an empty sub-partition s'_j;
        $s'_j.\widetilde{B} = p.\widetilde{B} - p.\alpha$;
        $s'_j.\alpha = s'_j.\widetilde{B}/|InternalNodes(s'_j.cut)|$;
        Add s'_j to V;
return V;

but *not* $\{I_{\{1,2\}}, I_{\{3,4\}}\}$. The initial partition p is added to an empty queue $\mathbb{Q}$.

For each partition in the queue, we need to generate its sub-partitions and identify the non-empty ones for further partitioning. Due to noise required by differential privacy, a sub-partition cannot be *deterministically* identified as non-empty. Probabilistic operations are needed for this purpose. For each operation, a certain portion of privacy budget is required to obtain the noisy size of a sub-partition based on which we decide whether it is "empty". Algorithm 1 keeps partitioning "non-empty" sub-partitions until leaf partitions are reached.

Example 4.1. Given the dataset in Table 1 and a fan-out value 2, a possible taxonomy tree is presented in Figure 1, and a possible partitioning process is illustrated in Figure 2. Partitions $\{I_{\{3,4\}}\}$, $\{I_{\{1,2\}}, I_3\}$ and $\{I_{\{1,2\}}, I_4\}$ are considered "empty" and, therefore, not further partitioned.

Privacy Budget Allocation. The use of the total privacy budget B needs to be carefully allocated to each probabilistic operation to avoid unexpected termination of the algorithm. Since the operations are used to determine the noisy sizes of the sub-partitions resulted from partition operations, a naive allocation scheme is to bound the maximum number of partition operations needed in the entire algorithm and assign an equal portion to each of them. This approach, however, does not perform well. Instead, we propose a more sophisticated adaptive scheme. We reserve $B/2$ to obtain the noisy sizes of leaf partitions, which are used to construct the release, and use the rest $B/2$ to guide the partitioning process. For each partition, we independently calculate the maximum number of partition operations further needed and assign privacy budget to partition operations based on the number.

The portion of privacy budget assigned to a partition operation is further allocated to the resulting sub-partitions to check their noisy sizes (to see if they are "empty"). Since all sub-partitions from the same partition operation contain *disjoint* records, due to the *parallel composition* property [24], the portion of privacy budget could be used *in full* on each sub-partition. This scheme guarantees that more specific partitions always obtain more privacy budget (see Appendix F.2 for a formal proof), complying with the rationale that more general partitions contain more records and, therefore, are more resistant to a smaller privacy budget.

Theorem 4.1. *Given a non-leaf partition p with a hierarchy cut and an associated taxonomy tree H, the maximum number of partition operations needed to reach leaf partitions is $|InternalNodes(cut)| = \sum_{u_i \in cut} |InternalNodes(u_i, H)|$, where $|InternalNodes(u_i, H)|$ is the number of internal node of the subtree of H rooted at u_i.*

Proof. See Appendix F.1. □

Each partition tracks its unused privacy budget $\widetilde{B}$ and calculates the portion of privacy budget α for the next partition operation. Any privacy budget left from the partitioning process is added to leaf partitions.

Example 4.2. For the partitioning process illustrated in Figure 2, partitions $\{I_1, I_2\}$, $\{I_{\{1,2\}}, I_{\{3,4\}}\}$, $\{I_{\{1,2\}}, I_3, I_4\}$, and $\{I_1, I_2, I_3, I_4\}$ receive privacy budget $5B/6$, $B/6$, $B/6$ and $2B/3$ respectively.

Sub-Partition Generation. "Non-empty" sub-partitions can be identified by either *exponential mechanism* or *Laplace mechanism*. For exponential mechanism, we can get the noisy number N of non-empty sub-partitions, and then use exponential mechanism to extract N sub-partitions by using the number of records in a sub-partition as the score function. This approach, however, does not take advantage of the fact that all sub-partitions contain *disjoint* datasets, resulting in a relatively small privacy budget for each operation and thus less accurate results. For this reason, we employ Laplace mechanism for generating sub-partitions, whose details are presented in Procedure 1.

For a *non-leaf* partition, we generate a candidate set of taxonomy tree nodes from its hierarchy cut, containing all

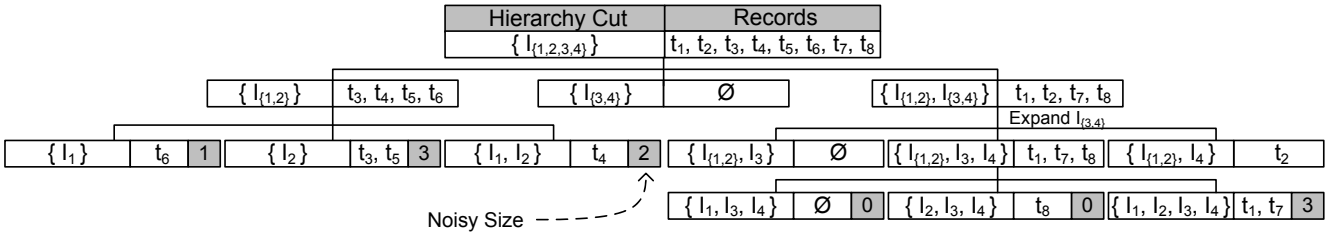


Figure 2: The partitioning process

non-leaf nodes that are of the largest height in H, and then *randomly* select a node u from the set to expand, generating a total of 2^l sub-partitions, where $l \leq f$ is the number of u's children in H. The sub-partitions can be exhaustively generated by replacing u by the combinations of its children. For example, the partition $\{I_{\{1,2\}}\}$ generates three sub-partitions: $\{I_1\}$, $\{I_2\}$ and $\{I_1, I_2\}$. This technique, however, is inefficient.

We propose an efficient implementation by *separately* handling non-empty and empty sub-partitions of a partition p. Non-empty sub-partitions, usually of a small number, need to be explicitly generated. We issue a counting query for the noisy size of each sub-partition by Laplace mechanism. We use the noisy size to make our decision. We consider a sub-partition "non-empty" if its noisy size $\geq \sqrt{2}C_2 \times height(p.cut)/p.\alpha$. We design the threshold as a function of the standard deviation of the noise and the height of p's hierarchy cut, the largest height of all nodes in p's hierarchy cut. The rationale of taking into consideration the height is that more general partitions should have more records to be worth being partitioned. A constant C_2 is added to the function for the reason of efficiency: we want to prune empty sub-partitions as early as possible. While this heuristic is arbitrary, it provides good experimental results on different real-life datasets.

For empty sub-partitions, we do not explicitly generate all possible ones, but employ a *test-and-generate* method: generate a uniformly random empty sub-partition without replacement only if the noisy count of an empty sub-partition's true count 0 is greater than the threshold. To satisfy differential privacy, empty and non-empty sub-partitions must use the *same* threshold. A C_2 value that is slightly greater than 1 can effectively prune most empty sub-partitions without jeopardizing non-empty ones.

For a *leaf* partition, we use the reserved $B/2$ plus the privacy budget left from the partitioning process to obtain its noisy size. To minimize the effect of noise, we add a leaf partition p only if its noisy size $\geq \sqrt{2}C_1/(B/2+p.\widetilde{B})$. Typically, C_1 is a constant in the range of $[1, C_2]$. We argue that since the data publisher has full access to the raw dataset, she could try different C_1 and C_2 values and publish a reasonably good release. We consider how to automatically determine C_1 and C_2 values in future work.

We illustrate how *DiffPart* works in Appendix C.

4.2 Analysis

Privacy Analysis. We prove that Algorithm 1 together with Procedure 1 satisfies B-differential privacy. In essence, the only information obtained from the underlying dataset is the noisy sizes of the partitions (or equivalently, the noisy answers of a set of counting queries). Due to noise, any itemset from the universe may appear in the sanitized release. In the previous work [24], it has been proven that partitioning a dataset by *explicit* user inputs does not violate differential privacy. However, the actual partitioning result should not be revealed as it violates differential privacy. This explains why we need to consider every possible sub-partition and use its noisy size to make decision.

Let a sequence of partitionings that consecutively distributes the records in the initial partition to leaf partitions be a *partitioning chain*. Due to Theorem B.2, the privacy budget used in each partitioning chain is independent of those of other chains. Therefore, if we can prove that the total privacy budget used in each partitioning chain is less than or equal to B, we get the conclusion that Algorithm 1 together with Procedure 1 satisfies B-differential privacy.

Let m be the total number of partitionings in a partitioning chain and n_i the maximum number of partitionings calculated according to Theorem 4.1. We can formalize the proposition to be the following equivalent problem.

$$B \geq \underbrace{\frac{B}{2}\cdot\frac{1}{n_1}}_{\text{first partitioning}} + \underbrace{\frac{B}{2}\cdot(1-\frac{1}{n_1})\cdot\frac{1}{n_2}}_{\text{second partitioning}} + \cdots + \underbrace{\frac{B}{2}\prod_{i=1}^{m-1}(1-\frac{1}{n_i})\cdot\frac{1}{n_m} + \frac{B}{2}}_{\text{last partitioning}}$$

Subject to $n_i \geq n_{i+1}+1$ *and* $n_m = 1$.

Each item of the right hand side (RHS) of the above equation represents the portion of privacy budget allocated to a partition operation. The entire RHS gives the total privacy budget used in the partitioning chain. We prove the correctness of the equation in Appendix F.4. Therefore, our approach satisfies B-differential privacy.

Utility Analysis. We theoretically prove that Algorithm 1 guarantees that the sanitized dataset $\widetilde{D}$ is (ϵ, δ)-useful for counting queries.

THEOREM 4.2. *The result of Algorithm 1 by invoking Procedure 1 is* (ϵ, δ)*-useful for counting queries.*

PROOF. See Appendix F.3. □

Complexity Analysis. The runtime complexity of Algorithm 1 and Procedure 1 is $O(|D| \cdot |I|)$, where $|D|$ is the number of records in the input dataset D and $|I|$ the size of the item universe. The main computational cost comes from the distribution of records from a partition to its sub-partitions. The complexity of distributing the records for a single partition operation is $O(|D|)$ because a partitioning can affect at most $|D|$ records. According to Theorem 4.1, the maximum number of partitionings needed for the entire process is the number of internal nodes in the taxonomy tree H. For a taxonomy tree with a fan-out $f \geq 2$, the number

Table 2: Experimental dataset statistics

Datasets	$\|D\|$	$\|I\|$	$max\|t\|$	$avg\|t\|$
MSNBC	989,818	17	17	1.72
STM	1,210,096	1,012	64	4.82

of internal nodes is $\frac{|I|-1}{f-1}$. Therefore, the overall complexity of our approach is $O(|D| \cdot |I|)$.

Applicability. We discuss the applicability of our approach to other types of data, e.g. relational data, in Appendix D.

5. EXPERIMENTAL EVALUATION

In the experiments, we examine the performance of our algorithm in terms of *utility* for different data mining tasks, namely *counting queries* and *frequent itemset mining*, and *scalability* of handling large set-valued datasets. We compare our approach (*DiffPart*) with Dwork et al.'s method (introduced in Section 4.1 and referred as *Basic* in the following) to show the significant improvement of *DiffPart* on both utility and scalability. The implementation was done in C++, and all experiments were conducted on an Intel Core 2 Duo 2.26GHz PC with 2GB RAM.

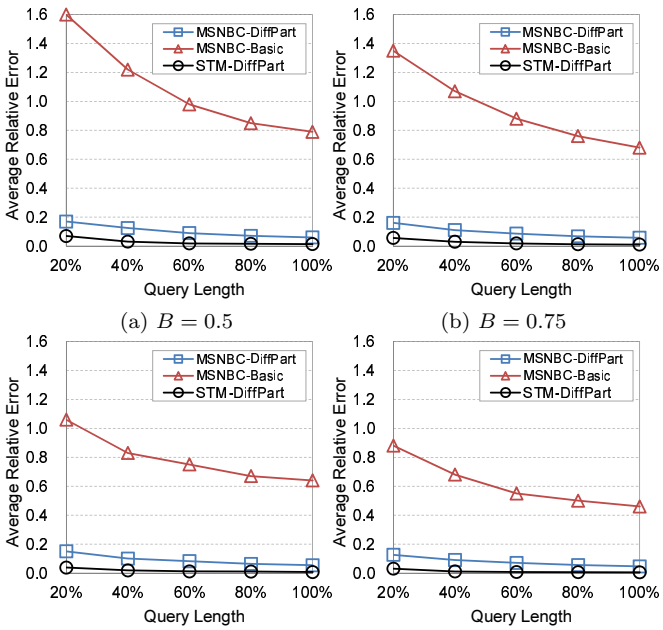
Two *real-life* set-valued datasets, *MSNBC* [1] and *STM* [2], are used in the experiments. *MSNBC* originally describes the URL categories visited by users in time order. We converted it into set-valued data by ignoring the sequentiality, where each record contains a set of URL categories visited by a user. *MSNBC* is of a small universe size. We deliberately choose it so that we can compare *DiffPart* to *Basic*. *STM* records the sets of subway and/or bus stations visited by passengers in Montréal area within a week. It is of a relatively high universe size, for which *Basic* (and the methods in [4, 10, 34, 35]) fails to sanitize. The characteristics of the datasets are summarized in Table 2, where $max|t|$ is the maximum record size and $avg|t|$ the average record size.

5.1 Utility

Following the evaluation scheme from previous works [34], we measure the utility of a counting query Q over the sanitized dataset $\widetilde{D}$ by its *relative error* with respect to the actual result over the raw dataset D. Specifically, the relative error of Q is computed as $\frac{|Q(\widetilde{D})-Q(D)|}{max\{Q(D),s\}}$, where s is a *sanity bound* that weakens the influence of the queries with extremely small selectivities. *Selectivity* is defined as the fraction of records in the dataset satisfying all items in Q [34]. In our experiments, s is set to 0.1% of the dataset size, the same as [34].

In our first set of experiments, we examine the relative error of counting queries with respect to different privacy budgets. For each dataset, we randomly generate 50,000 counting queries with varying numbers of items. We call the number of items in a query the *length* of the query. We divide the query set into 5 subsets such that the query length of the i-th subset is uniformly distributed in $[1, \frac{i \cdot max|t|}{5}]$ and each item is randomly drawn from I. In the following figures, all relative error reported is the average of 10 runs.

Figure 3 shows the average relative error under varying privacy budget B from 0.5 to 1.25 with fan-out $f = 10$ for each query subset. The X-axes represent the maximum query length of each subset in terms of the percentage of $max|t|$. The relative error decreases when the privacy budget increases because less noise is added. The error of *Basic* is significantly larger than that of *DiffPart* in all cases. When the query length decreases, the performance of *Basic* deteriorates substantially because the queries cover exponentially more itemsets that never appear in the original dataset and, therefore, contain much more noise. In contrast, our approach is more stable with different query lengths. It is foreseeable that queries with a length greater than $max|t|$ result in less error. In addition to better utility, *DiffPart* is more efficient than *Basic*, which fails to sanitize the *STM* dataset due to its large universe size.



(a) $B = 0.5$ (b) $B = 0.75$

(c) $B = 1.0$ (d) $B = 1.25$

Figure 3: Average relative error vs. privacy budget

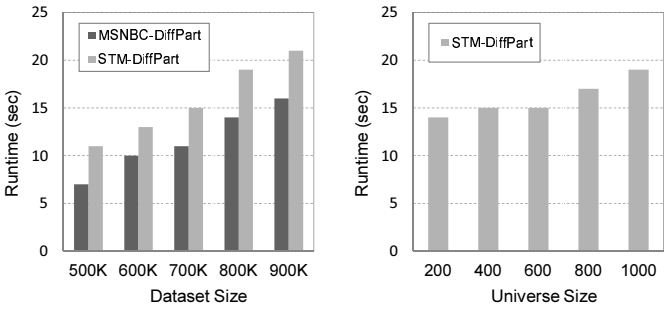
Due to the space limit, we report more experimental results on utility in Appendix E.

5.2 Scalability

We study the scalability of *DiffPart* over large datasets. According to the complexity analysis in Section 4.2, dataset size and universe size are the two factors that dominate the complexity. Therefore, we present the runtime of *DiffPart* under different dataset sizes and universe sizes in Figure 4. Figure 4.a presents the runtime of *DiffPart* under different dataset sizes. We randomly extract records from the two datasets to form smaller test sets and set $B = 1.0$, $f = 10$. As expected, the runtime is linear to the dataset size. Figure 4.b studies how the runtime varies under different universe sizes, where $B = 1.0$ and $f = 10$. Since $MSNBC$ is of a small universe size, we only examine the runtime of *DiffPart* on STM. We generate the test sets by limiting STM's universe size. After reducing the universe size, the sizes of the test sets also decrease. We fix the dataset size under different universe sizes to 800,000. It can be observed again that the runtime scales linearly with the universe size. In summary, our approach scales well to large set-valued datasets. It takes less than 35 seconds to sanitize the STM dataset, whose $|D| = 1,210,096$ and $|I| = 1,012$.

[1] *MSNBC* is publicly available at UCI machine learning repository (http://archive.ics.uci.edu/ml/index.html).

[2] *STM* is provided by *Société de transport de Montréal* (STM) (http://www.stm.info).



(a) Runtime vs. $|D|$ (b) Runtime vs. $|I|$

Figure 4: Runtime vs. different parameters

6. CONCLUSIONS

In this paper, we propose a probabilistic top-down partitioning algorithm for publishing set-valued data in the framework of differential privacy. Compared to the existing works on set-valued data publishing, our approach provides stronger privacy protection with guaranteed utility. The paper also contributes to the research of differential privacy by demonstrating that an efficient *non-interactive* solution could be achieved by carefully making use of the underlying dataset. Our experimental results on different real-life datasets demonstrate the effectiveness and efficiency of our approach.

7. ACKNOWLEDGMENTS

We sincerely thank the reviewers for their insightful comments. The research is supported in part by the new researchers start-up program from Le Fonds québécois de la recherche sur la nature et les technologies (FQRNT), Discovery Grants (356065-2008) and Canada Graduate Scholarships from the Natural Sciences and Engineering Research Council of Canada (NSERC).

8. REFERENCES

[1] C. C. Aggarwal and P. S. Yu. A condensation approach to privacy preserving data mining. In *EDBT*, 2004.

[2] B. Barak, K. Chaudhuri, C. Dwork, S. Kale, F. McSherry, and K. Talwar. Privacy, accuracy, and consistency too: A holistic solution to contingency table release. In *PODS*, 2007.

[3] R. Bhaskar, S. Laxman, A. Smith, and A. Thakurta. Discovering frequent patterns in sensitive data. In *SIGKDD*, 2010.

[4] A. Blum, K. Ligett, and A. Roth. A learning theory approach to non-interactive database privacy. In *STOC*, 2008.

[5] J. Cao, P. Karras, C. Raissi, and K.-L. Tan. ρ-uncertainty inference proof transaction anonymization. In *VLDB*, 2010.

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

[7] C. Dwork. A firm foundation for private data analysis. *Communications of the ACM*, 2011.

[8] C. Dwork, F. McSherry, K. Nissim, and A. Smith. Calibrating noise to sensitivity in private data analysis. In *Theory of Cryptography Conference*, 2006.

[9] C. Dwork, F. McSherry, and K. Talwar. The price of privacy and the limits of lp decoding. In *STOC*, 2007.

[10] C. Dwork, M. Naor, O. Reingold, G. N. Rothblum, and S. Vadhan. On the complexity of differentially private data release: Efficient algorithms and hardness results. In *STOC*, 2009.

[11] A. V. Evfimievski, R. Srikant, R. Agrawal, and J. Gehrke. Privacy preserving mining of association rules. *Inf. Syst.*, 29(4), 2004.

[12] A. Friedman and A. Schuster. Data ming with differential privacy. In *SIGKDD*, 2010.

[13] B. C. M. Fung, K. Wang, R. Chen, and P. S. Yu. Privacy-preserving data publishing: A survey of recent developments. *ACM Computing Surveys*, 42(4), 2010.

[14] B. C. M. Fung, K. Wang, and P. S. Yu. Anonymizing classification data for privacy preservation. *TKDE*, 2007.

[15] S. R. Ganta, S. P. Kasiviswanathan, and A. Smith. Composition attacks and auxiliary information in data privacy. In *SIGKDD*, 2008.

[16] G. Ghinita, Y. Tao, and P. Kalnis. On the anonymization of sparse high-dimensional data. In *ICDE*, 2008.

[17] J. Han and M. Kamber. *Data mining: Concepts and Techniques*. Morgan Kaufmann, San Francisco, 2006.

[18] M. Hay, V. Rastogi, G. Miklau, and D. Suciu. Boosting the accuracy of differentially private histograms through consistency. In *VLDB*, 2010.

[19] Y. He and J. F. Naughton. Anonymization of set-valued data via top-down, local generalization. In *VLDB*, 2009.

[20] D. Kifer. Attacks on privacy and definetti's theorem. In *SIGMOD*, 2009.

[21] A. Korolova, K. Kenthapadi, N. Mishra, and A. Ntoulas. Releasing search queries and clicks privately. In *WWW*, 2009.

[22] K. LeFevre, D. J. Dewitt, and R. Ramakrishnan. Mondrian multidimensional k-anonymity. In *ICDE*, 2006.

[23] A. Machanavajjhala, D. Kifer, J. M. Abowd, J. Gehrke, and L. Vilhuber. Privacy: Theory meets practice on the map. In *ICDE*, 2008.

[24] F. McSherry. Privacy integrated queries: An extensible platform for privacy-preserving data analysis. In *SIGMOD*, 2009.

[25] F. McSherry and K. Talwar. Mechanism design via differential privacy. In *FOCS*, 2007.

[26] A. Narayanan and V. Shmatikov. Robust de-anonymization of large sparse datasets. In *IEEE Symposium on Security and Privacy*, 2008.

[27] A. Roth and T. Roughgarden. Interactive privacy via the median mechanism. In *STOC*, 2010.

[28] L. Sweeney. k-anonymity: a model for protecting privacy. *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems*, 10(5), 2002.

[29] M. Terrovitis, N. Mamoulis, and P. Kalnis. Privacy-preserving anonymization of set-valued data. In *VLDB*, 2008.

[30] M. Terrovitis, N. Mamoulis, and P. Kalnis. Local and global recoding methods for anonymizing set-valued data. *VLDBJ*, 20(1), 2010.

[31] V. Vapnik and A. Chervonenkis. Theory of pattern recognition, 1974. (in Russian).

[32] K. Wang, B. C. M. Fung, and P. S. Yu. Handicapping attacker's confidence: An alternative to k-anonymization. *KAIS*, 11(3), 2007.

[33] R. C.-W. Wong, A. W.-C. Fu, K. Wang, and J. Pei. Anonymiziation-based attacks in privacy-preserving data publishing. *TODS*, 34(2), 2009.

[34] X. Xiao, G. Wang, and J. Gehrke. Differential privacy via wavelet transforms. In *ICDE*, 2010.

[35] Y. Xiao, L. Xiong, and C. Yuan. Differentially private data release through multidimensional partitioning. In *VLDB workshop on SDM*, 2010.

[36] Y. Xu, B. C. M. Fung, K. Wang, A. W. C. Fu, and J. Pei. Publishing sensitive transactions for itemset utility. In *ICDM*, 2008.

[37] Y. Xu, K. Wang, A. W. C. Fu, and P. S. Yu. Anonymizing transaction databases for publication. In *SIGKDD*, 2008.

APPENDIX

A. NOTATIONAL CONVENTIONS

The table below provides a summary of the notational conventions used in this paper.

$\mathcal{A}$	Privacy mechanism
$B, \widetilde{B}$	Privacy budget, unused privacy budget
$\mathcal{C}$	Concept class
$D, \widetilde{D}$	Raw dataset, sanitized dataset
f	Fan-out
I	Item universe
H	Context-free taxonomy tree
N	Noisy count
p	Partition
Q	Counting query
t	Record in a dataset
u	Taxonomy tree node
$\|D\|, \|\widetilde{D}\|, \|I\|$	Raw dataset size, sanitized dataset size, universe size

B. COMPOSITION PROPERTIES

For a sequence of computations, its privacy guarantee is provided by the composition properties. Any sequence of computations that each provides differential privacy in isolation also provides differential privacy in sequence, which is known as *sequential composition* [24]. The implication is that differential privacy is robust to collusions among adversaries.

THEOREM B.1. [24] *Let $\mathcal{A}_i$ each provide α_i-differential privacy. A sequence of $\mathcal{A}_i(D)$ over the dataset D provides $(\sum_i \alpha_i)$-differential privacy.*

In some special cases, in which a sequence of computations are conducted on *disjoint* datasets, the privacy cost does not accumulate, but depends only on the worst guarantee of all computations. This is known as *parallel composition*. This property could and should be used to obtain good performance.

THEOREM B.2. [24] *Let $\mathcal{A}_i$ each provide α_i-differential privacy. A sequence of $\mathcal{A}_i(D_i)$ over a set of disjoint datasets D_i provides $(max(\alpha_i))$-differential privacy.*

C. AN EXAMPLE OF APPLYING ALGORITHM 1

This section provides an example of applying Algorithm 1 and Procedure 1 on the sample dataset in Table 1.

EXAMPLE C.1. Given the sample dataset in Table 1, a fan-out value 2, and the total privacy budget B, *DiffPart* works as follows (see Figure 2 for an illustration). It first creates the context-free taxonomy tree H illustrated in Figure 1 and generalizes all records to a single partition with the hierarchy cut $\{I_{\{1,2,3,4\}}\}$. A portion of privacy budget $B/6$ is allocated to the first partition operation because there are 3 internal nodes in H (and $B/2$ is reserved for leaf partitions).

The algorithm then creates three sub-partitions with the hierarchy cuts $\{I_{\{1,2\}}\}$, $\{I_{\{3,4\}}\}$, and $\{I_{\{1,2\}}, I_{\{3,4\}}\}$ respectively by replacing the node $I_{\{1,2,3,4\}}$ by different combinations of its children, leading t_3, t_4, t_5, and t_6 to the sub-partition $\{I_{\{1,2\}}\}$ and t_1, t_2, t_7 and t_8 to the sub-partition $\{I_{\{1,2\}}, I_{\{3,4\}}\}$. Suppose that the noisy sizes indicate that these two sub-partitions are "non-empty". Further splits are needed on them. There is no need to explore the sub-partition $\{I_{\{3,4\}}\}$ any more as it is considered "empty".

The portions of privacy budget for the next partition operations are *independently* calculated for the two partitions. For the partition $\{I_{\{1,2\}}\}$, there is at most one more partition operation and, therefore, it gets the privacy budget $B/3$; for the partition $\{I_{\{1,2\}}, I_{\{3,4\}}\}$, $B/6$ is allocated as there are still two internal nodes in its hierarchy cut. A further split of $\{I_{\{1,2\}}\}$ creates three leaf partitions, $\{I_1\}$, $\{I_2\}$, and $\{I_1, I_2\}$. For the partition $\{I_{\{1,2\}}, I_{\{3,4\}}\}$, assume that $I_{\{3,4\}}$ is randomly selected to expand. This generates three sub-partitions: $\{I_{\{1,2\}}, I_3\}$, $\{I_{\{1,2\}}, I_4\}$, and $\{I_{\{1,2\}}, I_3, I_4\}$ with t_2 in $\{I_{\{1,2\}}, I_4\}$, and t_1, t_7, t_8 in $\{I_{\{1,2\}}, I_3, I_4\}$. Assume that the partition $\{I_{\{1,2\}}, I_3, I_4\}$ is considered "non-empty". One more partition operation is needed and $B/6$ privacy budget is allocated.

After the last partitioning, we get three more leaf partitions with the hierarchy cuts $\{I_1, I_3, I_4\}$, $\{I_2, I_3, I_4\}$ and $\{I_1, I_2, I_3, I_4\}$. For all leaf partitions, we use the reserved $B/2$ plus the privacy budget left from the partitioning process to calculate their noisy sizes. This implies $5B/6$ for $\{I_1\}$, $\{I_2\}$, and $\{I_1, I_2\}$, and $2B/3$ for $\{I_1, I_3, I_4\}$, $\{I_2, I_3, I_4\}$ and $\{I_1, I_2, I_3, I_4\}$.

One interesting observation is that with the partitioning process, the hierarchy cuts of the sub-partitions resulted from the same partition operation become more similar. For this reason, to some extent the effect of noise for counting queries is mitigated (recall that the mean of noise is 0).

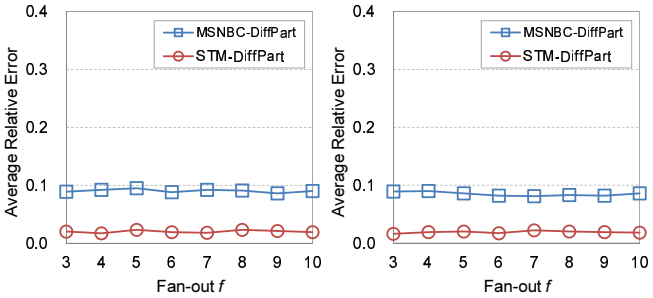
D. DISCUSSION OF APPLICABILITY

It is worthwhile discussing the applicability of our approach in the context of *relational data*. The core of our idea is to limit the output domain by taking into consideration the underlying dataset. In the paper, we propose a probabilistic top-down partitioning process based on a context-free taxonomy tree in order to adaptively narrow down the output domain. For relational data, (categorical) attributes are usually associated with taxonomy trees. Therefore, a similar probabilistic partitioning process could be used. The difference is that the partitioning process needs to be conducted by considering the correlations among multiple taxonomy trees. In this case, exponential mechanism could be used in each partition operation to choose an attribute to split. Different heuristics (e.g. information gain, gini index or max) could be used as the score function. Following the idea, we maintain that our idea could adapt existing deterministic sanitization techniques, such as TDS [14] and Mondrian [22], to satisfy differential privacy. This approach would outperform existing works [4, 10, 34, 35] on publishing relational data in the framework of differential privacy in terms of both utility and efficiency for the same reasons explained in this paper. We consider it in our future work.

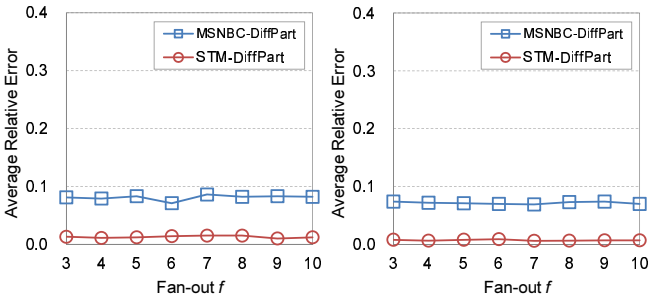
E. ADDITIONAL EXPERIMENTS

In this section, we present additional experimental results of the utility of sanitized data for counting queries and frequent itemset mining.

Counting Query. We continue to study the effect of fan-out, universe size and dataset size on relative error.

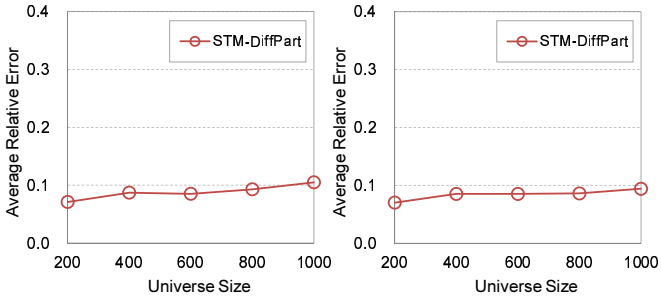


(a) $B = 0.5$ (b) $B = 0.75$

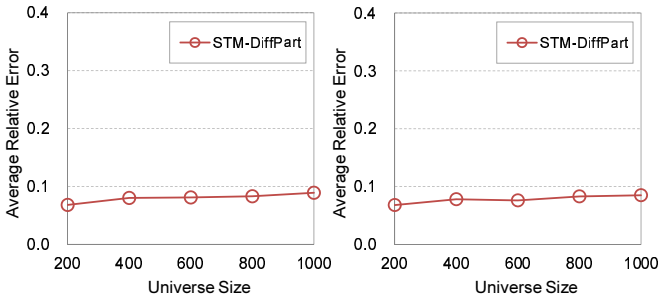


(c) $B = 1.0$ (d) $B = 1.25$

Figure 5: Average relative error vs. fan-out



(a) $B = 0.5$ (b) $B = 0.75$



(c) $B = 1.0$ (d) $B = 1.25$

Figure 6: Average relative error vs. universe size

Figure 5 illustrates the average relative error under different values of fan-out f with privacy budget B ranging from 0.5 to 1.25 while fixing the query length to be $60\% \cdot max|t|$. In general, *DiffPart* generates relatively stable results for different fan-out values. For smaller fan-out values, each partitioning receives less privacy budget; however, there are more levels of partitionings, which increases the chance of pruning more empty partitions. The fact makes the relative error of smaller fan-out values comparable to that of larger fan-out values. The insensitivity of our approach to different fan-out values is a desirable property, which makes a data publisher easier to obtain a good release.

Figure 6 presents the average relative error under different universe sizes with privacy budget B varying from 0.5 to 1.25. We set the fan-out $f = 10$ and fix the query length to 10 (we deliberately choose a small length to make the difference more observable). Since $MSNBC$ is of a small universe size, we only examine the performance of *DiffPart* on STM. We generate the test datasets in a similar setting to that of Figure 4.b. To make a fair comparison, we fix the dataset size under different universe sizes to 800,000. We can observe that the average relative error decreases when the universe size becomes smaller, because there is a greater chance to have more records falling into a partition, making the partition more resistant to larger noise. We can also observe that the datasets with smaller universe sizes obtain more stable relative error under varying privacy budgets. This is due to the same reason that smaller universe sizes result in partitions with larger sizes, which are less sensitive to varying privacy budgets.

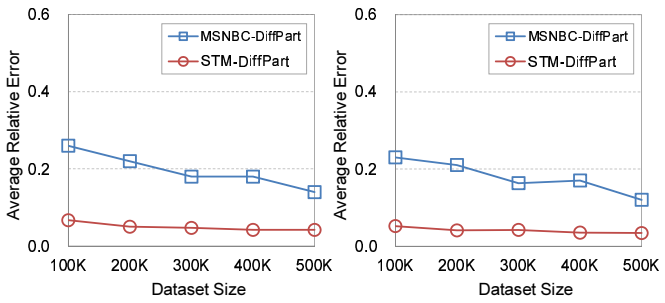
In theory, a dataset has to be large enough to obtain good utility under differential privacy. We experimentally study how the utility varies under different dataset sizes on the two real-life set-valued datasets. We generate the test datasets in a similar setting to that of Figure 4.a and present the results in Figure 7, where B varies from 0.5 to 1.25, $f = 10$, and the query length is $60\% \cdot max|t|$. It can be observed that the two datasets behave differently to varying dataset sizes. The relative error of $MSNBC$ improves significantly when the privacy budget increases, while the change of STM's error is small. This indicates the fact that when the dataset size is not large enough, the distribution of the underlying records is key to the performance. In addition, we can observe that when the privacy budget is small, the error is more sensitive to the dataset size. It is because the number of records in a partition needs to be greater than the magnitude of noise (which is inversely proportion to the privacy budget) in order to obtain good utility.

Frequent Itemset Mining. We further validate the utility of sanitized data by frequent itemset mining, which is a more concrete data mining task. Given a positive number K, we calculate the top K most frequent itemsets on the raw dataset D and the sanitized dataset $\widetilde{D}$ respectively and examine their similarity. Let $F_K(D)$ denote the set of top K itemsets calculated from D and $F_K(\widetilde{D})$ the set from $\widetilde{D}$. For a frequent itemset $F_i \in F_K(D)$, let $sup(F_i, F_K(D))$ denote its support in $F_K(D)$ and $sup(F_i, F_K(\widetilde{D}))$ denote its support in $F_K(\widetilde{D})$. If $F_i \notin F_K(\widetilde{D})$, $sup(F_i, F_K(\widetilde{D})) = 0$. We define the utility metric to be

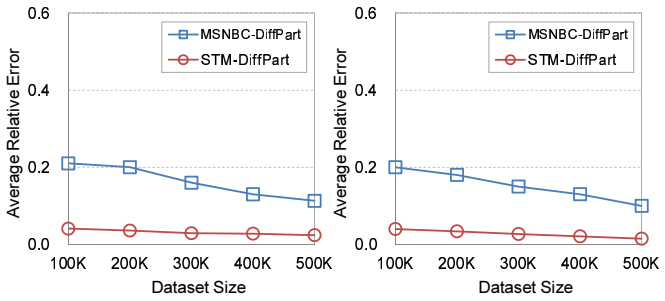
$$1 - \frac{\sum_{F_i \in F_K(D)} \frac{|sup(F_i, F_K(D)) - sup(F_i, F_K(\widetilde{D}))|}{sup(F_i, F_K(D))}}{K},$$

where 1 means that $F_K(D)$ is identical to $F_K(\widetilde{D})$ (even the support of every frequent itemset); 0 means that $F_K(D)$ and $F_K(\widetilde{D})$ are totally different. Specifically, we employ *MAFIA* [3] to mine frequent itemsets.

[3]A maximal frequent itemset mining tool, available at http://himalaya-tools.sourceforge.net/Mafia/

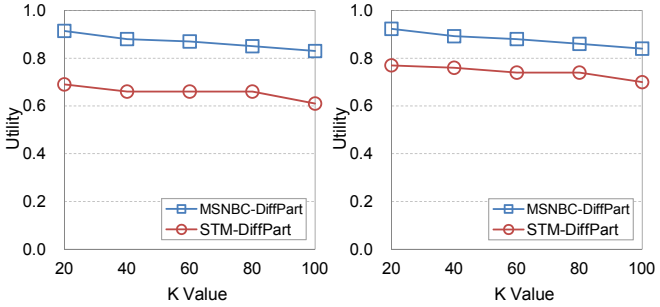


(a) $B = 0.5$ (b) $B = 0.75$

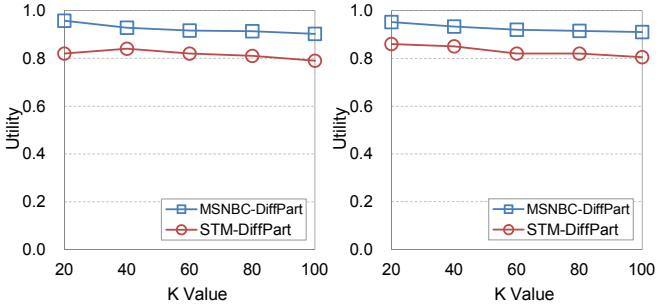


(c) $B = 1.0$ (d) $B = 1.25$

Figure 7: Average relative error vs. dataset size



(a) $B = 0.5$ (b) $B = 0.75$



(c) $B = 1.0$ (d) $B = 1.25$

Figure 8: Utility for frequent itemset mining

In Figure 8, we study the utility of sanitized data for frequent itemset mining under different privacy budgets and different K values with $f = 10$. We observe two general trends from the experimental results. First, the privacy budget has a direct impact on frequent itemset mining. A higher budget results in better utility since the partitioning process is more accurate and less noise is added to leaf partitions. The differences of the supports of top K frequent itemsets between $F_K(D)$ and $F_K(\widetilde{D})$ actually reflect the performance of *DiffPart* for counting queries of extremely small length (because the top-K frequent itemsets are usually of a small length). We can observe that the utility loss (the difference between $F_K(D)$ and $F_K(\widetilde{D})$) is less than 30% except the case $B = 0.5$ for STM. Second, utility decreases when K value increases. When K value is small, in most cases the sanitized datasets are able to give the identical top-K frequent itemsets as the raw datasets, and the utility loss is mainly caused by the differences of the supports. When K value becomes larger, there are more false positives (itemsets wrongly included in the output) and false drops (itemsets mistakenly excluded), resulting in worse utility. Nevertheless, the utility loss is still less than 22% when $K = 100$ and $B \geq 1.0$ on both datasets.

F. PROOFS

F.1 Proof of Theorem 4.1

THEOREM 4.1. *Given a non-leaf partition p with a hierarchy cut and an associated taxonomy tree H, the maximum number of partition operations needed to reach leaf partitions is $|InternalNodes(cut)| = \sum_{u_i \in cut} |InternalNodes(u_i, H)|$, where $|InternalNodes(u_i, H)|$ is the number of internal node of the subtree of H rooted at u_i.*

PROOF. Given a partition p, our algorithm selects one non-leaf taxonomy tree node from its hierarchy cut to expand at a time. Our algorithm stops when every non-leaf taxonomy tree node in p's hierarchy cut is specialized to a leaf node. For a non-leaf node u in the hierarchy cut, *in the worst case*, it will be replaced by the combination containing all its children. If the children are not leaf node, they need to be split, and *in the worst case* again, it will be replaced by the combination containing all its children. That is, we need to go through all internal nodes of the subtree of H rooted at u. Therefore, in order to make all non-leaf nodes in p's hierarchy cut to leaf nodes, we need, in the worst case, $\sum_{u_i \in cut} |InternalNodes(u_i, H)|$ partitionings (partition operations).

Take the dataset in Table 1 as an example. Consider a partition with the hierarchy cut $\{I_{\{1,2,3,4\}}\}$. After the first partitioning, the sub-partition with the hierarchy cut $\{I_{\{1,2\}}, I_{\{3,4\}}\}$ represents the worst case. Suppose $I_{\{1,2\}}$ is selected to split, the sub-partition with the hierarchy cut $\{I_1, I_2, I_{\{3,4\}}\}$ presents the worst case. After that, we need one more split on $I_{\{3,4\}}$. Therefore, in the worst case, the total number of partition operations required is 3, which is the number of internal nodes of the taxonomy tree in Figure 1. □

F.2 Proof of Adaptive Privacy Budget Allocation Scheme

We prove that our adaptive allocation scheme always assigns more privacy budget to more specific partitions below. Let n_i be the maximum number of partition operations calculated according to Theorem 4.1. Let $\frac{B}{2}\prod_{i=1}^{m-2}(1-\frac{1}{n_i})\cdot\frac{1}{n_{m-1}}$ be the privacy budget assigned to a partition and $\frac{B}{2}\prod_{i=1}^{m-1}(1-\frac{1}{n_i})\cdot\frac{1}{n_m}$ the privacy budget assigned to its sub-partitions, which are more specific. We have $n_i \geq n_{i+1}+1$ because the maximum number of partition operations further needed for a partition is always one more than that of its sub-partitions (we need *at least* one more partition op-

eration to split it to its sub-partitions). We can observe the following.

$$\begin{aligned}\frac{B}{2}\prod_{i=1}^{m-1}(1-\frac{1}{n_i})\cdot\frac{1}{n_m} &= \frac{B}{2}\prod_{i=1}^{m-2}(1-\frac{1}{n_i})\cdot\frac{n_{m-1}-1}{n_{m-1}}\cdot\frac{1}{n_m}\\ &\geq \frac{B}{2}\prod_{i=1}^{m-2}(1-\frac{1}{n_i})\cdot\frac{n_m}{n_{m-1}}\cdot\frac{1}{n_m}\\ &= \frac{B}{2}\prod_{i=1}^{m-2}(1-\frac{1}{n_i})\cdot\frac{1}{n_{m-1}}\end{aligned}$$

Using transitivity, we conclude that more specific partitions always receive more privacy budget.

F.3 Proof of Theorem 4.2

THEOREM 4.2. *The result of Algorithm 1 by invoking Procedure 1 is (ϵ,δ)-useful for counting queries.*

PROOF. Given any counting query Q that covers up to m distinct itemsets in the entire output domain, the accurate answer of Q over the input dataset D is $Q(D)=\sum_{i=1}^{m}Q(I_i)$, where I_i is the itemset covered by Q; the answer of Q over $\widetilde{D}$ is $Q(\widetilde{D})=\sum_{i=1}^{m}(Q(I_i)+N_i)$, where N_i is the noise added to I_i. By the definition of (ϵ,δ)-usefulness, to prove Theorem 4.2 is to prove that with a probability $1-\delta$,

$$\begin{aligned}|Q(\widetilde{D})-Q(D)| &= |\sum_{i=1}^{m}(Q(I_i)+N_i)-\sum_{i=1}^{m}Q(I_i)|\\ &= |\sum_{i=1}^{m}N_i|\\ &\leq \sum_{i=1}^{m}|N_i|\\ &\leq \epsilon\end{aligned}$$

We have the following observations.

- For I_i such that $I_i\notin D\cap I_i\notin\widetilde{D}$, $N_i=0$. Let the size of such I_i be $m'\leq m$.
- For I_i such that $I_i\in D\cap I_i\in\widetilde{D}$, $N_i\sim Lap(1/\bar{B})$, where $\bar{B}=B/2+\widetilde{B}$.
- For I_i such that $I_i\notin D\cap I_i\in\widetilde{D}$, $N_i\sim Lap(1/\bar{B})$, where $\bar{B}=B/2+\widetilde{B}$.
- For I_i such that $I_i\in D\cap I_i\notin\widetilde{D}$, $N_i\sim Lap(1/\beta)+\gamma$, where $\beta=B/(2\cdot|InternalNodes(H)|)\leq\bar{B}$ (the smallest privacy budget used in the entire partitioning process) and $\gamma=\sqrt{2}C_2\log_f|I|/\beta$ is introduced by the threshold in Algorithm 1 and Procedure 1.

Therefore, we need to prove that with probability $1-\delta$,

$$\begin{aligned}\sum_{i=1}^{m}|N_i|=\sum_{i=1}^{m-m'}|N_i| &\leq \sum_{i=1}^{m-m'}(|Y_i|+\gamma)\\ &\leq \sum_{i=1}^{m-m'}|Y_i|+(m-m')\cdot\gamma\\ &\leq \epsilon\end{aligned}$$

where Y_i is a random variable i.i.d from $Lap(1/\beta)$. If every $|Y_i|\leq\epsilon_1$ where $\epsilon_1=\frac{\epsilon}{m-m'}-\gamma$, we have $\sum_{i=1}^{m}|N_i|\leq\epsilon$. Let us call the event that any single $|Yi|>\epsilon_1$ a FAILURE. We can calculate

$$Pr[FAILURE]=2\int_{\epsilon_1}^{\infty}\frac{\beta}{2}exp(-\frac{\beta x}{2})dx=exp(-\beta\epsilon_1)$$

Since every Y_i is independent and identically distributed, we have

$$\begin{aligned}Pr[\sum_{i=1}^{m}|N_i|\leq\epsilon] &= Pr[\sum_{i=1}^{m-m'}|Y_i|\leq\epsilon-(m-m')\cdot\gamma]\\ &\geq (1-Pr[FAILURE])^{m-m'}\\ &\geq (1-exp(-\beta\epsilon_1))^{m-m'}\end{aligned}$$

In [35], it has been proven that

$$(1-exp(-\beta\epsilon_1))^{m-m'}\geq 1-(m-m')exp(-\beta\epsilon_1)$$

Therefore, we get

$$\begin{aligned}Pr[\sum_{i=1}^{m}|N_i|\leq\epsilon] &\geq 1-(m-m')exp(-\beta\epsilon_1)\\ &\geq 1-(m-m')exp(\beta\gamma-\frac{\beta\epsilon}{m-m'})\end{aligned}$$

This completes the proof. □

F.4 Proof of Privacy Analysis

The equation needed to prove in Section 4.2 can be rewritten as the following equivalent equation:

$$\frac{1}{n_1}+\sum_{i=1}^{m-1}(\prod_{j=1}^{i}(1-\frac{1}{n_j})\cdot\frac{1}{n_{i+1}})\leq 1$$

$$\textit{Subject to } n_i\geq n_{i+1}+1 \textit{ and } n_m=1.$$

We add one more non-negative item $\prod_{i=1}^{m-1}(1-\frac{1}{n_i})\cdot(1-\frac{1}{n_m})$ to the left hand side of the above equation. We obtain the following.

$$\begin{aligned}&\frac{1}{n_1}+\sum_{i=1}^{m-1}(\prod_{j=1}^{i}(1-\frac{1}{n_j})\cdot\frac{1}{n_{i+1}})+\prod_{i=1}^{m-1}(1-\frac{1}{n_i})\cdot(1-\frac{1}{n_m})\\ =\ &\frac{1}{n_1}+\sum_{i=1}^{m-2}(\prod_{j=1}^{i}(1-\frac{1}{n_j})\cdot\frac{1}{n_{i+1}})+\prod_{i=1}^{m-1}(1-\frac{1}{n_i})\cdot\frac{1}{n_m}\\ &+\prod_{i=1}^{m-1}(1-\frac{1}{n_i})\cdot(1-\frac{1}{n_m})\\ =\ &\frac{1}{n_1}+\sum_{i=1}^{m-2}(\prod_{j=1}^{i}(1-\frac{1}{n_j})\cdot\frac{1}{n_{i+1}})+\prod_{i=1}^{m-1}(1-\frac{1}{n_i})\\ =\ &\frac{1}{n_1}+\sum_{i=1}^{m-2}(\prod_{j=1}^{i}(1-\frac{1}{n_j})\cdot\frac{1}{n_{i+1}})\\ &+\prod_{i=1}^{m-2}(1-\frac{1}{n_i})\cdot(1-\frac{1}{n_{m-1}})\\ =\ &\cdots\\ =\ &1\end{aligned}$$

This completes the proof.

Since $n_m=1$, we can get that the item added above $\prod_{i=1}^{m-1}(1-\frac{1}{n_i})\cdot(1-\frac{1}{n_m})=0$. This indicates that our allocation scheme makes full use of the total privacy budget.