Differentially-Private Multidimensional Data Publishing

Khalil Al-Hussaeni, Concordia University
Benjamin C. M. Fung, McGill University
Farkhund Iqbal, Zayed University
Junqiang Liu, Zhejiang Gongshang University
Patrick C. K. Hung, University of Ontario Institute of Technology

Various organizations collect data about individuals for various reasons, such as service improvement. In order to mine the collected data for useful information, data publishing has become a common practice among those organizations and data analysts, research institutes, or simply the general public. The quality of published data significantly affects the accuracy of the data analysis, and thus affects decision making at the corporate level. In this study, we explore the research area of privacy-preserving data publishing, i.e., publishing high-quality data without compromising the privacy of the individuals whose data are being published. Syntactic privacy models, such as k-anonymity, impose syntactic privacy requirements and make certain assumptions about an adversary's background knowledge. To address this shortcoming, we adopt *differential privacy*, a rigorous privacy model that is independent of any adversary's knowledge and insensitive to the underlying data. The published data should preserve individuals' privacy, yet remain useful for analysis. To maintain data utility, we propose *DiffMulti*, a workload-aware and differentially-private algorithm that employs *multidimensional generalization*. We devise an efficient implementation to the proposed algorithm and use a real-life data set for experimental analysis. We evaluate the performance of our method in terms of data utility, efficiency, and scalability. When compared to closely-related existing methods, *DiffMulti* significantly improved data utility; in some cases, by orders of magnitude.

Key Words: Data sharing, privacy protection, differential privacy, multidimensional generalization

1. INTRODUCTION

Data gathering has been witnessing an exponential growth thanks to modern advancement in information technology. The possession of collected data gives power to the data holder by enhancing data analysis and aiding in decision making. Examples include government agencies collecting census data for demographic analysis to provide better social services; transport authorities collecting trajectories for traffic analysis to enhance the city's transportation network; hospitals collecting patients' symptoms for better future diagnosis; and online service providers collecting online surfing habits for building and enhancing recommendation systems.

There are cases where the data holder may not always have the expertise to perform the required data analysis [Hafner 2006], or the collected data must be published as mandated by law [Carlisle et al. 2007]. Consequently, data publishing has become a common practice for the mutual benefit of the data holder and the data recipient. It is of no less importance that the privacy of individuals whose data are being published should be safe-guarded. To bridge the gap between these two seemingly conflicting requirements, several privacy models have been proposed in the literature. We categorize these models into two types: *syntactic* and *semantic* models.

In the context of syntactic privacy, the output data set has to comply with a syntactic privacy requirement. A prime example is k-anonymity and its various extensions [Samarati 2001; Sweeney 2002; Machanavajjhala et al. 2006; Li et al. 2007] by

Authors' addresses: K. Al-Hussaeni, CIISE, Concordia University, Montreal, Canada, H3G 1M8; B. C. M. Fung, School of Information Studies, McGill University, Montreal, Canada, H3A 1X1; F. Iqbal, College of Technological Innovation, Zayed University, U.A.E.; J. Liu, School of Information and Electronic Engineering, Zhejiang Gongshang University, Hangzhou, China, 310018; P. C. K. Hung, Faculty of Business and Information Technology, University of Ontario Institute of Technology, Oshawa, Ontario, Canada, L1H 7K4; emails: k_alhus@ciise.concordia.ca, ben.fung@mcgill.ca, farkhund.iqbal@zu.ac.ae, jjliu@alumni.sfu.ca, patrick.hung@uoit.ca.

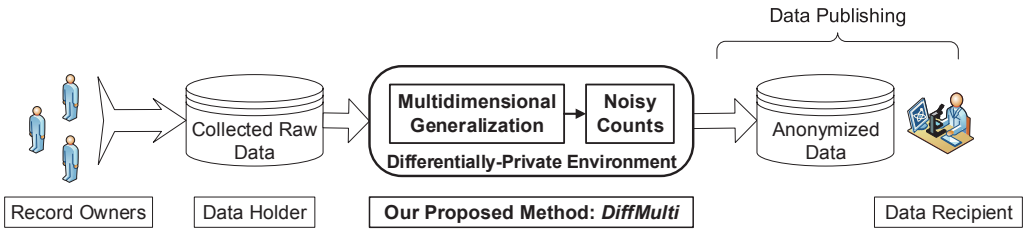


Fig. 1. Data publishing under the non-interactive setting

which every record in the output data set has to be hidden within a group of k records. Syntactic privacy, though effective, only reduces the possibility of privacy attacks by making certain assumptions about an attacker's background knowledge about the individuals in the published data. Moreover, syntactic privacy is inherently prone to certain attacks, such as *minimality attack* [Wong et al. 2007], *composition attack* [Ganta et al. 2008], and *deFinetti attack* [Kifer 2009]. Therefore, we adopt *differential privacy* [Dwork 2008; 2011], a semantic privacy model that provides resistance against syntactic-based privacy attacks.

Differential privacy is a probabilistic privacy model that provides provable privacy guarantees and works independently of any attacker's background knowledge. Intuitively, differential privacy ensures that the output of any analysis performed on the published data does not overly depend on any single participant. Individual's privacy is protected in a sense that her participation (or withdrawal) would not significantly change the output of the analysis. This indirectly removes the privacy concerns of both the participants and the data holders.

Differential privacy introduces the concept of *privacy budget* ϵ. A higher budget results in a more accurate (less noisy) differentially-private output. The literature has defined two settings wherein ϵ can be utilized: *interactive* and *non-interactive*. In the interactive setting [Dwork et al. 2006; Friedman and Schuster 2010], the raw data is kept in the data holder's possession and a data miner/requester issues a set of queries to which the data holder provides differentially-private answers. Each query would consume a portion of ϵ. Once the entire budget has been consumed, the data holder can no longer receive more queries and the database has to shut down completely. Whereas in the non-interactive setting [Barak et al. 2007; Xiao et al. 2011b; Hay et al. 2010], the data holder utilizes the entire privacy budget to anonymize the whole raw data set into an ϵ-differentially-private version, which is then published without restriction or limitation on data usage. In many real-life data sharing scenarios, publishing the data is far more convenient due to the flexibility it gives to the data recipient in terms of analysis power. In this paper, we focus on the non-interactive setting.

Fig. 1 illustrates an overview of privacy-preserving data publishing scenario under the non-interactive setting. In general, a *data holder* collects data from individual *record owners* and wants to release the collected data to a *data recipient* for data analysis without compromising the privacy of the record owners. The research problem studied in this paper is how to convert the raw data into a differentially-private version, via *multidimensional generalization* and *noise addition*, while maintaining data utility in the published data.

1.1. Motivation

Methods based on the non-interactive setting mainly rely on publishing a noisy version of the contingency table of the raw data set [Barak et al. 2007; Ding et al. 2011; Xiao et al. 2011b; Cormode et al. 2012; Qardaji et al. 2013b]. In other words, a true count of

Table I. Raw data table

Continent	YoB	Class
North_America	1947	Y
Australia	1953	Y
North_America	1955	N
Europe	1957	N
Asia	1959	N
North_America	1968	Y

every possible combination of the data set's domain values on every attribute is first generated, and then a noise is added to every such count to satisfy differential privacy. Finally, all noisy counts are published. Although the published data is differentially-private, we reason that this approach yields extremely distorted data. If the data set is high dimensional, then the true counts diminish to very small amounts, making the added noise very large in comparison. In this paper, we do not adopt this approach, as it yields highly distorted data that are far from being useful for any further analysis. We further validate this observation when we evaluate our method in Section 5.

To enhance the utility of the published data, we *generalize* the raw data by replacing raw domain values with less specific yet semantically consistent values, as dictated by a pre-defined generalization hierarchy. Fig. 2 shows the generalization hierarchy of the attributes of Table I in the form of taxonomy trees. The literature has defined two types of generalization: *single-dimensional* and *multidimensional*. In this paper, we enforce multidimensional generalization, backed up by the study in [LeFevre et al. 2006] that suggests that the multidimensional type of generalization significantly improves the utility of the anonymized data over the single-dimensional type. We note that the previous work in [LeFevre et al. 2006] achieves multidimensional generalization by applying the k-anonymity privacy model, whereas we adopt differential privacy as our privacy model. We reason that it is possible to achieve high-quality generalized data using differential privacy. The following is an illustrating example.

EXAMPLE 1.1. *The* Population Data BC *(*PopData*)*[1] *is a not-for-profit organization that collects health-related data from a variety of sources with the hope of harnessing such data for the advancement of human well-being.* PopData *does not possess the means to conduct research on the collected data; therefore, it offers data sharing among researchers* upon request*. Even though patient-specific identifiers, such as Name and Address, are removed from the published data, data sharing is a lengthy process heavily based on trusting the requester (i.e., data recipient). In an attempt to increase the benefits of the collected data, we propose an alternative data-sharing solution that is not based on trust, thus allowing a wider range of data recipients. Our proposed solution produces high-quality anonymized data and guarantees patients' privacy.*

Let Table I represent a group of patients' raw data. The Class *attribute, having Y and N as the only domain values, indicates whether a patient has a chronic disease. Fig. 2 shows the domain hierarchy of attributes $Continent$ and YoB (Year of Birth). Fig. 3 (a) to (c) depict a spatial representation of Table I and its two generalized versions, respectively. Fig. 3 (c) is the result of applying our anonymization method.*

Every rectangle in Fig. 3 (b) and (c) represents a generalization region in the domain space, and every solid circle represents a record from Table I. Without loss of generality, let the added noise to every region due to differential privacy be equal to 1 or 0. The added noise represents a synthetic record, and is indicated by an empty circle. Under single-dimensional generalization, all *the raw values on the $Continent$ attribute have been generalized to the topmost general value $Any\_Continent$. On the other hand,*

[1] https://www.popdata.bc.ca/

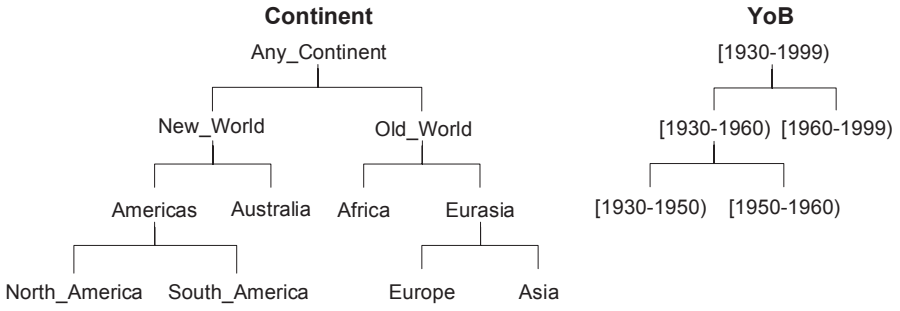
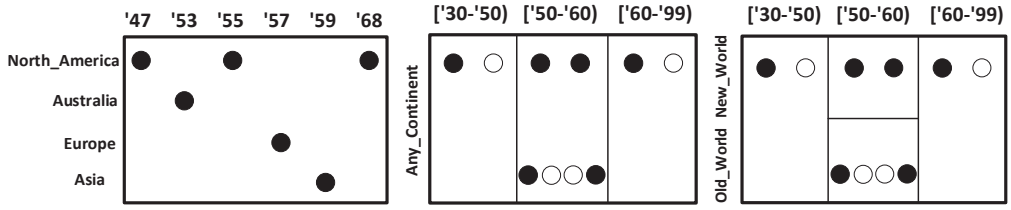


Fig. 2. Taxonomy trees



(a) Raw data (b) Single-dimensional (c) Multidimensional

Fig. 3. Spatial representations of Table I and its diff-priv generalizations

the multidimensional approach generalizes some *raw values to* $Any\_Continent$ *while other raw values have been replaced with more specific hierarchical values, namely,* New_World *and* Old_World*. Thus, the multidimensional approach inflicts less data distortion due to generalization than the single-dimensional approach.* ■

In this paper, we investigate a differentially-private multidimensional solution for data release.

1.2. Contributions

To our knowledge, this is the first work to propose a concrete differentially-private algorithm that employs multidimensional generalization for relational data release in the non-interactive setting. Previous related endeavors have been proposed for publishing high-quality anonymized data [LeFevre et al. 2006; Mohammed et al. 2011; Qardaji and Li 2012; Zhang et al. 2014]. Single-dimensional generalization has been used in [Mohammed et al. 2011] to publish differentially-private data. However, we reason that a multidimensional approach greatly increases data utility, as suggested by [LeFevre et al. 2006]. Even though the proposed work in [LeFevre et al. 2006] is a multidimensional partitioning approach, the guiding privacy model is k-anonymity, which is susceptible to syntactic-based privacy attacks [Wong et al. 2007; Ganta et al. 2008; Kifer 2009]. We argue that stronger privacy guarantees, i.e., through differential privacy, can be achieved without compromising data utility. Previous work [Qardaji and Li 2012] proposed a partition-based general framework capable of publishing differentially-private data; however, it does not scale for large data sets [Qardaji et al. 2014]. Other techniques [Zhang et al. 2014] utilize Bayesian networks to release differentially-private data that approximate the distribution of the high-dimensional input data. In this paper, we evaluate the performance of our proposed algorithm in terms of data utility, efficiency, and scalability. Experimental results of running comparisons between our method and the aforementioned techniques range from our

method producing comparable results at worst to performing by an order of magnitude better at best.

We summarize our contributions as follows:

— We utilize a top-down specialization approach that provides efficient multidimensional partitioning of regions in the domain space of the data set. Given a region to be further partitioned, this approach provides direct access to the records in that region without having to scan the entire data set. This property adds a scalable aspect to our method when anonymizing large data sets.

— We argue that data records are unlikely to be evenly distributed across the domain space of the data set. Hence, our proposed method gives more attention to dense regions in order to yield less abstract data for more accurate analysis of the output differentially-private data.

— Our proposed method performs multidimensional specialization in a differentially-private way on both categorical and numerical attributes. Depending on the target workload, a carefully-selected categorical value is split in accordance with a pre-defined taxonomy tree, whereas a proper numerical split point is dynamically determined for every multidimensional region throughout runtime. This releases the data holder from the burden of performing extensive preprocessing of the data set.

— We carry out extensive experiments and compare with closely-related methods in the literature. Results suggest that our proposed multidimensional algorithm is capable of significantly improving data utility without compromising the rigorous requirement of differential privacy.

The rest of the paper is organized as follows. Section 2 is a literature review of the subject. Differential privacy and generalization are discussed and formally defined in Section 3. Section 4 details our proposed algorithm. Section 5 presents an experimental evaluation. Lastly, Section 6 concludes the paper.

2. RELATED WORK

We categorize the related works and explain the difference between each category and our work.

Researchers have dedicated a great amount of work to propose and improve *syntactic* privacy models. In this context, the output data set must adhere to a given syntactic privacy condition in order to restrict the feasibility of attacks on records and sensitive values. A prime example of syntactic privacy is k-anonymity [Samarati 2001; Sweeney 2002] and its various extensions such as t-closeness [Li et al. 2007] and ℓ-diversity [Machanavajjhala et al. 2006].

Several privacy-preserving algorithms have been proposed for classification analysis [Weiss and Kulikowski 1991]. Iyengar [2002] was the first to address this issue by devising a genetic algorithm, which proved to be costly. In his work, classification accuracy on training data was measured using his proposed *classification metric*, which was later used by Bayardo and Agrawal [2005]. Fung et al. [2007] proposed Top-Down Specialization (TDS), a heuristic approach by which a relatively accurate classifier can be built based on the anonymized data. LeFevre et al. [2006; 2008] proposed *Mondrian*, which considers the anonymity problem for classification. *Mondrian* employs multidimensional generalization to achieve better data utility and, hence, improves classification accuracy. All the aforementioned works use a k-anonymity-based privacy model achieved by means of single-dimensional generalization. On the other hand, *Mondrian* achieves anonymity by transforming the data in a multidimensional manner. In Section 5, we will experimentally compare our differentially-private multidimensional algorithm, called *DiffMulti*, with *Mondrian*.

Several methods have been proposed for publishing a private contingency table [Barak et al. 2007; Ding et al. 2011; Xiao et al. 2011b; Cormode et al. 2012; Qardaji et al. 2013b]. Barak et al. [2007] used linear programming to post-process a differentially private output in order to publish a set of consistently integral marginals of a contingency table. Though it guarantees differential privacy, Barak et al.'s work does not improve accuracy in the output data. A similar problem has also been studied by Ding et al. [2011] and Qardaji et al. [2013b]. Xiao et al. [2011b] succeeded in improving the accuracy of a differentially-private contingency table by proposing *Privelet*, a method based on wavelet transformation on the data attributes. On the same note, Cormode et al. [2012] proposed to optimize the computation required when publishing a contingency table of a sparse data set, in a differentially-private way. They achieved that by utilizing compact summaries computed *directly* from the input data set, as opposed to computing a noisy contingency table first, which is costly for sparse data sets. However, the utility of their private summaries is similar to that of a generated contingency table. We argue that releasing a private contingency table can be damaging to the accuracy of the analysis as the added noise grows larger for sparse data sets. To account for this shortcoming, our method releases a generalized version of the input data set; thus, increasing the record counts by allowing more records to be semantically grouped together. We will experimentally elaborate on this point in Section 5.

Qardaji et al. [2014] proposed *PriView* to enhance releasing marginal contingency tables for high-dimensional *binary* data sets for the *general purpose* of answering count queries. *PriView* utilizes covering design to carefully select certain low-dimensional *views* (sets of attributes), and then reconstructs k-way marginals from such views. Even though *PriView* can be extended to handle categorical data sets, as suggested by the authors, it fails to process data sets with numerical attributes. The work in [Qardaji et al. 2013a] studies how to effectively partition *2-dimensional* GPS-like data points, and proposes an adaptive-grid method that significantly improves data partitioning over state-of-the-arts. Our proposed method shares the same partitioning intuition with the adaptive-grid method in [Qardaji et al. 2013a]: we need to impose fine-grained partitioning on dense regions and coarse-grained partitioning on sparse regions. In an attempt to estimate the joint distribution of high-dimensional *numerical* data, Li et al. [2014] leveraged copula functions and proposed *DPCopula*. Copula functions require attributes to have large domains, and fail otherwise. Therefore, *DPCopula* treats categorical attributes with large domains (e.g., more than 10 domain values [Li et al. 2014]) as continuous. Unlike the above techniques, we propose *DiffMulti* that is capable of handling both categorical and numerical attributes with no restriction on the domain size of any attribute. Moreover, *DiffMulti* incorporates different partitioning strategies that improve data utility, depending on the target workload of the published data.

Our work can be related to those which focus on releasing private histograms [Chawla et al. 2005; Hay et al. 2010; Li et al. 2010; Xiao et al. 2014]. A histogram is a set of disjoint regions containing data points over the domain of a data set. Several works have been proposed to release private histograms. Chawla et al. [2005] presented a theoretical work by which they proposed a recursive approach for releasing private histograms in the multidimensional space. However, their privacy constraint is based on a syntactic privacy model. Hay et al. [2010] proposed a method for releasing more accurate private histograms under differential privacy; however, their work is limited to single-dimensional histograms as in [Blum et al. 2008; Xu et al. 2013]. Although [Hay et al. 2010] and [Xiao et al. 2011b] provide noise optimization for *range* queries, Li et al. [2010] enhanced their work by achieving optimal noise variance for a variety of *workload* queries. Xiao et al. [2014] proposed a method by which multidimensional partitioning was used to release differentially-private histograms.

A closely related method to our work is *DPCube* [Xiao et al. 2014], which publishes private histograms for random workloads. Even though DPCube publishes histograms, it is possible to synthesize a noisy contingency table of the underlying data set D by issuing a set of queries spanning all combinations of domain values. We point out three major differences between our proposed method *DiffMulti* and *DPCube*. First, while it is possible to synthesize a noisy contingency table using *DPCube*, if the input data set D is sparse or high-dimensional with a large domain, then performing operations on the noisy contingency table results in poor data utility. This is because sparse data points tend to be scattered over the domain space resulting in extremely small counts in most subdomains. Thus, raw counts will be outweighed by the added noise, rendering a query answer useless. This notion will be empirically demonstrated in Section 5. Second, *DPCube* handles nominal or discretized attributes, whereas *DiffMulti* is capable of handling both nominal and numerical attributes. Third, DPCube, along with all the techniques in [Hay et al. 2010; Xiao et al. 2011b; Li et al. 2010; Xiao et al. 2011a], is based on the *interactive* model that requires the queries to be provided in advance. In the case of multiple users querying D, ϵ will be distributed among those users [Xiao et al. 2014]. If a user is assigned a small fraction of ϵ, more noise will be added to the true query answer. Once all ϵ is consumed by the queries, the data holder has to shut down the database entirely. In contrast, our *non-interactive* method releases differentially-private data, giving the data recipient much more flexibility in performing various analysis with no limitation on data access. We experimentally compare *DiffMulti* to *DPCube* in Section 5.

To account for the inherent challenge of releasing sparse or high-dimensional data in a differentially private way, Zhang et al. [2014] proposed a method called *PrivBayes*. In *PrivBayes*, a Bayesian network is utilized to construct a set of low-dimensional subcubes that approximate the joint distribution in order to release a synthetic data set which in turn approximates the distribution of the high-dimensional input data set. We will compare with *PrivBayes* in terms of classification accuracy in Section 5.

In a closely related research area, the problem of differentially-private *release* of relational data has been studied in [Mohammed et al. 2011; Qardaji and Li 2012]. The general idea in both works is to partition the input data set into smaller groups of "similar" records, then release a noisy count of the records in each resultant partition/region. Mohammed et al. [2011] proposed an algorithm called *DiffGen*, a top-down specialization approach that aims at producing a generalized version of the input data in a differentially-private setting. *DiffGen* uses a single-dimensional partitioning strategy that greedily chooses a split attribute that maximizes the utility of the output data set, without violating differential privacy. When an attribute is chosen, a split is performed in accordance with the hierarchy of the domain values dictated by an input taxonomy tree. Our work leverages multidimensional partitioning to improve data utility while adhering to the rigorous requirement of differential privacy. Mohammed et al. improved classification accuracy when they experimentally compared with *DiffPC4.5* [Friedman and Schuster 2010], an interactive approach for classification analysis. In Section 5, we will compare the performance of our proposed method with *DiffGen* in terms of runtime, classification accuracy, and other utility metrics.

In [Qardaji and Li 2012], the authors also assumed a non-interactive setting and proposed a *general framework* for releasing relational data under differential privacy. They proposed a meta-algorithm, called *RPS*, that takes into consideration the distribution of the data points (records) in the multidimensional space R of the data set. The meta-algorithm recursively performs *binary* partitioning over R to achieve nearly balanced regions. A median point is computed in a differentially-private way from the domain of the chosen attribute to produce two non-overlapping regions. Noisy counts of data points in each resultant region are then returned to compose the overall san-

itized data set. *RPS* has two weaknesses. First, for the case of relational data, *RPS* randomly chooses a split attribute [Qardaji and Li 2012]. On the other hand, we propose a greedy algorithm that performs a series of iterations. In a single iteration, our method carefully chooses a split value from a set containing at most d values, where d is the total number of attributes. Moreover, the chosen attribute is picked with the goal of minimizing information loss. As a result, carefully choosing a split value is advantageous (in terms of data utility) over the random splitting strategy of *RPS*, which has a probability of $1/domain\_size$ in finding the best split value. Our experimental evaluation in Section 5 further validates this claim. Second, even though *RPS* is theoretically capable of choosing a split point that spans across multiple dimensions (attributes), it does not scale for data sets with a large number of dimensions because it inefficiently considers all combinations of values across all dimensions [Qardaji et al. 2014]. More specifically, in every iteration, *RPS* chooses one split point from $\Omega(A_1) \times \ldots \times \Omega(A_d)$ possible combinations, while our method significantly decreases the number of computations by selecting a split point from a maximum of d points in the d-dimensional space, where d is the number of attributes and $\Omega(A_i)$ is the domain of attribute A_i in a given region. In summary, *RPS* is a general framework that can be instantiated differently based on the desired implementation, whereas our proposed algorithm is a specific approach designed for workload-aware, differentially-private, multidimensional relational data release. In Section 5, we report the utility measure resulting from both *RPS* and our method.

3. BACKGROUND

3.1. Differential Privacy

Differential privacy [Dwork 2006] is a rigorous privacy model that works independently of any adversary's background knowledge about a target victim. This privacy model guarantees that a differentially-private data set $\hat{D}$ can be obtained from two input data sets that differ in size by one record. Let D and D' be two *neighboring* input data sets such that both data sets differ by at most one record, denoted by $|D \triangle D'| \leq 1$. A differentially-private mechanism will yield an output data set $\hat{D}$ with the same probability from both D and D'. An adversary will not be able to acquire extra knowledge about a target victim because the output data set could have been obtained from a neighboring data set that does not contain the victim's record. In other words, the analysis of $\hat{D}$ is insensitive to the addition/removal of a single record in the input data set.

DEFINITION 3.1 (ϵ-*differential privacy*). *A randomized algorithm Ag gives ϵ-differential privacy if for any neighboring data sets D and D' differing by at most one record, and for any possible output data set $\hat{D}$,*

$$\boldsymbol{Pr}[Ag(D) = \hat{D}] \leq \boldsymbol{exp}(\epsilon) \times \boldsymbol{Pr}[Ag(D') = \hat{D}], \tag{1}$$

where the probability is taken over the randomness of the Ag. ■

The parameter ϵ is an input parameter that is specified by the data holder. ϵ is usually of a small value, typically $0 < \epsilon \leq 1$ [Dwork 2006; Friedman and Schuster 2010; Dwork 2011]. ϵ specifies the degree of desired privacy in the output data set, and thus is called the *privacy budget*. This implies that higher values of ϵ will incur higher privacy costs, resulting in lower privacy protection and higher data utility. Vice versa, lower values of ϵ result in higher privacy at the expense of lower data utility in the output data set. In Section 5, we report how different values of ϵ affect classification accuracy. For more theoretical analysis, we refer the reader to [Dwork and Roth 2014].

To satisfy ϵ-differential privacy, an algorithm must be insensitive to the underlying data. To offset the effect of neighboring data sets, i.e., the existence or absence of a single record, two methods were proposed in the literature: the *Laplace mechanism* and the *exponential mechanism*. Both mechanisms rely on the privacy parameter ϵ and the *sensitivity* of a function that maps the input data set to real values. The sensitivity of a function is the maximum amount inflicted on the output due to adding or removing a single record in the input data set.

DEFINITION 3.2 (*Sensitivity*). *For any function $f : D \rightarrow \mathbb{R}^d$, the sensitivity of f is*

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

for all D, D' differing at most by one record. ■

Suppose function f answers count queries over a data set D. Given any neighboring data set D', any answer from f would differ by at most 1; therefore, the sensitivity of f is 1.

Laplace Mechanism. To satisfy ϵ-differential privacy when the output is a real value, a function f should return a noisy answer that would mask the effect of a neighboring data set. In this context, Dwork et al. [2006] proposed the Laplace mechanism, which first computes the true output of a function and then adds a noise drawn from the Laplace distribution. The noise is calibrated based on the privacy parameter ϵ and the sensitivity of the function Δf. Formally, the Laplace mechanism takes as inputs a data set D, the privacy parameter ϵ, and a function f, and outputs $\hat{f(D)} = f(D) + \texttt{Lap}(\lambda)$, where $\texttt{Lap}(\lambda)$ is a noise drawn from the Laplace distribution with probability density function $\Pr(x|\lambda) = \frac{1}{2\lambda}\texttt{exp}(-|x|/\lambda)$ of variance $2\lambda^2$ and mean 0.

THEOREM 3.1. *[Dwork et al. 2006]* Given any function $f : D \rightarrow \mathbb{R}^d$ over an arbitrary domain of data set D with d attributes, an algorithm Ag that adds independently generated noise with distribution $\texttt{Lap}(\Delta f/\epsilon)$ to each of the d outputs satisfies ϵ-differential privacy. ■

As an example, a function f that answers count queries would have a sensitivity $\Delta f = 1$. According to Theorem 3.1, $\hat{f(D)} = f(D) + \texttt{Lap}(1/\epsilon)$ is ϵ-differentially private.

Exponential Mechanism. In certain cases, it does not make sense to add noise to the output of a function when the output is not real. Let $u : (D \times \mathcal{T}) \rightarrow \mathbb{R}$ be a utility function that assigns a real-valued score to every output $t \in \mathcal{T}$. The purpose is to select an output with the highest score, as higher scores imply better utility. McSherry and Talwar [2007] proposed the exponential mechanism that takes a data set D, the privacy parameter ϵ, an output range $\mathcal{T}$, and a utility function u, and selects an output $t \in \mathcal{T}$ that is close to the optimum output with respect to u. The exponential mechanism maintains a probability distribution over the range of outputs $\mathcal{T}$. The likelihood of selection grows exponentially for higher scores.

THEOREM 3.2. *[McSherry and Talwar 2007]* Given any utility function $u : (D \times \mathcal{T}) \rightarrow \mathbb{R}$ with sensitivity $\Delta u = \max_{\forall t, D, D'} |u(D,t) - u(D',t)|$, an algorithm Ag that chooses an output t with probability proportional to $\texttt{exp}(\frac{\epsilon u(D,t)}{2\Delta u})$ satisfies ϵ-differential privacy. ■

Composition. Differential privacy enjoys two composition properties. The first property is called *sequential composition* and it applies to the case where a sequence of computations is performed on a *single* data set. In this scenario, if every computation

guarantees differential privacy, then the entire sequence gives the accumulated privacy guarantee. The second property is called *parallel composition* and it applies to situations where a sequence of computations is performed on *disjoint* data sets. In this scenario, the sequence gives differential privacy with the worst privacy guarantee, i.e., the highest privacy budget, among all computations.

LEMMA 3.1 (*Sequential composition* [MCSHERRY 2009]). Let each computation Ag_i provide ϵ_i-differential privacy. A sequence of $Ag_i(D)$ over the data set D provides $(\sum_i \epsilon_i)$-differential privacy. ■

LEMMA 3.2 (*Parallel composition* [MCSHERRY 2009]). Let each computation Ag_i provide ϵ-differential privacy. A sequence of $Ag_i(D_i)$ over a set of disjoint data sets D_i provides ϵ-differential privacy. ■

3.2. Generalization

Generalization is the act of replacing raw values in the input data set D with more general and less semantically specific values. Such general values are specified by a pre-defined generalization hierarchy, e.g., the taxonomy trees shown in Fig. 2. Let D, an input data table, be defined over a set of attributes $\mathcal{A} = \{A_1, \ldots, A_d\}$, where $\Omega(A_i)$ represents the domain of attribute A_i. *Single-dimensional generalization* is defined by a function $\phi_i : v \rightarrow p$ for *each* attribute $A_i \in \mathcal{A}$, where p is a value in the generalization hierarchy. As a result, if a value v is chosen for generalization, all instances of v in D will be generalized as well.

To preserve more information, we employ *multidimensional generalization*. In this setting, a *vector of values* is considered for generalization instead of considering a single value at a time. The idea is to divide the d-dimensional domain space of D into non-overlapping generalization regions. Every region R_i contains a set of generalized records from D, where every raw record is uniquely mapped to its corresponding region. By that intuition, a region can be considered an equivalence group because every region contains a disjoint subset of generalized records from D.

DEFINITION 3.3 (*Multidimensional generalization*). *Given a raw data set D defined over a set of attributes $\mathcal{A} = \{A_1, \ldots, A_d\}$, and some user-defined taxonomy trees, multidimensional generalization is defined by a* single *global function $\phi : \Omega(A_1) \times \ldots \times \Omega(A_d) \rightarrow \{R_1, R_2, \ldots, R_m\}$ that maps an entire record in D to its corresponding generalization region R_i, where the d-dimensional domain space of D is divided into disjoint generalization regions $R_1, R_2, \ldots, R_m$.* ■

According to the above definition, generalizing a raw value v in a given record entails considering the entire vector of raw values in that record. In other words, given a group of records that share a raw value v on attribute A_i, the combination of raw values on every other attribute A_j, where $i \neq j$, in a given record will determine the value p to which v will be generalized.

EXAMPLE 3.1. *Consider Table I and its anonymized version under multidimensional generalization, as spatially represented in Fig. 3 (c). $North\_America$ in records $\{North\_America, 1947\}$ and $\{North\_America, 1968\}$ has been generalized to $Any\_Continent$, whereas $North\_America$ in record $\{North\_America, 1955\}$ has been generalized to New_World.* ■

Given any region R_i, each dimension represents a generalized value p of the underlying subdomain. We propose an algorithm that is capable of handling both categorical and numerical attributes. For categorical attributes, p is drawn from the generalization hierarchy as specified by the taxonomy trees. For numerical attributes, on the other hand, the algorithm will adaptively select a binary split point by which a division on

the continuous domain will result in two disjoint intervals. The process of generating generalization regions is called *partitioning*. A good partitioning strategy is essential for improving data utility in the output data set.

For an algorithm to satisfy ϵ-differential privacy, all operations must be insensitive to the underlying data. In other words, given two neighboring data sets D and D', where $|D \triangle D'| \leq 1$, the algorithm must bound the probability of obtaining the same output data set in accordance with Definition 3.1. At the same time, the algorithm must find a proper generalization function $\phi : \Omega(A_1) \times \ldots \times \Omega(A_d) \rightarrow D^d$ depending on the expected workload of the published data. Fixing ϕ will make the algorithm satisfy the ϵ-differential privacy requirement; however, the output data will not be tailored to the expected analysis[2]. As a result, the data utility will drop significantly, rendering any analysis on the output data useless. In Section 4, we present our algorithm for performing effective partitioning to maximize data utility without violating the ϵ-differential privacy requirement.

3.3. Problem Statement

We informally describe our problem as follows: A data holder is in possession of a data set D that contains a multiset of records, where each record belongs to a unique individual. All person-specific information, such as YoB and SSN, have been removed. D is defined over a set of attributes $\mathcal{A} = \{A_1, \ldots, A_d\}$ that can also be found in a non-sanitized, publicly available data set containing a group of records that belong to the same individuals whose records are in D. In addition, D contains a *Class* attribute A^{cls} that is used for classification analysis. We assume that A_i is either categorical or numerical, and A^{cls} is categorical. Finally, we assume that for each categorical attribute $A_i \in \mathcal{A}$, a taxonomy tree is provided that defines the hierarchy of values in $\Omega(A_i)$. We do not require a taxonomy tree for numerical attributes as it requires an extensive preprocessing of D to determine appropriate splitting values. Rather, our proposed algorithm performs this task adaptively upon runtime.

A data holder wishes to publish an ϵ-differentially-private version of D for either general data analysis or specific analytical tasks, such as classification analysis or cluster analysis.

DEFINITION 3.4 (ϵ-*differentially-private multidimensional generalization*). *Given a raw data set D, a set of taxonomy trees, and a privacy budget ϵ, we wish to produce an ϵ-differentially-private version $\hat{D}$ of D by means of multidimensional generalization in order to improve data utility for more accurate analysis of $\hat{D}$.* ■

In the next section, we propose a heuristic based on a greedy recursive splitting of domain space.

4. ANONYMIZATION ALGORITHM

We present *DiffMulti*, a *diff*erentially-private algorithm that employs *multi*dimensional generalization for relational data release. Our proposed algorithm operates in two phases, as illustrated in Fig. 1. We provide an overview, then proceed to discuss the major operations in each phase. Namely, Sections 4.2 and 4.3 detail Phase 1, and Section 4.4 details Phase 2. Our algorithm is outlined in Section 4.5, followed by a discussion.

[2]Unless performed randomly, having a fixed generalization function ϕ is a non-trivial task. The domain space of ϕ is as large as the cardinality of the input data set. Moreover, the codomain of ϕ is a set of d-dimensional regions, each bounded by either an interval or a value from the generalization hierarchy. Our proposed algorithm effectively partitions the regions to maintain data utility.

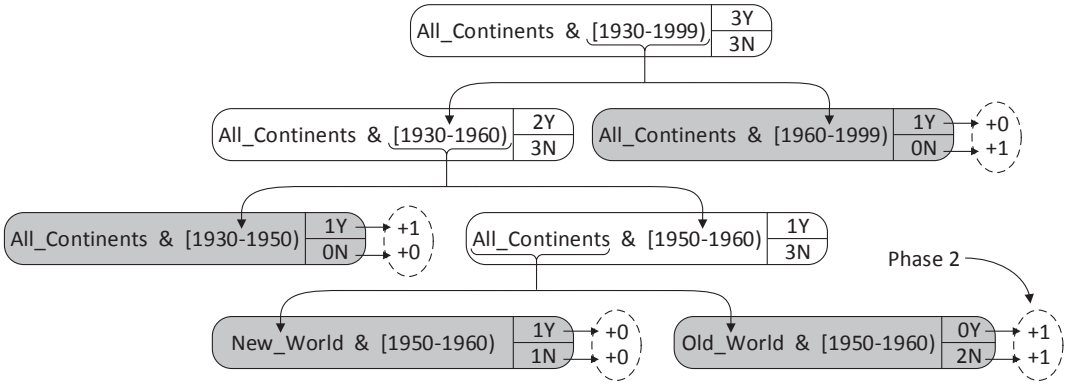


Fig. 4. Raw data records structured as tree of partitions

4.1. Overview

We propose an anonymization algorithm that operates in two phases. In *Phase 1*, all the records of the input data set D are generalized over the set of attributes $\mathcal{A} = \{A_1, \ldots, A_d\}$ in a differentially private manner. When a group of records is generalized to the same set of values, they form a unique *equivalence group*. Then, Phase 1 recursively specializes every equivalence group. In *Phase 2*, our algorithm publishes noisy record counts pertaining to the final equivalence groups, where records are least generalized. A straightforward way to implement this method is by scanning the entire raw records to determine a split attribute, then scanning the entire records again to specialize and split them into equivalence groups. We note that in order to specialize the records of an equivalence group, it is sufficient to scan those records apart from the rest of the data set records. Performing specialization by applying the straightforward solution gives rise to the scalability issue, wherein scanning the entire data records becomes exhaustive for very large data sets. We utilize an efficient tree data structure that provides data access on the granular level, as illustrated in Fig. 4. A node in this tree is referred to as a *partition*. For every value in the *Class* attribute A^{cls}, a partition P_i maintains a generalized record and a pertinent count that refers to the number of raw records generalized to the same attribute values and that share the same *Class* value. Record scanning is, thus, confined to the size of a single partition, $|P_i| = \sum_{x=1}^{|\Omega(A^{cls})|} count_x$.

DiffMulti, detailed in Algorithm 1, performs a sequence of *specializations* on the data records, as follows: first, the algorithm creates a root partition, P_i, that contains all raw records generalized to the topmost values in the domain hierarchy on every attribute in $\mathcal{A}$. At this point, publishing a Laplacian-noisy version of $|P_i|$ guarantees ϵ-differential privacy; however, the data records are at a high level of abstraction, and analysis on such general data is far from accurate. We reduce data generality by *specializing* P_i. The maximum number of specializations is a user-input parameter to our algorithm, h. When a specialization takes place, denoted by $v \rightarrow child(v)$, a general value v gets replaced by its less general child values $child(v)$ from the domain hierarchy. In our tree data structure, every specialization creates new disjoint child partitions, each holding a set of records that generalize to the same values. A child partition $P_{c_j} \in \{P_{c_1}, \ldots, P_{c_\gamma}\}$ represents a distinct region R in the domain space of D. Unlike single-dimensional partitioning, which performs specializations globally among all data records, our method employs multidimensional partitioning by which only a certain set of records are affected by $v \rightarrow child(v)$. In other words, a single-

dimensional partitioning strategy splits across all regions in the domain space of D, while our method performs a split $v \rightarrow child(v)$ only within one region. This step is advantageous for data utility because at the end of Phase 1, less generalization is imposed on the raw records.

EXAMPLE 4.1. *We continue from Example 1.1. The root partition in Fig. 4 contains all the data records in Table I generalized to the topmost values, i.e.,* $Any\_Continent$ *and* $[1930 - 1999)$*, in addition to the true counts of records having the same value on the* Class *attribute. Given the root partition, the set of candidate values for specialization is* $\{Any\_Continent, [1930 - 1999)\}$*. A specialization is performed on the value that results in the least amount of information loss. For simplicity, let us assume that the specialization* $[1930 - 1999) \rightarrow \{[1930 - 1960), [1960 - 1999)\}$ *retains more data utility than* $All\_Continents \rightarrow \{New\_World, Old\_World\}$*. Consequently, the root partition is specialized on the* YoB *attribute, where a child partition is created for every child value of* $[1930 - 1999)$*.* DiffMulti *iteratively specializes every new partition in the tree until a leaf partition is reached, where records are published according to noisy versions of their true counts. Leaf partitions are colored in grey for ease of presentation.* ■

4.2. Choosing a Candidate

As mentioned in Section 4.1, a specialization $v \rightarrow child(v)$ replaces v with its child values from the domain hierarchy associated with an attribute A, where $v \in \Omega(A)$. Our algorithm greedily chooses a value v from a set of candidate values in a given partition P_i, where each value belongs to a distinct attribute. The set of candidates includes only the values in the generalized record that represents the partition. Hence, a single specialization requires choosing a value from a maximum of d candidates, where d is the number of attributes in the raw data set D. Selecting a value for specializing a partition is based on a real-valued score computed by a greedy utility function, which varies depending on the target workload. For example, the root partition in Fig. 4 contains the candidate set $\{Any\_Continent, [1930-1999)\}$, where $[1930-1999)$ has been chosen for specialization.

Depending on the purpose of the data release, the chosen candidate should have either the highest or the lowest score. Scores give the sense of how much utility (or loss) the data set has maintained (or suffered) due to specializing a candidate. For example if the released data is intended for classification analysis, then `Max` would be the utility function, and the "best" candidate for specialization is the one with the highest score. If the data holder is to design their own utility function, we suggest the following guideline. The key principle to designing a good utility function is to design a function that is biased towards choosing a candidate value that results in evenly distributing data records among child partitions - to avoid the case where some partitions end up with a very small number of records. This is because partitions with a small number of records have distorted noisy counts, where the added Laplace noise becomes more dominant upon data publishing (Phase 2), resulting in inaccurate data analysis. We would like to note that in the case of incorporating a bad utility function (i.e., one that is not biased towards evenly distributing records), our proposed method accommodates such behavior by employing a *fair* distribution strategy, as explained later in this section.

Given a partition for specialization, *DiffMulti* computes the scores of the candidates in the partition. Then, *DiffMulti* makes use of the exponential mechanism, presented in Theorem 3.2, to select a candidate. The exponential mechanism exponentially favors higher scores while maintaining privacy. Scores are inverted for the exponential mechanism in the cases where lower scores are more favorable. Herein, we present three utility functions; the *discernibility metric* and the *Normalized Certainty Penalty*

are two general-purpose metrics for when the target workload of the released data is unknown, and the third utility metric is Max for a specific data analysis task, namely, classification analysis.

The first utility function is the *discernibility metric* (DM), which is a general-purpose quality metric that measures the amount of data loss the raw data set has suffered due to anonymization [Bayardo and Agrawal 2005]. Every record r in the anonymized data set is assigned a penalty value equal to the size of the group of records that are indistinguishable from r. More specific to our method, a generalized record in a partition P_i is given a numeric penalty equal to $|P_i|$. Given a candidate v, $\texttt{DM}(D, v)$ is computed as follows:

$$\texttt{DM}(D, v) = \sum_{P_c \in child(P_v)} (|P_c|^2), \tag{3}$$

where P_v is v's partition, and $child(P_v)$ is the set of P_v's child partitions. Lower values of $\texttt{DM}(D, v)$ imply that specializing on v inflicts less anonymization distortion compared to other candidates. The discernibility penalty of the entire output data set $\hat{D}$ can be computed as follows:

$$\texttt{DM}(\hat{D}) = \sum_{\forall P_i in \hat{D}} (|P_i|^2). \tag{4}$$

The sensitivity $\Delta\texttt{DM}(D, v) = 2|D|+1$, where $|D|$ denotes the number of records in D. For simplicity, let us consider finding $\Delta\texttt{DM} = \max_{D,D'} ||\texttt{DM}(D) - \texttt{DM}(D')||_1$, where D and D' are two data sets differing by 1 record. Assume that the output data set contains one partition. The square operation has a greater impact on a data set D' s.t. $|D'| = |D|+1$ (as opposed to $|D| - 1$). Thus, $\Delta\texttt{DM} = |D'|^2 - |D|^2 = 2|D| + 1$. The same reasoning can be applied to find $\Delta\texttt{DM}(D, v)$.

The second utility function is the *Normalized Certainty Penalty* (NCP), described in [Xu et al. 2006], which measures information loss due to generalization. Thus, lower values imply better data utility. Given a generalization hierarchy represented as a taxonomy tree, let e denote a node in the taxonomy tree and let $\mathcal{L}$ denote the set of leaf nodes, where a leaf node represents a raw value. If a raw value l is generalized to node e_l in the taxonomy tree, then the NCP value of l is computed as follows:

$$\texttt{NCP}(l) = \begin{cases} 0, & e_l \text{ is leaf} \\ |e_l|/|\mathcal{L}|, & \text{otherwise,} \end{cases} \tag{5}$$

where $|e_l|$ is the number of leaf nodes of the subtree of the taxonomy tree rooted at e_l. For a numerical attribute A_{num}, $|e_l|$ would be the range of the generalization interval and $|\mathcal{L}|$ is the domain range of A_{num}. Hence, $0 \leq \texttt{NCP}(i) \leq 1$. Given a candidate v, $\texttt{NCP}(D, v)$ is computed as follows:

$$\texttt{NCP}(D, v) = \sum_{c \in child(v)} (supp_c \cdot \texttt{NCP}(D, c)), \tag{6}$$

where $supp_c$ is the number of records that include a raw value generalized to c in the taxonomy tree. To find the sensitivity of $\texttt{NCP}(D, v)$ in Equation 6, let $\texttt{NCP}(D, c) = 1$. $supp_c$ can change at most by 1. Therefore, $\Delta\texttt{NCP}(D, v) = 1$. Lastly, the NCP of a generalized data set $\hat{D}$ can be measured as follows:

$$\texttt{NCP}(\hat{D}) = \frac{\sum_{\forall i \in \hat{D}} (supp_i \cdot \texttt{NCP}(i))}{\sum_{\forall i \in \hat{D}} (supp_i)}. \tag{7}$$

Finally, our third utility function is Max, and it is used to publish an anonymized data set for classification analysis. Given a candidate v, $\texttt{Max}(D, v)$ computes the score of v by summing the highest class frequencies of v's child values. In other words, if v's partition were to be specialized, the score of v would be equal to the result of adding

the largest count in every child partition. $\texttt{Max}(D, v)$ helps to build a better classifier by going for the favor of choosing partitions with purest class frequencies. Intuitively, higher scores are more desirable.

$$\texttt{Max}(D, v) = \sum_{c \in child(v)} (\max_{cls}(|D_c^{cls}|)). \tag{8}$$

The absence or addition of 1 record in D would change $\texttt{Max}(D, v)$ by at most 1. Therefore, the sensitivity of $\texttt{Max}(D, v)$ is 1.

Choosing a value with the best score (i.e., highest or lowest, depending on the utility function) does not satisfy differential privacy because the process is data dependent. Therefore, given a utility function u and a set of candidates $Cand_i$ from partition P_i, our algorithm utilizes the exponential mechanism to choose a candidate $v_i \in Cand_i$. This step is outlined in Line 11 of Algorithm 1.

THEOREM 4.1. Choosing a candidate value for specialization satisfies ϵ'-differential privacy. ■

PROOF 4.1. *Let $Cand_i$ be the set of candidate values from which a single value is to be chosen for specialization. Our algorithm selects a value $v_i \in Cand_i$ with the following probability:*

$$\frac{exp(\frac{\epsilon'}{2\Delta u} u(D, v_i))}{\sum_{v \in Cand_i} exp(\frac{\epsilon'}{2\Delta u} u(D, v))}, \tag{9}$$

where $u(D, v_i)$ is a score computed from a utility function, and Δu is the sensitivity of the utility function u. According to Theorem 3.2, selecting a value with probability proportional to $exp(\frac{\epsilon' u(D,t)}{2\Delta u})$ satisfies ϵ'-differential privacy. ■

Allocating the Number of Specializations. One of the input parameters of Algorithm 1 is a positive integer that represents the maximum number of specializations, $h \in \mathbb{N}$. Specializing a single partition, i.e., distributing the records of the specialized partition among its newly created child partitions, decreases h by 1. The algorithm starts with one partition that includes the entire data records generalized to the topmost level, followed by a series of specializations on each of the child partitions. This process gives rise to the following question: *What is the maximum number of specializations that can be performed on each child partition?* Differently put, how can the unused portion of h be distributed among the child partitions? For ease of presentation, let $P_i.h' \in \mathbb{N}$ refer to the maximum number of specializations that can be performed starting at partition P_i, where $0 \leq h' < h$. A straightforward solution would be to evenly distribute the remaining of h among the child partitions, as demonstrated by the following example.

EXAMPLE 4.2. *Let $h = 21$. Suppose partition P_1 is specialized and two child partitions, P_2 and P_3, are created. Initially, $P_i.h' = h = 21$. $P_1 \rightarrow \{P_2, P_3\}$ consumes 1 specialization from h. As a result, $P_2.h' = P_3.h' = (21 - 1)/2 = 10$. Therefore, every child partition gets to be iteratively specialized 10 times at most.* ■

The above method is simple and does not violate differential privacy because the sensitivity of evenly distributing $h' - 1$ among child partitions is 0. Having said that, recall that every partition represents a region in the multidimensional space of the input data set D. It is unlikely that the data points (raw records) are evenly distributed throughout the domain space. Indeed, some regions are denser than others.

EXAMPLE 4.3. *In Example 4.2, we showed that if the remainder of h was evenly distributed among the child partitions, then $P_2.h' = P_3.h' = 10$. Assume that $|P_1| = 10$,*

$|P_2| = 8$, *and* $|P_3| = 2$, *where* $|P_i|$ *is the number of records in* P_i. *This means that a set of 2 records and a set of 8 records will be specialized equally.* ■

Even distribution of the number of specializations does not take into consideration dense regions in the domain space. Shifting specializations towards dense regions has a positive impact on data utility because more records get specialized, hence producing less abstract data. We make a contribution in this work by imposing a *fair* distribution of the number of specializations rather than an even one. Let $P_i \rightarrow \{P_{c_1}, P_{c_2}, \ldots, P_{c_\gamma}\}$, the number of specializations assigned to a child partition P_{c_i} is proportional to its size. Let $P_{c_i}.h' \in \mathbb{N}$,

$$P_{c_i}.h' = \begin{cases} 0, & |P_{c_i}| = 0 \\ \lfloor \frac{|P_{c_i}|}{|P_i|} \cdot (P_i.h' - 1) \rfloor, & \text{otherwise.} \end{cases} \tag{10}$$

EXAMPLE 4.4. *Applying the fair distribution strategy to Example 4.2, we get* $P_2.h' = (8/10) \cdot (21 - 1) = 16$ *and* $P_3.h' = (2/10) \cdot (21 - 1) = 4$. *Hence, the 8 records in* P_2 *will be specialized 4 times more than the 2 records in* P_3, *allowing to retain more data utility. Moreover, under even distribution, the dense region, represented by partition* P_2, *is specialized 10 times (Example 4.2); whereas under fair distribution, the dense region suffers less data loss because it is specialized 16 times.* ■

Equation 10 does not respect the indeterministic nature of differential privacy. This is because, given a partition, P_x, Equation 10 calculates the exact record count in $|P_x|$, which may change for a neighboring input data set D'. Therefore, we use the Laplace mechanism to return noisy record counts, instead. This step is outlined in Line 13 of Algorithm 1.

THEOREM 4.2. Fair distribution of the number of specializations satisfies ϵ'-differential privacy. ■

PROOF 4.2. *Given 1 as the sensitivity of a count query, a privacy budget* ϵ', *and a true record count* $|P_x|$, *a differentially-private version of Equation 10 calculates* $|P_x| + Lap(1/\epsilon')$, *for* $|P_x| \neq 0$. *According to Theorem 3.1, adding independently generated noise from the Laplace distribution* $Lap(1/\epsilon')$ *to a true query count satisfies* ϵ'*-differential privacy.* ■

If the calculated value of $P_{c_i}.h'$ contains a fraction, then the value is rounded down to the nearest non-negative integer. If the summation of all fractions among the child partitions is ≥ 1, then this remainder can be randomly distributed among the child partitions to avoid losing portions of h. Note that this post-processing step conforms with differential privacy because this step is a public rule and is not dependent on the input data set [Kifer and Lin 2010].

4.3. Determining a Numerical Split Point

Given a partition and its set of candidates, where each candidate has a real-valued score, a specialization is performed on the value chosen by the exponential mechanism. We refer to this value as the *split value*. A specialization $v \rightarrow child(v)$ replaces the split value v with its child values. We perform a specialization as follows: in a partition P_i, once a value $v \in P_i$ is chosen for specialization, our algorithm creates a child partition P_{c_j} for every child value $c_j \in child(v)$. Then, it distributes the records in P_i among the child partitions where each record goes to its pertinent child partition. A score is computed for the new values in every child partition. Finally, the parent partition P_i is deleted as it contains no further useful information to the specialization process. We

note that the raw values of the records are maintained throughout runtime in order to properly split the records among child partitions.

To satisfy differential privacy, any operation should be independent of the underlying data. We project this rule on finding the split value for both categorical and numerical attributes. We note that a *categorical split value* is a generalized value drawn from a pre-defined generalization hierarchy (taxonomy tree), and a *numerical split value* is an interval over the continuous domain. If the exponential mechanism chooses a categorical split value, then adding or removing a record from the input data set will not affect how a split value is determined. As a result, having a taxonomy tree induces sensitivity $= 0$ on determining a categorical split value. On the other hand, if the exponential mechanism chooses a numerical split value, then a numerical *split point* has to be found from the chosen interval. Due to the inconvenience of defining a taxonomy tree for numerical attributes, our proposed algorithm adaptively determines a numerical split point that divides an interval over the attribute domain into two disjoint and successive subintervals. A simple solution to finding a split point can be to pick a random point from the raw records and split the records into two child partition accordingly. This solution suffers from two shortcomings. First, the chosen point may not exist in a neighboring data set that differs by 1 record. Hence, choosing a point directly from the raw data records violates differential privacy. Second, this solution does not pay attention to the utility of the partitioned data. In the following, we present a method for carefully choosing a numerical split point that maximizes data utility without violating differential privacy.

We compute a score for every value v_i in the domain of the numerical attribute $\Omega(A_{num})$, then use the exponential mechanism to choose a value $v_i \in \Omega(A_{num})$ with the following probability:

$$\frac{\exp(\frac{\epsilon'}{2\Delta u}u(D, v_i))}{\int_{v\in\Omega(A_{num})} \exp(\frac{\epsilon'}{2\Delta u}u(D, v))\, dv}, \tag{11}$$

where $u(D, v_i)$ is a score computed from a utility function u with sensitivity Δu.

Computing a score for every value in the domain can be exhaustive. We observe that intervals of consecutive values along the attribute domain can have the same score. More formally,

OBSERVATION 4.1. Given a utility function u and a numerical attribute A_{num} with domain range $\Omega(A_{num}) = \{I_1, I_2, \ldots, I_k\}$, where I_n is an interval of consecutive values. Then, $\forall I_n \in \{I_1, I_2, \ldots, I_k\}$, we have $u(D, v_i) = u(D, v_j)\ \forall v_i, v_j \in \Omega(I_n)$. ■

Following this observation, we partition the domain range into successive intervals $\{I_1, I_2, \ldots, I_k\}$ such that every two successive intervals are separated by a numerical value from the input data. We generated a score for every interval and use the exponential mechanism to choose an interval with the following probability:

$$\frac{\exp(\frac{\epsilon'}{2\Delta u}u(D, v_i)) \times |\Omega(I_n)|}{\sum_{j=1}^{k}(\exp(\frac{\epsilon'}{2\Delta u}u(D, v_j)) \times |\Omega(I_m)|)}, \tag{12}$$

where $v_i \in \Omega(I_n)$, $v_j \in \Omega(I_m)$, and $|\Omega(I_n)|$ and $|\Omega(I_m)|$ are the sizes of the intervals I_n and I_m, respectively. Once the exponential mechanism returns an interval, we randomly sample a value from the returned interval, since all its values have the same score. The randomly chosen value represents the numerical split point. The step of determining a numerical split point is outlined in Line 9 of Algorithm 1.

THEOREM 4.3. Choosing a numerical split point satisfies ϵ'-differential privacy. ■

PROOF (SKETCH) 4.3. *Choosing a numerical split point involves two steps: (1) choosing an interval I_n from a set of intervals $\{I_1, I_2, \ldots, I_k\}$; and (2) sampling a value from I_n. Proving that the first step satisfies ϵ'-differential privacy can be straightforwardly deduced from Proof 4.1 and is, therefore, omitted from this proof. The second step randomly samples a value from I_n. This step is data independent; therefore, it satisfies differential privacy by definition [Dwork 2006].* ■

EXAMPLE 4.5. *Let us reexamine Fig. 4. The numerical split value* $[1930 - 1999)$ *is chosen for specialization at the root node. Therefore, we partition the domain of attribute* YoB *into the set* $\{[1930 - 1947), [1947 - 1953), [1953 - 1955), [1955 - 1957), [1957 - 1959), [1959 - 1968), [1968 - 1999)\}$. *After that, a score is generated for each interval. Let us assume that interval* $[1959 - 1968)$ *has the highest score among all the other candidate intervals in the set. Finally, a split point is randomly picked from* $[1959 - 1968)$*, which is* 1960 *in this example, that divides* $[1930 - 1999)$ *into two intervals:* $[1930 - 1960)$ *and* $[1960 - 1999)$. ■

As we mentioned in Section 4.2, the NCP of an interval I_n is computed from the ratio $\frac{|I_n|}{|\Omega(A_{num})|}$. This results in every value in the interval having a slightly different score than the other. To avoid finding a score for every single value, we choose the midpoint to represent the score and split point of an interval.

4.4. Publishing Record Counts

In Section 4.1, we mentioned that *DiffMulti* is comprised of two phases: iteratively specializing records, followed by publishing generalized records. Phase 1 ends when no further specializations can be performed. Phase 2 publishes record counts in every leaf partition that resulted from Phase 1. However, as discussed earlier in Section 4.2, publishing true counts violates differential privacy because counts change depending on the underlying data set. Again, we use the Laplace mechanism to publish a noisy version of the true count. Fig. 4 shows the added Laplacian noise in a dashed ellipse in every leaf partition.

Recall from Section 4.1 that a partition holds $|\Omega(A^{cls})|$ groups of generalized records, we wish to release noisy group counts that satisfy $\frac{\epsilon}{2}$-differential privacy. Therefore, in Line 21, Algorithm 1 publishes $C + \texttt{Lap}(2/\epsilon)$ for every $P_{l_i} \in PL$, where C is the true group count in a leaf partition and PL denotes the set of leaf partitions. Noisy record counts are rounded down to the nearest non-negative integer, a step that does not violate differential privacy [Kifer and Lin 2010].

THEOREM 4.4. Publishing record counts satisfies $\frac{\epsilon}{2}$-differential privacy. ■

The proof is similar to Proof 4.2; therefore, it is omitted from the discussion. Next, we will examine the properties of differential privacy to prove that *DiffMulti* is differentially private.

Privacy Budget Allocation. To satisfy differential privacy, it is imperative for any operation to be insensitive to the underlying data set. In Sections 4.2- 4.4, we investigated the main operations of *DiffMulti* and showed that each operation on its own is differentially-private. To prove that *DiffMulti* is differentially-private as a whole, we will examine sequential composition (Lemma 3.1) and parallel composition (Lemma 3.2) in order to carefully allocate a privacy budget $0 < \epsilon' \leq \epsilon$ to each operation, where ϵ is an input parameter.

The algorithm exhibits two behaviors in which parallel composition can be witnessed. In Phase 1, specializing a partition $P_i \rightarrow \{P_{c_1}, \ldots, P_{c_\gamma}\}$ results in child partitions containing disjoint sets of records. Hence, allocating a privacy budget $= \epsilon_x$ to

each child partition $P_{c_j} \in \{P_{c_1}, \ldots, P_{c_\gamma}\}$ will result in a total budget consumption $= \epsilon_x$ among $\{P_{c_1}, \ldots, P_{c_\gamma}\}$. In Phase 2, a leaf partition contains $|\Omega(A^{cls})|$ groups of disjoint records. Again, assigning ϵ_x to each group in the partition to publish noisy counts will consume a total of ϵ_x.

Sequential composition manifests along a root-to-leaf path in the tree of partitions because the same set of records is being iteratively processed starting from the topmost-general root partition until a leaf partition is reached. Moreover, the set of root-to-leaf paths falls under parallel composition, as we explained above. Therefore, the privacy budget consumption is added along the *longest* root-to-leaf path.

Let us start by examining the amount of privacy budget needed to process a single partition. Lines 9, 11, and 13 of Algorithm 1 outline three ϵ'-differentially-private operations. Therefore, a single partition requires $3 \times \epsilon'$ to complete a specialization in a differentially-private way. Line 9 is a special case for the topmost-general root partition because we need to find a split point for *all* numerical attributes, whereas any subsequent child partition contains *at most one* numerical value for which we need to find a split point. Hence, Line 9 computes a split point $|A_{num}|$ times in the root partition, where $|A_{num}|$ is the number of numerical attributes. Given ϵ as the entire privacy budget for Algorithm 1, we dedicate $\frac{\epsilon}{2}$ to specializing partitions along a root-to-leaf path (Phase 1) and $\frac{\epsilon}{2}$ for publishing record counts (Phase 2). The Laplacian noise in Phase 2 is, therefore, $\texttt{Lap}(2/\epsilon)$. The budget for each of the three differentially-private operations in Phase 1 $\epsilon' = \frac{\epsilon}{2(|A_{num}|+3|g|)}$, where $|g|$ is the length of the longest path.

In conclusion, based on the above privacy analysis and Theorems 4.1- 4.4, *DiffMulti* is ϵ-differentially private.

4.5. Proposed Algorithm

Algorithm 1 outlines the key steps of our proposed method, *DiffMulti*. Lines 1-20 describe Phase 1 and Line 21 describes Phase 2. The algorithm accepts three input parameters: a raw data set D, a privacy budget ϵ, and the maximum number of specializations h. The output is an ϵ-differentially-private data set $\hat{D}$.

In Phase 1, *DiffMulti* performs a sequence of specializations on partitions. A specialization is performed only if it is *valid*. A partition that does not incur a valid specialization is considered a leaf partition, which is used in Phase 2 to publish noisy record counts.

DEFINITION 4.1 (*A valid specialization*). *Given a partition P_i, we say that P_i incurs a* valid specialization *iff all the following hold true: the number of specializations h' assigned to P_i, $P_i.h' > 0$; $\exists v \in P_i$ s.t. $|child(v)| \neq 0$; and the number of specializations from the root partition to $P_i \leq |g|$.* ■

Line 1 generalizes all raw values in D to one root partition, P_i, that contains the topmost values. Let PT be a list that holds partitions as a LIFO stack. At any given iteration, PT stores the partitions to be specialized at a later point during execution time. Line 2 initializes the list of *temporary partitions* PT with P_i. Line 3 initializes the set of *leaf partitions* PL, which stores the partitions holding the data records to be published. Line 4 sets the length of the longest root-to-leaf path to the total number of levels of all the taxonomy trees that belong to all attributes $\mathcal{A}$ in D except the *Class* attribute. We further elaborate on this point in Section 4.6. Line 5 is executed once to initialize ϵ', as per the discussion in Section 4.4. This ϵ' will be used for all the differentially-private operations in the algorithm. Lines 6-20 recursively perform specializations on the partitions in PT. Line 7 provides the next partition to be specialized, P_i, and Line 8 checks its validity. If P_i does not yield a valid specialization, it is removed from PT and added to PL in Lines 17 and 18. Otherwise, Line 9 chooses,

Algorithm 1 *DiffMulti*

Input: Raw data set D, privacy budget ϵ, and number of specializations h
Output: Diff-private & multidim-generalized data set $\hat{D}$

1: $P_i \leftarrow$ Initialize every value in D to the topmost value;
2: $PT \leftarrow P_i$;
3: $PL \leftarrow \emptyset$;
4: $|g| = \sum^{|\mathcal{A}|} |taxonomy\ tree\ height_i|$;
5: $\epsilon' \leftarrow \frac{\epsilon}{2(|A_{num}|+3|g|)}$;
6: **while** PT != $\emptyset$ **do**
7: $P_i \leftarrow PT$.pop();
8: **if** P_i incurs a valid specialization **then**
9: Determine a split point for every new valid $v_{num} \in P_i$ with probability $\propto \exp(\frac{\epsilon'}{2\Delta u} u(D, v_{num}))$;
10: Compute the score for every valid $v \in P_i$;
11: Select $v \in P_i$ with probability $\propto \exp(\frac{\epsilon'}{2\Delta u} u(D, v))$;
12: Specialize $P_i \rightarrow \{P_{c_1}, \ldots, P_{c_\gamma}\}$;
13: Use Equation 10 to determine, with $\mathtt{Lap}(1/\epsilon')$, the number of specializations h' assigned to every $P_{c_j} \in \{P_{c_1}, \ldots, P_{c_\gamma}\}$;
14: $PT \leftarrow PT - P_i$;
15: PT.push(P_{c_j}) for every $P_{c_j} \in \{P_{c_1}, \ldots, P_{c_\gamma}\}$;
16: **else**
17: $PT \leftarrow PT - P_i$;
18: $PL \leftarrow PL \cup P_i$;
19: **end if**
20: **end while**
21: **return** each group in $P_{l_i} \in PL$ with count $(C + \mathtt{Lap}(2/\epsilon))$

in a differentially-private way, a split point for every new numerical value $v_{num} \in P_i$. This step creates two child values for v_{num} with non-overlapping intervals. Line 10 computes a score for every valid value $v \in P_i$. A utility function is chosen beforehand depending on the target workload. Line 11 uses the exponential mechanism to select a value v. Line 12 specializes $v \rightarrow child(v)$ and creates a partition P_c for every child value $c \in child(v)$. After that, Line 13 assigns a noisy number of specializations h' to each child partition P_c in proportion to the number of records in P_c, where $0 \leq h' < h$. Line 14 removes the parent partition P_i from PT, whereas Line 15 pushes the set of child partitions $\{P_{c_1}, \ldots, P_{c_\gamma}\}$ onto the beginning of PT. The recursion (Lines 6-20) stops when the list of temporary partitions PT becomes empty, i.e., no further valid specialization can be performed on any partition $\in PT$. The algorithm terminates when Line 21 synthesizes the ϵ-differentially-private data set $\hat{D}$ by returning a Laplacian-noisy version of the counts in all the partitions in PL.

Top-Down Specialization. Algorithm 1 uses a tree data structure (Fig. 4) reminiscent of the one proposed in [Fung et al. 2007]. A significant difference between both relies on the fact that our structure is not indexed. Rather, our technique works on the node level due to the multidimensional aspect of our algorithm, where every node in the tree represents a partition containing a unique set of records. Such structuring of data records to generate a differentially-private and multidimensionally-generalized version of a raw data set provides the following advantages: First, the exponential

mechanism chooses from the set of candidates that exists in the same partition. This avoids having to link other candidates by scanning other partitions. Second, in any iteration, all records being specialized exist within the same partition, which provides direct access to those records. This provides efficiency by avoiding scanning the entire data set when computing scores or splitting records from parent to child partitions. Third, the tree structure allows Algorithm 1 to organize child partitions in a LIFO stack so that the tree grows in a depth-first order. This provides space efficiency by performing a sequence of specializations on records that exist within the same memory block, i.e., moving from parent to child, as opposed to moving from parent to sibling. Fourth, Algorithm 1 does not need to complete a full run to finish anonymizing the data, rather, at the end of any iteration the data is anonymous and a noisy version is ready to be published.

Complexity Analysis. We find the complexity of *DiffMulti* in terms of the number of records in the input data set, $|D|$. Given a numerical attribute A_{num}, a split point is found by first sorting the partition records on A_{num}, and then scanning them once to compute a score for every numerical value on A_{num} that appears in the partition records. Sorting partition records on a single numerical attribute costs $O(|P_i| \log |P_i|)$, where $|P_i|$ is the number of records in P_i. If P_i is the root partition, then $|P_i| = |D|$; otherwise, $|P_i|$ is usually much smaller than $|D|$. Thus, Line 9 of Algorithm 1 costs $O(|A_{num}| \times |D| \log |D|)$ for the root partition and $O(|P_i| \log |P_i|)$ otherwise, where $|A_{num}|$ is the number of numerical attributes.

In Line 10, a score is computed for every candidate in the current partition. Data records in the root partition are scanned once for every attribute to determine the score. For all other partitions, no record scanning is necessary because all the required pieces of information have been obtained in an earlier iteration (Line 12); thus, this operation can be done in a constant time. Therefore, given a set of attributes $\mathcal{A}$, and assuming that all candidates in P_i are valid, Line 10 costs $O(|\mathcal{A}| \times |P_i|)$ for the root partition and $O(1)$ otherwise.

The complexity of the exponential mechanism is proportional to the number of selections from which the mechanism will choose. The exponential mechanism appears twice in the algorithm. First, in Line 9 the purpose is to select a split point from a given numerical attribute. This process requires partitioning the numerical attribute into successive intervals $\{I_1, I_2, \ldots, I_k\}$ then choosing one interval from the set (as demonstrated in Example 4.5). Therefore, the cost of the exponential mechanism in Line 9 is $O(k)$. Second, in Line 11 the purpose is to select a candidate from a set of candidates, which has a maximum size equal to the number of attributes in D. Hence, the cost of Line 11 is $O(|\mathcal{A}|)$. $|D|$ is usually much larger than the number of intervals and the number of attributes. We, therefore, choose to neglect the complexity of the exponential mechanism.

In Line 12, the current partition is specialized on the selected value v from Line 11. The records in P_i are distributed among its child partitions, a step that requires scanning those records to determine the child partition to which every record belongs. Thanks to the tree structure, partition P_i gives direct access to pertinent records in D. To boost the efficiency of our implementation, along with record scanning, we collect pieces of information that will be used in computing the scores in the child partitions in subsequent iterations (Line 10). Specifically, for each $c \in child(v)$ we collect $|D_c|$, $|D_c^{cls}|$, $|D_z|$, and $|D_z^{cls}|$, where cls is a value on the *Class* attribute and $z \in child(c)$. Thus, specializing a partition costs $O(|P_i|)$. The rest of the lines in Algorithm 1 are done in a constant time.

From the analysis above, the most expensive operation in the algorithm is sorting partition records (Line 9). Given that the algorithm performs at most h specializations,

and given a fixed number of attributes, the total cost of the algorithm is $O(h \times |P_i| \times log|P_i|)$. In order to express the time complexity in terms of the number of records in the input data set, $|D|$, we assume the worst case by which partitions are specialized according to a *binary* tree structure where every leaf partition contains 1 record. Every level collectively processes $|D|$ records and the tree can have at most $log|D|$ level. As a result, ***DiffMulti*** has a worst-case runtime of $O(|D| \cdot \log |D| \cdot \log |D|)$.

4.6. Discussion

Generalization Hierarchies. Unlike existing methods [Chen et al. 2011] that assume a context-free hierarchy (one that does not preserve the underlying semantics of the data), we assume that a hierarchy for a categorical attribute in the context of relational data publishing is pre-defined by the data holder and is specific to the hierarchy's corresponding attribute. Yet, we can provide some general guideline. If the data holder has the flexibility to define multiple hierarchies, defining a flat hierarchy should be avoided because a hierarchy with a large number of child nodes results in a small number of records in the child partitions (Fig. 4). Consequently, increasing the effect of the added noise upon records publishing. Whereas a better hierarchy contains less child nodes and more depth. Not only does such design allow for more generalization options, but it also produces less noisy record counts because partitions at the higher level of the generalization tree have larger record counts, minimizing the effect of the added Laplace noise.

Numerical Hierarchies. Our approach dynamically generates hierarchies for numerical attribute, as opposed to performing a preprocessing step to discretize numerical domains, for the following reasons. Unlike categorical attributes, data holders often do not have a good sense on how to discretize numerical attributes. It is more user-friendly if the algorithm can automatically discretize the attributes on-the-fly. Suppose the data holder wants to pre-define a hierarchy for a numerical attribute. There are two possible ways. The first way is to define the hierarchy independently of the raw data. This may result in poor utility, e.g., lower classification accuracy, because the discretization process does not consider the distribution of the classes. Our proposed approach, on the other hand, takes data utility into consideration in the anonymization and discretization processes. The second way is to pre-define a hierarchy from the raw data. In this case, the process must be differentially-private. Our proposed method incorporates the discretization into the specialization process. That is, a numerical attribute is discretized only after being compared with the rest of the attributes in the input data set. Otherwise, discretization will waste privacy budget by making unnecessary specializations. For the above reasons, dynamic discretization is more elegant and user-friendly than having a separate preprocessing step.

Longest Path. In Section 4.4, we discussed privacy budget allocation and how to compute ϵ' given that partitions are structured as a tree. In such tree, the *length* of a root-to-leaf path refers to the number of specializations along that path. Thanks to the parallel composition property, we only need to consider the length of the longest root-to-leaf path, $|g|$, when computing ϵ' because g performs the longest sequence of differentially-private operations. For example, the length of the longest root-to-leaf path in Fig. 4 is 3.

Knowing $|g|$ in advance is a challenging task because: (1) the depth of the tree of partitions is indeterministic due to multidimensional generalization; and (2) ϵ' needs to be computed in advance so that ***DiffMulti*** can perform the differentially-private operations in Phase 1 (Lines 9, 11, and 13). Herein, we provide theoretical and practical estimation of $|g|$.

Every partition specialization in the tree decreases the number of specializations h by 1. We consider the following two cases of tree growth. If the tree fans out, i.e., the

Table II. *Adult* data set

# of records	45,222
# of numerical attributes	6
# of categorical attributes	8
# of *Class* attributes	1 (*salary*: "≤ 50K" or "> 50K")
Total # of attributes	15

Table III. Notations

Notation	**Description**
ϵ	Privacy budget
h	Number of specializations
BA	Baseline Accuracy
LA	Lower Bound Accuracy
C4.5, *SVM*, *ID3*	Classifiers
k in k-anonymity	Anonymity threshold
k in k-means	Number of clusters

upper levels contain a large number of partitions, then h will be mainly consumed horizontally. With little h left, the tree will not grow deeper. However, if the tree branches with small number of child partitions, then more h will be available to allow for the tree to grow deeper. Our interest lies in the case where the tree grows as deep as possible; therefore, we consider a perfect binary tree of partitions where all leaf partitions are at the same level and every leaf partition contains 1 record from D. With $|D|$ leaf partitions, the hight of the tree would be $log_2|D|$. Hence, the length of the longest path is $|g| = log_2|D|$. We note that this is a theoretical estimate built on the assumption that all data records conform to the perfect binary distribution of partitions. It is safe to say that real-life data sets are unlikely to follow such tree structure as some child partitions are likely to be empty causing the tree to grow deeper than $log_2|D|$ levels. An alternative, and more practical, solution is to consider the number of specializations required to transform a record from its topmost general version to its raw version. This is achieved by summing the heights of all the taxonomy trees for a given data set. The result is an integer that represents the length of longest possible root-to-leaf path. For all the experiments in the following section, we use the latter approach to estimate $|g|$.

5. EXPERIMENTAL EVALUATION

We evaluate the performance of our proposed method, *DiffMulti*, with respect to the utility of the output data, efficiency in terms of runtime, and scalability for handling large data sets. The goal is to measure the impact of multidimensional generalization achieved by means of differential privacy. We compare our proposed algorithm with 4 closely related ones: (1) *DiffGen* [Mohammed et al. 2011], a differentially-private algorithm that performs single-dimensional (global) generalization; (2) *Mondrian* [LeFevre et al. 2006; 2008], an algorithm that enforces multidimensional generalization to publish k-anonymous data; (3) *DPCube* [Xiao et al. 2014], a differentially-private method for releasing private histograms in the interactive setting; and (4) *PrivBayes* [Zhang et al. 2014], a differentially-private algorithm that utilizes Bayesian networks to release high-dimensional data that approximate the distribution of the input data. Furthermore, we report the data utility of our method and that of *RPS* [Qardaji and Li 2012], a general framework for releasing relational data under differential privacy.

For all our evaluations, we use the widely employed and publicly available *Adult* data set [Frank and Asuncion 2010]. Tables II and III include a summary description of *Adult* and some of the notions used throughout this section, respectively.

Both *DiffMulti* and *DiffGen* are top-down specializations approaches; therefore, they share the notion of *the number of specializations*, h. When comparing *DiffMulti* with

DiffGen, we vary the number of specializations h in both methods and report the results. Even though h is an input parameter for both methods, the value of h ranges differently, as explained below.

h is an integer whose maximum value is tied to the taxonomy trees. *DiffGen* employs single-dimensional generalization, and thus h can be at most equal to the total of the number of non-leaf nodes in all the taxonomy trees. Whereas *DiffMulti* employs multidimensional generalization, resulting in h being massively greater than *DiffGen*'s h. More specifically, the maximum number of specializations for *DiffMulti* is computed as follows:

$$\begin{aligned} h = 1 &+ Inter_1 \\ &+ Leaves_1 + (Leaves_1 \times Inter_2) \\ &+ (Leaves_1 \times Leaves_2) + (Leaves_1 \times Leaves_2 \times Inter_3) \\ &+ \ldots \\ &+ (Leaves_1 \times Leaves_2 \times \ldots \times Leaves_{m-1}) \\ &+ (Leaves_1 \times Leaves_2 \times \ldots \times Leaves_{m-1} \times Inter_m). \end{aligned} \tag{13}$$

Or,

$$h = 1 + Inter_1 + \sum_{i=1}^{m-1} [(\prod_{j=1}^{i} Leaves_j)(1 + Inter_{i+1})], \tag{14}$$

where $Inter_i$ is the number of intermediate nodes in taxonomy tree i, $Leaves_i$ is the number of leaf nodes in taxonomy tree i, and m is the total number of taxonomy trees. Considering only the taxonomy trees of the categorical attributes of the *Adult* data set, h has a limit of nearly 50 for *DiffGen*, while for *DiffMulti* h is in millions. Those numbers assume that every combination of raw values exists in the raw data sets, which is not the case empirically.

There exists no sense of direct comparison between the number of specializations of *DiffMulti* and that of *DiffGen*. Therefore, when comparing the performance of both methods, we reasonably vary the value of h for both methods in an attempt to maintain relatively consistent results. In some other cases, the range might vary differently in order to display a certain behavior in the performance of a method.

The settings and configurations of the experiments are as follows: we implemented our method in C++. All experiments run on a PC with an Intel Core i5 CPU, 2.4GHz, and 8GB of RAM. Unless otherwise mentioned, we run each experiment 20 times and report the average of the obtained results in the graphs herein. Moreover, recall from Section 4.4 that $\epsilon' = \frac{\epsilon}{2(|A_{num}|+3|g|)}$, where g is the longest root-to-leaf path. Section 4.6 presented an estimation of $|g|$ by summing the heights of all the taxonomy trees. *Adult* has 8 distinct taxonomy trees pertaining to 8 categorical attributes. Summing the heights of these taxonomy trees yields 21. For the remaining 6 numerical attributes, no such hierarchy exists and so we assume the height of each attribute's taxonomy tree is 7 on average. Hence, the total sum is 63, and thus we set $|g| = 63$ for all our experiments. Specialization stops when a root-to-leaf path reaches length $= |g|$ as the dedicated privacy budget would have been consumed entirely. If the longest root-to-leaf path stopped before reaching $|g|$, the unused portion of the privacy budget is added to the privacy budget dedicated to the leaf partitions in order to preserve the entire budget ϵ. It is worth noting that choosing a height for a numerical taxonomy tree is a loose assumption, and reasonably varying this number has no significant impact on *DiffMulti*.

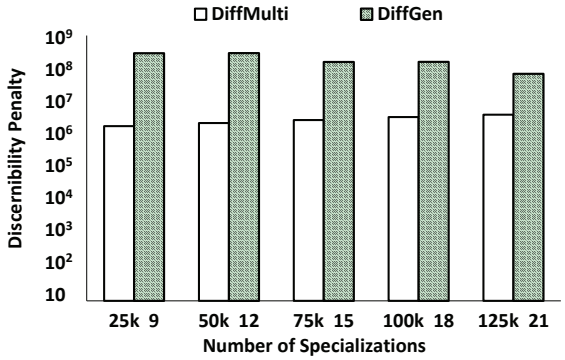


Fig. 5. Comparing *DiffMulti* and *DiffGen* in terms of discernibility penalty

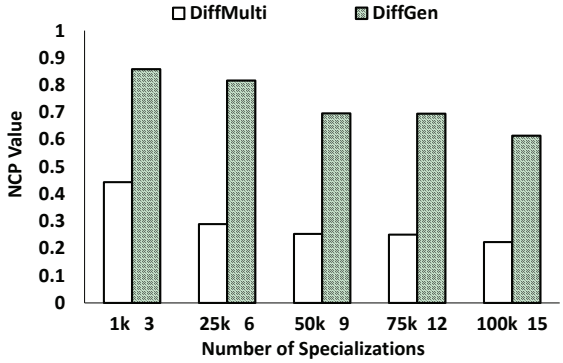


Fig. 6. Comparing *DiffMulti* and *DiffGen* in terms of NCP

5.1. Data Utility

We test our method using two general-purpose metrics, i.e., discernibility [Bayardo and Agrawal 2005] and NCP [Xu et al. 2006], and two specific target workloads, i.e., classification and clustering.

General-Purpose Metrics. Fig. 5 and 6 depict the utility of the output data set where discernibility [Bayardo and Agrawal 2005] and NCP [Xu et al. 2006] are the utility functions, respectively. Recall from Section 4.2 that both functions are general-purpose metrics that measure the amount of distortion in the output data set in comparison with its raw version. We compare our method *DiffMulti* with *DiffGen* in order to showcase the impact of multidimensional generalization over single-dimensional. In both figures, we vary the number of specializations of both methods and measure the distortion; higher reported values indicate higher distortion. Fig. 5 suggests that *DiffMulti* is able to significantly reduce the discernibility penalty by at least one order of magnitude than *DiffGen*. Fig. 6 reports the NCP values, where $0 \leq \texttt{NCP} \leq 1$ and $\texttt{NCP} = 1$ means that the output data set is generalized to the topmost value in the taxonomy tree of every attribute. Again, *DiffMulti* manages to maintain significantly less generalized output data thanks to its multidimensional approach. Both figures suggest that *DiffMulti* is robust with respect to the number of specializations as the incurred distortion does not change significantly.

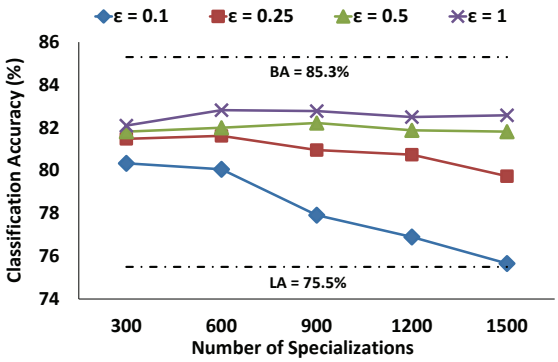


Fig. 7. *DiffMulti*: the impact of ϵ on classification accuracy

Classification Analysis. The objective of classification analysis is to build a classification model that accurately predicts the *Class* attribute in *Adult*; particularly, if an individual has income "> 50K". To achieve this goal, first we use the training records to create generalization regions in the multidimensional space. Then, we generalize the training and testing records to those regions. Training records are used to build a classification model, which in turn is used to predict the *Class* attribute in the testing records. For this set of experiments, we used three widely deployed classifiers: *C4.5* [Quinlan 1993], *ID3* [Hall et al. 2009], and *SVM* [Joachims 1999]. *C4.5* and *ID3* are trained with 2/3 of *Adult* records and tested with the remaining 1/3 records, whereas *SVM* is trained with 4/5 of the records and tested with the remaining 1/5. For all classifiers, we set `Max` to be the utility function.

Before applying any anonymization method on *Adult*, we first classify *Adult* as a raw data set to measure the *Baseline Accuracy* (BA), which is the best achieved accuracy. We aspire to obtain results close to the BA when classifying the output data of an anonymization method, which typically results in lower accuracies than the BA. Moreover, we measure the *Lower Bound Accuracy* (LA), which considers solely the *Class* attribute when training and testing the classifier. LA gives the sense of having a worst-case accuracy that should be surpassed by the output of a proposed anonymization method. Fig. 7, 8, 9, and 10 measure classification accuracy in percentage, where higher values imply better accuracy.

Fig. 7 depicts the impact of differential privacy on classification accuracy using *C4.5* classifier. We test *DiffMulti* by varying the privacy budget $0.1 \leq \epsilon \leq 1$ and the number of specializations $300 \leq h \leq 1500$. The privacy budget should not be set large, i.e., typically ≤ 1 [Dwork 2006; Friedman and Schuster 2010; Dwork 2011]. Higher privacy budgets result in more accurate classification because the differentially-private operations of the algorithm, i.e., data partitioning and publishing record counts, incur less noisy outputs.

Fig. 8 compares *DiffMulti* with both *DiffGen* and *Mondrian* in terms of classification accuracy using *C4.5* classifier. We set $\epsilon = 1$ in both *DiffMulti* and *DiffGen* and vary the number of specializations $1k \leq h \leq 10k$ and $2 \leq h \leq 20$, respectively. *Mondrian* is not a differentially-private method; rather, it publishes k-anonymous data via multidimensional generalization. Since both *DiffMulti* and *Mondrian* are multidimensional approaches, we compare both methods by setting $k = 60$ for *Mondrian*. Hence, the latter is represented by a single line in Fig. 8. We observe that *DiffMulti* is dominant in achieving higher accuracy, especially at lower numbers of specializations. However, for higher values of h, *Mondrian* achieves up to 2% better accuracy than *DiffMulti*. We

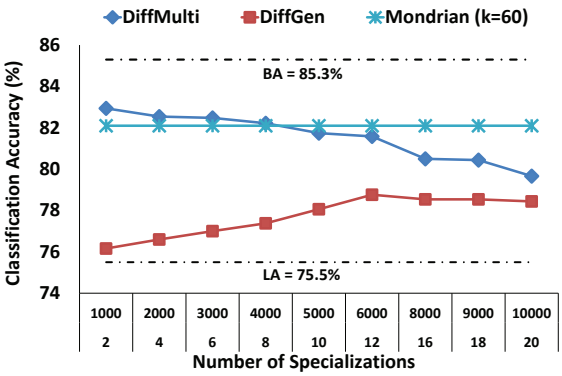


Fig. 8. Comparing *DiffMulti*, *DiffGen*, and *Mondrian* in terms of classification accuracy

note the following. First, *DiffMulti* possesses the flexibility to achieve better accuracies than *DiffGen* and *Mondrian*, given a certain range of h. Second, the 2% improvement in accuracy when applying k-anonymity-based *Mondrian* comes at the cost of releasing anonymous data that is vulnerable to attacker's background knowledge [Wong et al. 2007; Ganta et al. 2008; Kifer 2009]. On the other hand, any attack on the data released by our proposed differentially-private method is independent of any attacker's background knowledge or computational power, thanks to the strong privacy guarantees offered by differential privacy.

The authors of *DiffGen* conducted a similar experiment in [Mohammed et al. 2011] to compare the classification accuracy of their non-interactive *DiffGen* with the interactive *DiffP-C4.5* [Friedman and Schuster 2010], an algorithm that maintains differential privacy when building a classifier. Although *DiffP-C4.5* achieved promising results, *DiffGen* obtained improved accuracy at $\epsilon = 1$. However, the latter was constantly surpassed by *DiffMulti* for $h \leq 10k$. For $h > 10k$, however, the accuracy achieved by *DiffMulti* starts to decay. Taking all the reported results together, Fig. 8 suggests that *DiffMulti* is flexible to achieve comparable or better classification accuracy than existing methods for a certain range of h.

Fig. 7 and 8 show that as the number of specializations increases, the classification accuracy of *DiffMulti* decreases. This is due to the following two reasons. First, a higher number of specializations results in longer root-to-leaf paths. Consequently, the computed ϵ' assigned to a single differentially-private operation becomes very small compared to that of *DiffGen*, resulting in poor partitioning decisions that contribute to building an inaccurate classifier. Second, a higher number of specializations yields a larger number of leaf partitions. Consequently, each partition contains fewer records. As the number of records decreases, the effect of the added Laplace noise in Phase 2 of Algorithm 1 increases. With a relatively large noise, true record counts undergo severe distortion, which in turn adversely affects building a proper decision tree. This finding confirms that for high-dimensional data, publishing noisy counts of every possible combination of domain values greatly degrades the quality of analysis.

Fig. 9 compares *DiffMulti* with *DPCube* in terms of classification accuracy by using *ID3* classifier. We follow the same experimental settings as in [Xiao et al. 2014]: we use 30,162 records for training and 15,060 for testing, and we select from *Adult* the attributes *workclass*, *martial-status*, *race*, and *sex*. We can see that *DiffMulti* achieves slightly better classification accuracy than *DPCube*. While both methods achieve comparable accuracy, *DiffMulti* improves over *DPCube* by (1) being able to handle numerical attributes, and (2) being a non-interactive method that releases anonymous data

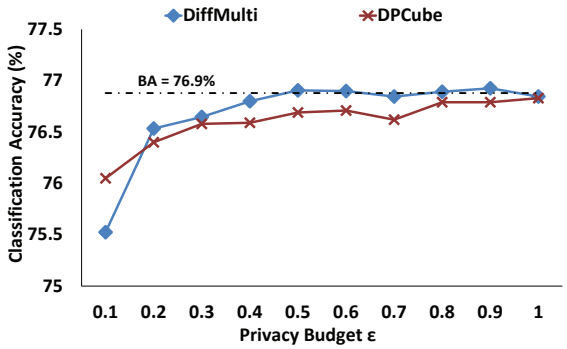


Fig. 9. Comparing *DiffMulti* and *DPCube* in terms of classification accuracy

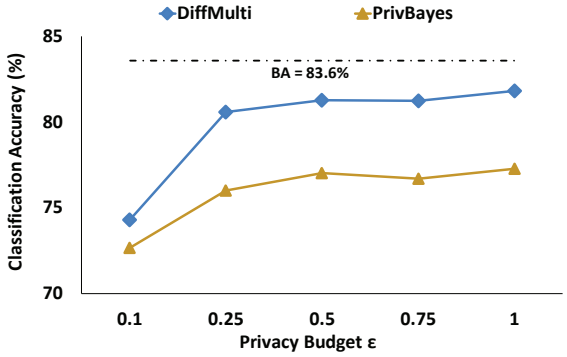


Fig. 10. Comparing *DiffMulti* and *PrivBayes* in terms of classification accuracy

with no restriction on data access or the number of users querying the published data. We also observe that in some cases *DiffMulti* is able to perform better than the BA. This is because DiffMulti transforms raw data by means of generalization, which tends to reduce noise for the classifier.

Fig. 10 depicts the performance of *DiffMulti* and *PrivBayes* using *SVM* classifier. We set the number of specializations for *DiffMulti* $h = 10k$ and used the default parameters for *PrivBayes*. We ran both methods over 5 different privacy budgets $0.1 \leq \epsilon \leq 1$ and reported the resulting accuracy of each run. *DiffMulti* constantly performs more accurate predictions than *PrivBayes* even at low privacy budgets (noisy output data). We note that, unlike *PrivBayes* which outputs raw data values, our solution performs a series of carefully selected generalizations which contribute to improving classification accuracy, as suggested by Fig. 8.

We carry out an additional experiment in which we compare *DiffMulti* with the *RPS* framework [Qardaji and Li 2012] in terms of classification accuracy. The accuracy of *RPS* was taken from the authors' paper. The same settings of *RPS* experiments were applied when conducting our experiment: *Adult* data set of 30,162 records and 11 attributes (See [Qardaji and Li 2012]), and 5-fold cross validation for measuring the classification accuracy. We fixed $\epsilon = 1$ for both methods and obtained the following results: the accuracy of the raw data $BA = 82.96\%$. For *RPS*, 79% was the highest achieved

accuracy, whereas *DiffMulti* achieved 81.54% when the number of specializations was set to $1k$ (explanation is in the above paragraph).

Cluster Analysis. The objective of this experiment is two-fold:

(1) We would like to measure the similarity between the clustering solutions resulting from clustering the raw data and the anonymous data. This will give a sense of how much "similarity" is preserved between anonymous records.
(2) We would like to evaluate the benefit of performing workload-aware partitioning, as opposed to general-purpose partitioning.

To achieve the **first objective**, we implement an intuitive evaluation method, called Match Point, that was proposed by Fung et al. [Fung et al. 2009]. We first create a $|D|$-by-$|D|$ matrix, $Matrix(D)$, that represents the clustering solution resulting from clustering the raw *Adult*, where $|D|$ is the number of records in *Adult*. Every row and column in the matrix represent a unique record, in the same order. A cell in $Matrix(D)$ contains the value 1 if the i^{th} record and the j^{th} record belong to the same cluster; 0 otherwise. Similarly, we create a $|\hat{D}|$-by-$|\hat{D}|$ matrix, $Matrix(\hat{D})$, that represents the clustering solution resulting from clustering the anonymous *Adult*. Match Point is defined as the percentage of matched values between $Matrix(D)$ and $Matrix(\hat{D})$. That is, if cells $Cell_{ij}$ in $Matrix(D)$ and its corresponding cell in $Matrix(\hat{D})$ have the same value, then we increase the match score by 1; 0 otherwise. Match Point is defined as follows:

$$MatchPoint(Matrix(D), Matrix(\hat{D}) = \frac{\sum_{1 \leq i,j \leq |D|} Cell_{ij}}{|D|^2}. \quad (15)$$

Match Point is defined over the range of [0, 1], where Match Point $= 1$ indicates a perfect match between the raw cluster and the anonymous cluster. Note that we compare the clustering solution of the raw records with the clustering solution of the *true* anonymous records because this will give us a more accurate insight of how effective *DiffMulti* is in preserving the "similarity" between data records, with no impact on the utility of the overall released data.

To achieve the **second objective**, we choose `Max` as a utility function in order to allow *DiffMulti* to perform workload-aware partitioning of the data. Additionally, we choose discernibility as a general-purpose utility function that minimizes distortion in general. We call the former set of experiments *ClusterMP* and the latter *DistortMP*.

In order to perform cluster analysis on raw *Adult*, we remove the *Class* attribute (*salary*) from the data set, cluster the data set, identify the cluster ID of every record in the resulting clustering solution, and add cluster ID as a *Class* attribute to the entire data set. After that, we use *DiffMulti* to anonymize *Adult*, which has raw cluster ID as the *Class* attribute. Finally, we remove the *Class* attribute from the anonymous data set, cluster the true anonymous records, and add the anonymous cluster ID as a *Class* attribute to the anonymous data set. For more on the process of cluster analysis, we refer the reader to [Fung et al. 2009].

Fig. 11 depicts the Match Point results for ClusterMP and DistortMP. We use the bisecting k-means clustering method [Kaufman and Rousseeuw 2009] provided by the CLUTO Clustering Toolkit [Karypis 2006]. We set the privacy budget $\epsilon = 1$, the number of clusters $k = 2$, and vary the number of specialization $1k \leq h \leq 7k$. From Fig. 11, we can make two observations. First, *DiffMulti* is able to preserve the proximity between data points in the high-dimensional space and, therefore, significantly improve cluster quality under the workload-aware partitioning strategy (ClusterMP) over the general-purpose partitioning strategy (DistortMP) - the experiments in ClusterMP constantly achieve Match Point $\geq 90\%$. Second, as the number of specializations in-

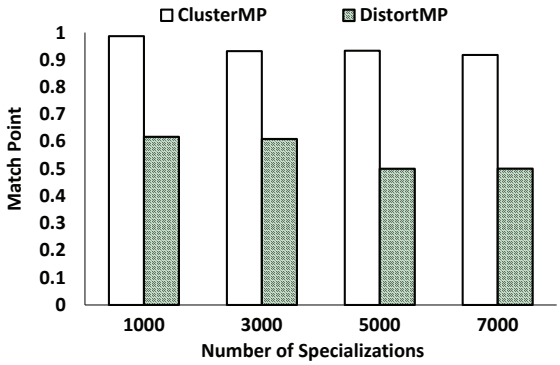


Fig. 11. Testing *DiffMulti* for clustering

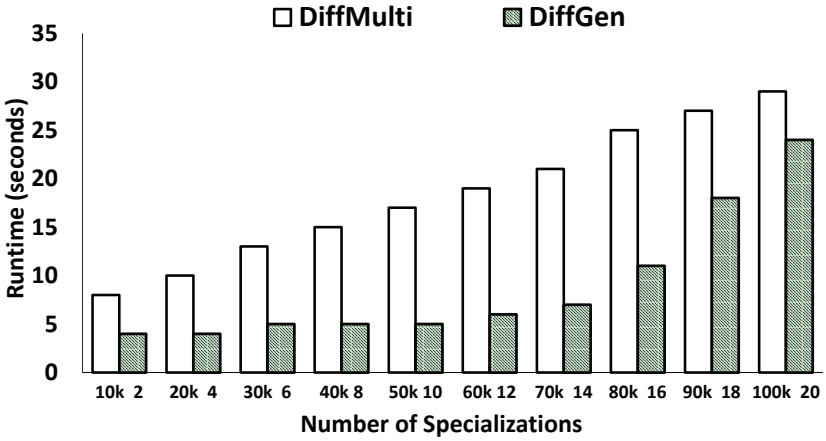


Fig. 12. Comparing *DiffMulti* and *DiffGen* in terms of runtime

creases, the similarity between the clustering solutions before and after anonymization begins to decrease. This is because *DiffMulti* performs a sequence of noisy specializations in order to satisfy differential privacy. As a result, less general data points are more affected by the noisy choice of a value to be specialized.

5.2. Efficiency and Scalability

Efficiency. In this experiment, the objective is to measure the runtime of our method and compare the results to those achieved by *DiffGen*. Fig. 12 depicts the runtime in seconds where the utility function is `Max`, $\epsilon = 1$, and the number of specializations is $10k \leq h \leq 100k$ for *DiffMulti* and $2 \leq h \leq 20$ *DiffGen*. The runtime of *DiffMulti* when performing $100k$ specializations is nearly 30s. Results suggest that our method, though it runs a few seconds longer than *DiffGen*, is insensitive to the number of specializations as the increase in runtime is insignificant. For a clear visualization, we did not include *Mondrian* in Fig. 12 because its runtime varied from a few minutes to nearly an hour for $20 \leq k \leq 100$. For the experiments in Fig. 10, our method took 10s to complete compared to a few minutes for *PrivBayes*, which was running on its default parameters. Particularly, the parameter that specifies the degree of Bayesian network

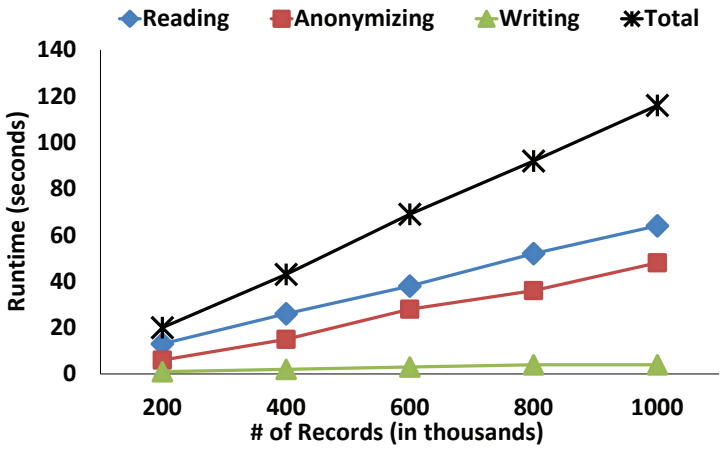


Fig. 13. Scalability of *DiffMulti*

k was set to 3. As k increases, the runtime of *PrivBayes* substantially increases; e.g., setting $k = 5$ causes *PrivBayes* to run for a few hours [Zhang et al. 2014].

Scalability. The purpose of this experiment is to examine the ability of our method to handle large data sets in terms of runtime. We generated 5 variations of the raw *Adult* data set, as follows: for every record in *Adult*, we generate $r - 1$ other records that contain some values drawn randomly from their pertinent attribute domain. The result is 5 data sets ranging in size from 200,000 to 1 million records. We run *DiffMulti* once on each data set, where the utility function is `Max` and $\epsilon = 1$. Fig. 13 depicts the runtime in seconds for reading the records, anonymizing the entire data set, and writing the differentially-private multidimensionally-generalized records. Anonymizing an *Adult*-variant data set with 1 million records finishes in less than 120s. Fig. 13 suggests that the runtime of *DiffMulti* increases reasonably as the number of records increases.

Overall. The privacy budget has a direct impact on the utility of the output data, and runtime is incremental with respect to the number of specializations and the number of input records.

6. CONCLUSION

In this chapter, we propose a differentially-private and multidimensional generalization algorithm, called *DiffMulti*. Publishing anonymized data in a *non-interactive* setting provides flexibility of usage to data recipients, as opposed to providing anonymized answers tailored to specific queries. We adopt differential privacy as our privacy model due to its independence of any attacker's background knowledge and insensitivity to the underlying data. Parallel to preserving privacy, the utility of the anonymized data is equally essential. Therefore, we effectively generalize the raw data into multidimensional regions to minimize information loss. Experimental evaluation on a real-life data set suggests that our proposed method is able to reduce information loss by at least one order of magnitude when compared with single-dimensional generalization, and improves data utility when compared with state-of-the-art private data release methods.

ACKNOWLEDGMENTS

The research is supported in part by the Discovery Grants (356065-2013) from the Natural Sciences and Engineering Research Council of Canada (NSERC), Canada Research Chairs Program (950-230623), Research Incentive Funds (R15046 and R15048) from Zayed University, and Research Grants (61272306) from the National Natural Science Foundation of China (NSFC). The work was partially completed while Benjamin C. M. Fung was visiting the Department of Computer Science at Hong Kong Baptist University.

REFERENCES

B. Barak, K. Chaudhuri, C. Dwork, S. Kale, F. McSherry, and K. Talwar. 2007. Privacy, Accuracy, and Consistency Too: A Holistic Solution to Contingency Table Release. In *Proceedings of the 26th ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems (PODS '07)*. 273–282.

R. J. Bayardo and R. Agrawal. 2005. Data Privacy Through Optimal k-Anonymization. In *Proceedings of the 21st International Conference on Data Engineering (ICDE '05)*. 217–228.

A. Blum, K. Ligett, and A. Roth. 2008. A Learning Theory Approach to Non-interactive Database Privacy. In *Proceedings of the Fortieth Annual ACM Symposium on Theory of Computing (STOC '08)*. 609–618.

D. M. Carlisle, M. L. Rodrian, and C. L. Diamond. 2007. *California Inpatient Data Reporting Manual, Medical Information Reporting for California, 5th Edition*. Technical Report. Office of Statewide Health Planning and Development.

S. Chawla, C. Dwork, F. McSherry, A. Smith, and H. Wee. 2005. Toward Privacy in Public Databases. In *Proceedings of the Second International Conference on Theory of Cryptography (TCC '05)*. 363–385.

R. Chen, N. Mohammed, B. C. M. Fung, B. C. Desai, and L. Xiong. 2011. Publishing Set-Valued Data via Differential Privacy. *The Proceedings of the VLDB Endowment* 4, 11 (2011), 1087–1098.

G. Cormode, C. Procopiuc, D. Srivastava, and T. T. L. Tran. 2012. Differentially Private Summaries for Sparse Data. In *Proceedings of the 15th International Conference on Database Theory (ICDT '12)*. 299–311.

B. Ding, M. Winslett, J. Han, and Z. Li. 2011. Differentially Private Data Cubes: Optimizing Noise Sources and Consistency. In *Proceedings of the 2011 ACM SIGMOD International Conference on Management of Data (SIGMOD '11)*. 217–228.

C. Dwork. 2006. Differential Privacy. In *Proceedings of the 33rd International Conference on Automata, Languages and Programming - Volume Part II (ICALP '06)*. 1–12.

C. Dwork. 2008. Differential Privacy: A Survey of Results. In *Proceedings of the 5th International Conference on Theory and Applications of Models of Computation (TAMC '08)*. 1–19.

C. Dwork. 2011. A Firm Foundation for Private Data Analysis. *Commun. ACM* 54, 1 (2011), 86–95.

C. Dwork, F. McSherry, K. Nissim, and A. Smith. 2006. Calibrating Noise to Sensitivity in Private Data Analysis. In *Proceedings of the Third Conference on Theory of Cryptography (TCC '06)*. 265–284.

C. Dwork and A. Roth. 2014. The Algorithmic Foundations of Differential Privacy. *Foundations and Trends in Theoretical Computer Science* 9, 3–4 (2014), 211–407.

A. Frank and A. Asuncion. 2010. UCI Machine Learning Repository. (2010). http://archive.ics.uci.edu/ml

A. Friedman and A. Schuster. 2010. Data Mining with Differential Privacy. In *Proceedings of the 16th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD '10)*. 493–502.

B. C. M. Fung, K. Wang, L. Wang, and P. C. K. Hung. 2009. Privacy-preserving Data Publishing for Cluster Analysis. *Data & Knowledge Engineering* 68, 6 (2009), 552–575.

B. C. M. Fung, K. Wang, and P. S. Yu. 2007. Anonymizing Classification Data for Privacy Preservation. *IEEE Transactions on Knowledge and Data Engineering* 19, 5 (2007), 711–725.

S. R. Ganta, S. Kasiviswanathan, and A. Smith. 2008. Composition Attacks and Auxiliary Information in Data Privacy. In *Proceedings of the 14th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD '08)*. 265–273.

Katie Hafner. 2006. And if You Liked the Movie, a Netflix Contest May Reward You Handsomely. *New York Times* (October 6, 2006).

M. Hall, E. Frank, G. Holmes, B. Pfahringer, P. Reutemann, and I. H. Witten. 2009. The WEKA Data Mining Software: An Update. *ACM SIGKDD Explorations Newsletter* 11, 1 (2009), 10–18.

M. Hay, V. Rastogi, G. Miklau, and D. Suciu. 2010. Boosting the Accuracy of Differentially Private Histograms Through Consistency. *Proceedings of the VLDB Endowment* 3, 1-2 (2010), 1021–1032.

V. S. Iyengar. 2002. Transforming Data to Satisfy Privacy Constraints. In *Proceedings of the Eighth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD '02)*. 279–288.

T. Joachims. 1999. Making Large-Scale SVM Learning Practical. In *Advances in Kernel Methods - Support Vector Learning*, B. Schölkopf, C. Burges, and A. Smola (Eds.). MIT Press, Cambridge, MA, Chapter 11, 169–184.

G. Karypis. October 2006. CLUTO - Software for Clustering High-Dimensional Datasets. (October 2006). http://glaros.dtc.umn.edu/gkhome/views/cluto.

L. Kaufman and P. J. Rousseeuw. 2009. *Finding Groups in Data: An Introduction to Cluster Analysis*. Vol. 344. John Wiley & Sons.

D. Kifer. 2009. Attacks on Privacy and de Finetti's Theorem. In *Proceedings of the 2009 ACM SIGMOD International Conference on Management of Data (SIGMOD '09)*. 127–138.

D. Kifer and B.-R. Lin. 2010. Towards an Axiomatization of Statistical Privacy and Utility. In *Proceedings of the 29th ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems (PODS '10)*. 147–158.

K. LeFevre, D. J. DeWitt, and R. Ramakrishnan. 2006. Mondrian Multidimensional K-Anonymity. In *Proceedings of the 22nd International Conference on Data Engineering (ICDE '06)*.

K. LeFevre, D. J. DeWitt, and R. Ramakrishnan. 2008. Workload-Aware Anonymization Techniques for Large-Scale Datasets. *ACM Transactions on Database Systems* 33, 3 (2008), 17:1–17:47.

C. Li, M. Hay, V. Rastogi, G. Miklau, and A. McGregor. 2010. Optimizing Linear Counting Queries Under Differential Privacy. In *Proceedings of the 29th ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems (PODS '10)*. 123–134.

H. Li, L. Xiong, and X. Jiang. 2014. Differentially private synthesization of multi-dimensional data using copula functions. In *Proceedings of the 17th International Conference on Extending Database Technology (EDBT '14)*, Vol. 2014. 475–486.

N. Li, T. Li, and S. Venkatasubramanian. 2007. t-Closeness: Privacy Beyond k-Anonymity and ℓ-Diversity. In *Proceedings of the 23rd International Conference on Data Engineering (ICDE '07)*. 106–115.

A. Machanavajjhala, J. Gehrke, D. Kifer, and M. Venkitasubramaniam. 2006. ℓ-diversity: Privacy Beyond k-anonymity. In *Proceedings of the 22nd IEEE International Conference on Data Engineering (ICDE '6)*. 24–24.

F. McSherry. 2009. Privacy Integrated Queries. In *Proceedings of the 2009 ACM SIGMOD International Conference on Management of Data (SIGMOD '09)*. 19–30.

F. McSherry and K. Talwar. 2007. Mechanism Design Via Differential Privacy. In *Proceedings of the 48th Annual IEEE Symposium on Foundations of Computer Science (FOCS '07)*. 94–103.

N. Mohammed, R. Chen, B. C. M. Fung, and P. S. Yu. 2011. Differentially Private Data Release for Data Mining. In *Proceedings of the 17th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD '11)*. 493–501.

W. Qardaji and N. Li. 2012. Recursive Partitioning and Summarization: A Practical Framework for Differentially Private Data Publishing. In *Proceedings of the 7th ACM Symposium on Information, Computer and Communications Security (ASIACCS '12)*. 38–39.

W. Qardaji, W. Yang, and N. Li. 2013a. Differentially private grids for geospatial data. In *Proceedings of the 29th IEEE International Conference on Data Engineering (ICDE '13)*. 757–768.

W. Qardaji, W. Yang, and N. Li. 2013b. Understanding Hierarchical Methods for Differentially Private Histograms. *Proceedings of the VLDB Endowment* 6, 14 (2013), 1954–1965.

W. Qardaji, W. Yang, and N. Li. 2014. PriView: Practical Differentially Private Release of Marginal Contingency Tables. In *Proceedings of the 2014 ACM SIGMOD International Conference on Management of Data (SIGMOD '14)*. 1435–1446.

J. R. Quinlan. 1993. *C4.5: Programs for Machine Learning*. Morgan Kaufmann.

P. Samarati. 2001. Protecting Respondents' Identities in Microdata Release. *IEEE Transactions on Knowledge and Data Engineering* 13, 6 (2001), 1010–1027.

L. Sweeney. 2002. K-anonymity: A Model for Protecting Privacy. *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems* 10, 5 (2002), 557–570.

S. M. Weiss and C. A. Kulikowski. 1991. *Computer Systems That Learn: Classification and Prediction Methods from Statistics, Neural Nets, Machine Learning, and Expert Systems*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA.

R. C.-W. Wong, A. W.-C. Fu, K. Wang, and J. Pei. 2007. Minimality Attack In Privacy Preserving Data Publishing. In *Proceedings of the 33rd international conference on Very large data bases (VLDB '07)*. 543–554.

X. Xiao, G. Bender, M. Hay, and J. Gehrke. 2011a. iReduct: Differential Privacy with Reduced Relative Errors. In *Proceedings of the 2011 ACM SIGMOD International Conference on Management of Data (SIGMOD '11)*. 229–240.

X. Xiao, G. Wang, and J. Gehrke. 2011b. Differential Privacy Via Wavelet Transforms. *IEEE Transactions on Knowledge and Data Engineering* 23, 8 (2011), 1200–1214.

Y. Xiao, L. Xiong, L. Fan, S. Goryczka, and H. Li. 2014. DPCube: Differentially Private Histogram Release through Multidimensional Partitioning. *Transactions on Data Privacy* 7, 3 (2014), 195–222.

J. Xu, W. Wang, J. Pei, X. Wang, B. Shi, and A. W. C. Fu. 2006. Utility-Based Anonymization Using Local Recoding. In *Proceedings of the 12th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD '06)*. 785–790.

J. Xu, Z. Zhang, X. Xiao, Y. Yang, G. Yu, and M. Winslett. 2013. Differentially Private Histogram Publication. *The VLDB Journal* 22, 6 (2013), 797–822.

J. Zhang, G. Cormode, C. M. Procopiuc, D. Srivastava, and X. Xiao. 2014. PrivBayes: Private Data Release via Bayesian Networks. In *Proceedings of the 2014 ACM SIGMOD International Conference on Management of Data (SIGMOD '14)*. 1423–1434.