Personalized Truncation for Personalized Privacy

DAJUN SUN, Hong Kong University of Science and Technology, China
WEI DONG*, Nanyang Technological University, Singapore
YUAN QIU, Hong Kong University of Science and Technology, China
KE YI, Hong Kong University of Science and Technology, China

In the standard model of differential privacy (DP), every user's privacy is treated equally, which is captured by a single privacy parameter ε. However, in many real-world situations, users may have diverse privacy concerns and requirements, some conservative while others liberal. This is formalized by the model of *personalized differential privacy (PDP)*, where each user may have a different privacy parameter ε. However, existing techniques for PDP cannot provide good utility for many fundamental problems such as basic counting and sum estimation. In this paper, we present the *personalized truncation mechanism* for these problems under PDP. We first show that, theoretically, it is never worse than previous mechanisms (up to polylogarithmic factors) on any instance, while can be much better in certain cases. Then we use extensive experiments on both real and synthetic data to demonstrate its empirical advantages. Our mechanism also works for user-level DP, thus supporting a large class of SJA queries over relational databases under foreign-key constraints.

CCS Concepts: • **Security and privacy → Database and storage security**; • **Information systems →** *Data management systems.*

Additional Key Words and Phrases: Differential privacy; SJA query processing

ACM Reference Format:
Dajun Sun, Wei Dong, Yuan Qiu, and Ke Yi. 2024. Personalized Truncation for Personalized Privacy. *Proc. ACM Manag. Data* 2, 6 (SIGMOD), Article 249 (December 2024), 25 pages. https://doi.org/10.1145/3698825

1 INTRODUCTION

Sum estimation is a fundamental problem under *differential privacy (DP)* and has wide application in many areas such as query answering [21], statistical analysis [16] and machine learning [1]. Given a database instance $\mathbf{I}$, in which each user u possesses an integer value $\mathbf{I}(u) \in [B] := \{0, 1, \ldots, B\}$, the goal is to produce a privatized estimate for $\mathrm{Sum}(\mathbf{I}) := \sum_u \mathbf{I}(u)$. Because the *global sensitivity* of $\mathrm{Sum}(\cdot)$ is B, the classical *Laplace mechanism* must add a noise of scale B/ε, where ε is the *privacy parameter* (all these technical terms will be defined more precisely in Section 3). When the domain size B is large, a noise of such a scale would dwarf the true sum.

*Wei Dong is the corresponding author.

Authors' addresses: Dajun Sun, Hong Kong University of Science and Technology, Hong Kong, China, dsunad@cse.ust.hk; Wei Dong, Nanyang Technological University, Singapore, Singapore, wei_dong@ntu.edu.sg; Yuan Qiu, Hong Kong University of Science and Technology, Hong Kong, China, yqiuac@cse.ust.hk; Ke Yi, Hong Kong University of Science and Technology, Hong Kong, China, yike@cse.ust.hk.

ACM 2836-6573/2024/12-ART249
https://doi.org/10.1145/3698825

The truncation mechanism. A simple yet effective approach towards functions with large global sensitivity is the *truncation mechanism* [10, 12, 20, 23, 24]. For some truncation threshold $\tau \in [B]$, define the truncated instance $\mathbf{I}_\tau$ such that $\mathbf{I}_\tau(u) = \min\{\mathbf{I}(u), \tau\}$ for all u. Since $\mathbf{I}_\tau$ has global sensitivity τ, one can invoke the Laplace mechanism on $\mathrm{Sum}(\mathbf{I}_\tau)$ with a noise scale of τ/ε, i.e., the privatized sum estimate is

$$\widetilde{\mathrm{Sum}}(\mathbf{I}) = \mathrm{Sum}(\mathbf{I}_\tau) + \tau/\varepsilon \cdot \mathrm{Lap}(1),$$

where $\mathrm{Lap}(1)$ denotes a unit Laplace noise.

The key issue for the truncation mechanism is thus how to choose the truncation threshold τ that minimizes the error

$$\mathrm{Error}(\mathbf{I}, \tau) := \mathrm{Sum}(\mathbf{I}) - \mathrm{Sum}(\mathbf{I}_\tau) + \tau/\varepsilon, \tag{1}$$

where $\mathrm{Sum}(\mathbf{I}) - \mathrm{Sum}(\mathbf{I}_\tau)$ is the bias introduced by the truncation and τ/ε is the (square root of) variance of the noise. Clearly, as τ gets larger, the bias reduces while the variance increases. Thus, one would try to optimize this bias-variance trade-off by selecting the best τ. However, directly using $\tau = \arg\min_\tau \mathrm{Error}(\mathbf{I}, \tau)$ violates DP, so the main technical challenge for the truncation mechanism to perform well lies in finding a near-optimal τ in a DP fashion. Notably, the R2T mechanism [10] finds a privatized $\tilde{\tau}$ such that[1]

$$\mathrm{Error}(\mathbf{I}, \tilde{\tau}) \le \widetilde{O}(1) \cdot \min_\tau \mathrm{Error}(\mathbf{I}, \tau) \tag{2}$$

on every instance $\mathbf{I}$. Where $\widetilde{O}(1)$ hides poly-logarithm factors in B, which can be considered as the "search cost" of privately finding the optimal τ in the value domain $[B]$.

Personalized differential privacy. The standard DP model provides uniform privacy protection for all users, which may not be desirable in real applications. For instance, companies may be interested in purchasing users' data to improve their products. In this case, users with lower privacy concerns may indicate weaker privacy preferences, provide more accurate information, and earn more from data sales. In contrast, privacy-conscious users may signal higher privacy levels to avoid revealing sensitive information, and correspondingly earn less. In this paper, we are interested in the case where different users may have different privacy requirements, which is known as *personalized differential privacy (PDP)* in the literature [3, 17, 22]. In this model, each user u may have a different privacy parameter. This is captured by a function $\Phi : \mathcal{U} \to \mathbb{R}^+$, where $\mathcal{U}$ denotes the possibly infinite user universe. It is required that the output distributions should be $\Phi(u)$-indistinguishable on any two neighboring instances that differ in u's data (the formal definition will be given in Section 3).

Let $\varepsilon_{\min} := \min_{u \in \mathcal{U}} \Phi(u)$ and $\varepsilon_{\max} := \max_{u \in \mathcal{U}} \Phi(u)$. One naive way to adapt the truncation mechanism (actually, any DP mechanism) to PDP is to invoke it with privacy parameter $\varepsilon_{\min}$, respecting the privacy of the most conservative user, hence all users. Then, the optimal truncation error is

$$\min_\tau \mathrm{Error}(\mathbf{I}, \tau) = \min_\tau \left(\mathrm{Sum}(\mathbf{I}) - \mathrm{Sum}(\mathbf{I}_\tau) + \tau/\varepsilon_{\min}\right), \tag{3}$$

and R2T can reach it within some logarithmic factors. However, this error can be excessively large. Note that $\varepsilon_{\min}$ is the smallest $\Phi(u)$ over all users in the user universe $\mathcal{U}$, not just those in the given instance $\mathbf{I}$. Technically, this is because the privacy parameter must not depend on the instance. A more intuitive explanation is that, for a user u who is not in the given instance $\mathbf{I}$, we must still guard against the neighboring instance $\mathbf{I}' = \mathbf{I} \cup \{u\text{'s data}\}$, i.e., the adversary should not know that u is *not* in $\mathbf{I}$. Thus, as long as there is one conservative user in the user universe, no matter whether their data is included in $\mathbf{I}$ or not, the error in (3) would be very large, even using the optimal truncation threshold τ.

[1] In the introduction, all the error guarantees hold with constant probability.

Personalized truncation for personalized privacy. We see that when $\varepsilon_{\min}$ is small, we must use a small τ to reduce the noise term $\tau/\varepsilon_{\min}$, which inevitably increases the bias. Critically, this bias-variance trade-off is hard to balance because we have only one knob, a single truncation threshold τ, to turn. To get out of the dilemma, in this paper we propose the *personalized truncation mechanism*. We generalize the truncation threshold τ from one single value to a function $\boldsymbol{\tau} : \mathcal{U} \to [B]$, or equivalently, a $|\mathcal{U}|$-dimensional vector $\boldsymbol{\tau} \in [B]^{\mathcal{U}}$. Note that we use $\boldsymbol{\tau}$ in bold to represent a function/vector and the normal letter τ for a single value. Then we truncate u's data to $\boldsymbol{\tau}(u)$, i.e., in the truncated instance $\mathbf{I}_{\boldsymbol{\tau}}$, user u's value is set to

$$\mathbf{I}_{\boldsymbol{\tau}}(u) := \min\{\mathbf{I}(u), \boldsymbol{\tau}(u)\}.$$

The observation is that, after truncation, we need a noise scale of $\boldsymbol{\tau}(u)/\Phi(u)$ to protect the data of u, so a noise scale of $\max_u(\boldsymbol{\tau}(u)/\Phi(u))$ suffices for the privacy of all users. Thus, for the personalized truncation mechanism, the optimal error becomes

$$\min_{\boldsymbol{\tau}} \text{Error}(\mathbf{I}, \boldsymbol{\tau}) = \min_{\boldsymbol{\tau}} \left(\text{Sum}(\mathbf{I}) - \text{Sum}(\mathbf{I}_{\boldsymbol{\tau}}) + \max_u \frac{\boldsymbol{\tau}(u)}{\Phi(u)} \right). \tag{4}$$

It is easy to see that (4) is always no worse than (3) since the standard truncation is a special case of personalized truncation with $\boldsymbol{\tau}(\cdot) \equiv \tau$. On the other hand, (4) can be significantly better than (3) as we can set a small $\boldsymbol{\tau}(u)$ for users with a small $\Phi(u)$. Example 1.1 shows that the gap in the error can be large.

Example 1.1. Suppose $\mathcal{U} = \{u_1, \ldots, u_n\}$, and their privacy parameters are $\Phi(u_1) = 1/n$, $\Phi(u_i) = \sqrt{i/n}$ for $i = 2, \ldots, n$. Consider an instance $\mathbf{I}$ in which $\mathbf{I}(u_i) = \sqrt{i}$ for $i = 1, \ldots, n$. Using a single truncation threshold τ, the error in (3) is $\Omega(n^{1.5})$. On the other hand, the optimal personalized truncation would set $\boldsymbol{\tau}(u_1) = 0$ and $\boldsymbol{\tau}(u_i) = \sqrt{i}$ for $i = 2, \ldots, n$, which leads to an error of $O(\sqrt{n})$. □

Our results. As in the standard truncation mechanism, the main technical challenge for personalized truncation is how to find a truncation threshold function $\boldsymbol{\tau}$ in a DP fashion while approaching the optimal error in (4). The issue is actually more challenging here because we face a much larger, even infinite, search space of size $B^{|\mathcal{U}|}$. Resolving this difficulty, the main technical contribution of this paper is a PDP mechanism that implicitly finds a $\boldsymbol{\tau}$ to achieve an error of

$$\widetilde{O}(1) \cdot \min_{\boldsymbol{\tau}} \text{Error}(\mathbf{I}, \boldsymbol{\tau}), \tag{5}$$

where $\widetilde{O}(1)$ hides poly-logarithm factors in $\frac{\varepsilon_{\max}}{\varepsilon_{\min}}$ and in B. When $\Phi(\cdot) \equiv \varepsilon$, we have $\varepsilon_{\max} = \varepsilon_{\min}$ and (5) degenerates into (2), so our mechanism subsumes R2T in the standard DP model. On the other hand, under PDP, (5) can be significantly smaller than (2) as illustrated in Example 1.1.

We also extend our PDP mechanism to support user-level privacy in relational databases [10, 11, 18, 24]. In this model, to be defined formally in Section 3, a user may own an arbitrary number of values, either by themselves or jointly with other users. This user-level DP model is equivalent to the DP model for relational databases under foreign-key constraints, hence capturing a large class of select-join-aggregate queries (possibly with self-joins). In particular, private graph data under node-DP [6, 8, 23] is an instantiation of this model, where an edge is jointly owned by its two endpoints. In the PDP version, each user u may have a different privacy parameter $\Phi(u)$, and it is required that any two neighboring instances that differ only in the data owned (solely or jointly) by u are $\Phi(u)$-indistinguishable. Note that in the graph setting, we get two such neighboring instances if we add/remove a node u and all its incident edges, which exactly corresponds to the privacy definition of node-DP.

To demonstrate the utility and efficiency of our mechanisms, we conduct a series of experiments in Section 8, where we compare our approaches with previous ones. The experimental results show the error of our methods is an order of magnitude smaller than previous approaches, with roughly the same computational cost.

2 RELATED WORK

Providing personalized privacy protection is a well motivated problem due to the diverse backgrounds of users and/or privacy requirements. In fact, this issue was studied even before differential privacy became the main-stream privacy model. For example, Xiao and Tao [35] defined a notion of personalized privacy in the context of k-anonymity, which was later extended to other related privacy models [19, 32, 37, 37]. However, these models do not provide privacy protection as rigorous as differential privacy [27, 39], so they have gradually faded over time.

The model of personalized differential privacy (also known as heterogeneous differential privacy) was proposed by Jorgensen et. al. [22], where they also designed the sampling mechanism and extended the inverse sensitivity based exponential mechanism [4, 9, 28] to PDP. The latter is further improved in [31]. There are some other works [3, 26] that study some more complicated problems, such as support vector machine and linear regression. It would be interesting to see if our results on sum estimation can be used to bring improvements to these problems as well. [17] studies how to track the privacy consumption of each user over multiple queries. A user's data is discarded when the cumulative privacy consumption exceeds that user's privacy preference.

All works mentioned above are under the central setting, where we assume there is a trusted central data curator that holds and analyzes users' data. The PDP definition easily extends to the local model [5, 7, 36, 38], where each user sends a privatized message to an untrusted data analyzer. In doing so, each user may directly use a different privacy parameter when privatizing their data. The (standard) truncation mechanism has been studied in the local DP model [20], and it would be interesting to see how the personalized truncation mechanism would work in the local PDP model.

There are also some other interesting papers that do not study the PDP model directly, but nevertheless have a "personalized" flavor. For example, [29, 34] consider a personalized privacy setting where each user determines which part of its data is public or private, and then provide (standard) DP protection only on the private part. The recent work by Zhang et. al. [40] points to another interesting direction called multi-analyst DP. They consider the case where the private dataset is analyzed by different parties with different privacy privileges and analyze how to answer as many queries as possible while maintaining high accuracy. Their model differs from our PDP model in the sense that their variety in privacy comes from the analyzer, not the users.

3 PRELIMINARIES

3.1 The Privacy Model

Our model of personalized user-level differential privacy combines the features of [10, 18, 24] and [22]. Let $\mathcal{U}$ be the universe of all users, which is possibly infinite. A database instance $\mathbf{I}$ is defined as a mapping $\mathbf{I} : 2^{\mathcal{U}} \to [B]$, namely, every subset $\boldsymbol{u}$ of users jointly own the value $\mathbf{I}(\boldsymbol{u})$. For notational clarity, we use $\boldsymbol{u}$ in bold to represent a subset of users and the normal letter u for a single user. If some $\boldsymbol{u}$ owns no data in $\mathbf{I}$, we set $\mathbf{I}(\boldsymbol{u}) = 0$, and such zero mappings do not need to be represented explicitly. Specially, we require $\mathbf{I}(\emptyset) = 0$. Let n be the number of nonzero values in $\mathbf{I}$. The goal is to estimate $\mathrm{Sum}(\mathbf{I}) = \sum_{\boldsymbol{u}} \mathbf{I}(\boldsymbol{u})$. Note that it is without loss of generality to assume every subset $\boldsymbol{u}$ of users own a single value $\mathbf{I}(\boldsymbol{u})$. If they own multiple values, we can simply define $\mathbf{I}(\boldsymbol{u})$ to be their sum. On the other hand, if a user u owns multiple values but jointly with different other users, then they cannot be combined, which would change the neighboring relationship, defined as follows.

For two instances $\mathbf{I}$ and $\mathbf{I}'$, we say that $\mathbf{I}$ contains $\mathbf{I}'$, denoted $\mathbf{I}' \preceq \mathbf{I}$, if $\mathbf{I}'(\boldsymbol{u}) \neq 0 \rightarrow \mathbf{I}(\boldsymbol{u}) = \mathbf{I}'(\boldsymbol{u})$ for every $\boldsymbol{u}$, i.e., $\mathbf{I}$ contains all the nonzero mappings of $\mathbf{I}'$. We say that $\mathbf{I}'$ is a neighbor of $\mathbf{I}$ that differs on user u, denoted as $\mathbf{I} \overset{u}{\sim} \mathbf{I}'$, if $\mathbf{I}' \preceq \mathbf{I}$ or $\mathbf{I} \preceq \mathbf{I}'$, and all the differences between $\mathbf{I}$ and $\mathbf{I}'$ are owned (solely or jointly) by u, i.e., for every $\boldsymbol{u}$, $\mathbf{I}'(\boldsymbol{u}) \neq \mathbf{I}(\boldsymbol{u}) \rightarrow u \in \boldsymbol{u}$.

Plugging this neighboring relationship into the standard DP definition [13, 14] yields user-level DP [18]:

Definition 3.1 (User-level differential privacy). *For $\varepsilon > 0$, a mechanism $\mathcal{M}$ is ε-differentially private, or simply ε-DP, if for any $u \in \mathcal{U}$, any two neighboring instances $\mathbf{I} \overset{u}{\sim} \mathbf{I}'$ and any y,*

$$\Pr[\mathcal{M}(\mathbf{I}) = y] \leq e^{\varepsilon} \cdot \Pr[\mathcal{M}(\mathbf{I}') = y].$$

The neighboring relationship defined above is known as the add/delete-one policy, i.e., $\mathbf{I}'$ is obtained from $\mathbf{I}$ by adding or deleting any number of values, all of which are owned by some user u (possibly jointly with some other users). Another commonly used policy is the change-one policy, which allows one to change these values. Because all these changes can be done by first deleting the old values and then adding the new values, any ε-DP mechanism under the add/delete-one policy is a 2ε-DP mechanism under the change-one policy, so the difference is at most a factor of 2.

This user-level DP model captures many interesting problems. When every $\boldsymbol{u}$ with $\mathbf{I}(\boldsymbol{u}) \neq 0$ has $|\boldsymbol{u}| = 1$, it degenerates into the simple "one value per user" case mentioned at the beginning of the paper, which is also known as tuple-level DP. The $|\boldsymbol{u}| > 1$ case allows us to model more complicated relationships between users and values. For example, to count the number of edges of a graph under node-DP, we set $\mathbf{I}(\{u, v\}) = 1$ if there is an edge between u, v, and set all other $\mathbf{I}(\boldsymbol{u})$ to 0 (ditto). The triangle counting problem under node-DP (resp. edge-DP) can be captured by setting $\mathbf{I}(\{u, v, w\}) = 1$ if nodes u, v, w form a triangle (resp. $\mathbf{I}(\{e_1, e_2, e_3\}) = 1$ if edges e_1, e_2, e_3 form a triangle). In the TPC-H benchmark data, suppose we would like to find the total sales quantity while respecting the privacy of every customer and supplier, we can set $\mathbf{I}(\{c, s\})$ as the sales quantity from supplier s to customer c. In fact, it has been shown [18] that this user-level DP model is equivalent to the DP model for relational databases under foreign-key constraints [10, 24], hence capturing a large class of SQL queries that are comprised of joins, selections, and aggregations. Please see [10, 18] for the precise definition of the class of queries supported.

To model the situation where different users may have different privacy requirements, we follow [22] and introduce a *privacy specification*, i.e., a function $\Phi : \mathcal{U} \rightarrow \mathbb{R}^+$, where $\Phi(u)$ specifies the privacy parameter of u. This leads to the following definition:

Definition 3.2 (Personalized Differential Privacy (PDP)). *Given a privacy specification Φ, a mechanism $\mathcal{M}$ satisfies Φ-personalized differential privacy, or simply Φ-PDP, if for every pair of neighboring datasets $\mathbf{I} \overset{u}{\sim} \mathbf{I}'$ and any y,*

$$\Pr[\mathcal{M}(\mathbf{I}) = y] \leq e^{\Phi(u)} \cdot \Pr[\mathcal{M}(\mathbf{I}') = y].$$

Note that if $\mathbf{I}$ and $\mathbf{I}'$ differ by only one value, say, jointly owned by u and v, then they are neighbors on both u and v. In this case, Definition 3.2 implicitly requires that the output distributions are $\min\{\Phi(u), \Phi(v)\}$-indistinguishable, thus protecting the privacy of both.

Let $\varepsilon_{\min} = \min_u \Phi(u)$, $\varepsilon_{\max} = \max_u \Phi(u)$. We assume $\Omega(\frac{1}{n}) \leq \varepsilon_{\min} \leq \varepsilon_{\max} \leq O(1)$. This is the most common parameter regime: A user with $\Phi(u) \leq \frac{1}{n}$ would introduce at least $\Omega(n)$ noise, so such users can be discarded upfront. On the other hand, a privacy parameter larger than a constant is considered too weak.

3.2 DP and PDP Basics

For any query Q, its *global sensitivity* is

$$\mathrm{GS}_Q(\mathbf{I}) = \sup_{\mathbf{I},\mathbf{I}':\mathbf{I}\sim\mathbf{I}'} |Q(\mathbf{I}) - Q(\mathbf{I}')|.$$

When the query Q is clear from the context, we omit the subscript Q and simply write GS.

One standard technique for achieving DP is to add a Laplace noise with scale GS/ε before releasing $Q(\mathbf{I})$:

Lemma 3.1 (Laplace Mechanism [15]). *The mechanism*

$$\mathcal{M}(\mathbf{I}) := Q(\mathbf{I}) + \frac{\mathrm{GS}}{\varepsilon} \cdot \mathrm{Lap}(1)$$

preserves ε-DP, where $\mathrm{Lap}(1)$ denotes a random variable drawn from the unit Laplace distribution.

The following tail bound implies that the error of the Laplace mechanism is $O(\mathrm{GS}/\varepsilon \cdot \log(1/\beta))$ with probability at least $1-\beta$:

Lemma 3.2 (Laplace tail bound). *If $X \sim \mathrm{Lap}(1)$, then $\Pr(|X| > \ln \frac{1}{\beta}) \le \beta$ for any $0 < \beta < 1$.*

A naive way to achieve PDP is to invoke a DP mechanism with $\varepsilon = \varepsilon_{\min}$. In the case of the Laplace mechanism, this naive PDP mechanism would add a Laplace noise of scale $\mathrm{GS}/\varepsilon_{\min}$ to $Q(\mathbf{I})$.

A potentially better PDP mechanism is the *sampling mechanism* [22]. While [22] only considered the $\ell = 1$ case (i.e., tuple-level PDP), the mechanism can be easily generated to user-level PDP, as follows. For some given ε, we sample each user $u \in \mathcal{U}$ with probability $\min\left\{\frac{e^{\Phi(u)}-1}{e^{\varepsilon}-1}, 1\right\}$. Note that all users with $\Phi(u) \ge \varepsilon$ are always sampled. Then define the sampled instance $S(\mathbf{I}, \varepsilon)$ such that

$$S(\mathbf{I}, \varepsilon)(\boldsymbol{u}) = \begin{cases} \mathbf{I}(\boldsymbol{u}), & \text{if for all } u \in \boldsymbol{u}, u \text{ is sampled;} \\ 0, & \text{otherwise.} \end{cases}$$

Lemma 3.3 (The Sampling Mechanism). *Let $\mathcal{M}_\varepsilon(\cdot)$ be an ε-DP mechanism. Then for any Φ, the mechanism $\mathcal{M}_\varepsilon(S(\cdot, \varepsilon))$ preserves Φ-PDP.*

Proof. See Appendix A in the full version [33]. □

The utility of the sampling mechanism critically depends on the sampling threshold ε. As a heuristic, [22] suggests to set ε to the average of $\Phi(u), u \in \mathcal{U}$. However, as shall be seen later, this is often not a good choice. In fact, the optimal ε depends on the instance $\mathbf{I}$, so we need to find it privately.

PDP also inherits the important composition property from DP:

Lemma 3.4 (Composition [14, 22]). *Given $\Phi_1, ..., \Phi_k$, let $\mathcal{M}$ be an adaptive composition of $\mathcal{M}_1$,..., $\mathcal{M}_k$, where each $\mathcal{M}_i$ is Φ_i-PDP, then $\mathcal{M}$ is Φ-PDP where $\Phi(u) = \sum_{i=1}^{k} \Phi_i(u)$.*

4 BASIC COUNTING

We see that the main technical challenge is to privately search through a very large space $[B]^{\mathcal{U}}$ of possible personalized truncation thresholds. To build up our solution, in this section we start with the simple case of $B = 1$ under tuple-level PDP, i.e., each user u just owns a bit $\mathbf{I}(u) \in \{0, 1\}$, and the goal is to count the number of 1's. In this case, the truncation threshold $\boldsymbol{\tau}$ is a $|\mathcal{U}|$-dimensional bit vector. Without loss of generality, assume $\Phi(u_1) \le \Phi(u_2) \le \cdots \le \Phi(u_{|\mathcal{U}|})$. The crucial observation that greatly reduces the search space is that it suffices to consider $\boldsymbol{\tau}$ in the form of $(0, \ldots, 0, 1, \ldots, 1)$. Equivalently, for some $\varepsilon = \Phi(u_i)$, we truncate the data to 0 for $u_1, \ldots, u_{i-1}$ while leaving $u_i, \ldots, u_{|\mathcal{U}|}$

untouched. To see why, consider an arbitrary $\boldsymbol{\tau}$ and let i be the smallest index such that $\boldsymbol{\tau}(u_i) = 1$. Recall that the error of the personalized truncation mechanism using $\boldsymbol{\tau}$ is

$$\text{Error}(\mathbf{I}, \boldsymbol{\tau}) = \text{Sum}(\mathbf{I}) - \text{Sum}(\mathbf{I}_{\boldsymbol{\tau}}) + \max_u \frac{\boldsymbol{\tau}(u)}{\Phi(u)}.$$

Then the noise term is $1/\Phi(u_i)$. Suppose there is some $j > i$ with $\boldsymbol{\tau}(u_j) = 0$. Observe that it would not make things worse if we changed it to $\boldsymbol{\tau}(u_j) = 1$: The noise term is not affected since $\Phi(u_i) \le \Phi(u_j)$, while the truncation bias cannot become larger when we use a higher truncation threshold. After doing so for every such j, we can convert $\boldsymbol{\tau}$ into the form of $(0, \ldots, 0, 1, \ldots, 1)$ without enlarging the error.

Therefore, for the basic counting problem, the personalized truncation mechanism simplifies to the following: For a privacy threshold ε, define the truncated instance $\bar{\mathbf{I}}_\varepsilon$ such that

$$\bar{\mathbf{I}}_\varepsilon(u) = \begin{cases} \mathbf{I}(u), & \text{if } \Phi(u) \ge \varepsilon; \\ 0, & \text{otherwise.} \end{cases}$$

The mechanism returns

$$\widetilde{\text{Sum}}\left(\bar{\mathbf{I}}_\varepsilon\right) = \text{Sum}\left(\bar{\mathbf{I}}_\varepsilon\right) + 1/\varepsilon \cdot \text{Lap}(1).$$

The error also simplifies to

$$\text{Error}(\mathbf{I}, \varepsilon) = \text{Sum}(\mathbf{I}) - \text{Sum}(\bar{\mathbf{I}}_\varepsilon) + 1/\varepsilon. \tag{6}$$

Thus, the remaining technical problem is to optimize (6). However, the optimal ε^* depends on the instance $\mathbf{I}$, as illustrated in the following example:

Example 4.1. Suppose there are n users, with privacy parameters $\Phi(u_1) = \cdots = \Phi(u_{n/2}) = \frac{1}{\sqrt{n}}, \Phi(u_{n/2+1}) = \ldots = \Phi(u_n) = 1$. Consider the instance $\mathbf{I}$ in which all of them have bit 1. Then the optimal threshold is $\varepsilon^* = \frac{1}{\sqrt{n}}$, which leads to an error of $O(\sqrt{n})$. On the other hand, if we change the bits of $u_1, \ldots, u_{n/2}$ to 0, then the optimal threshold is $\varepsilon^* = 1$, which has an error of $O(1)$. Moreover, as we change the bits of the first $n/2$ users from 1 to 0 one by one, there must be a time at which the optimal ε^* suddenly changes from $\frac{1}{\sqrt{n}}$ to 1, so it is highly sensitive to $\mathbf{I}$. Thus, using ε^* directly breaks privacy. □

To find a near-optimal ε^* while respecting privacy, one may attempt to minimize a privatized (6). However, (6) has the term $\text{Sum}(\mathbf{I})$, and privatizing it is exactly the problem we wanted to solve in the first place. To get out of this circular dependency, we borrow the idea from R2T [10]. We try doubling values of $\varepsilon = \varepsilon_{\min}, 2\varepsilon_{\min}, \ldots, 2^{\lceil \log \frac{\varepsilon_{\max}}{\varepsilon_{\min}} \rceil}\varepsilon_{\min}$. For each ε, we compute $\widetilde{\text{Sum}}\left(\bar{\mathbf{I}}_\varepsilon\right)$, but subtract by a term such that it is an underestimate of the true sum with high probability. In doing so, we must also divide the privacy budget to these $\lceil \log \frac{\varepsilon_{\max}}{\varepsilon_{\min}} \rceil$ privatized estimates. Finally, we return the maximum of these estimates as the final output. The details are given in Algorithm 1.

We prove its privacy and utility in the following theorem:

Theorem 4.2. *For any Φ, β, Algorithm 4.2 satisfies Φ-PDP. Additionally, for every $\mathbf{I}$, the error of Algorithm 1 is at most*

$$\min_\varepsilon \left(\left| \text{Sum}(\bar{\mathbf{I}}_\varepsilon) - \text{Sum}(\mathbf{I}) \right| + \frac{4\lceil \log \frac{\varepsilon_{\max}}{\varepsilon_{\min}} \rceil}{\varepsilon} \ln \frac{\lceil \log \frac{\varepsilon_{\max}}{\varepsilon_{\min}} \rceil}{\beta} \right) \tag{7}$$

with probability at least $1 - \beta$.

Algorithm 1: PDP Count

Input: $\mathbf{I}, \Phi, \beta$

$t = \lceil \log \frac{\varepsilon_{\max}}{\varepsilon_{\min}} \rceil$;

for $i \leftarrow 1, 2, \ldots, t$ **do**

  $\varepsilon_i = 2^{i-1}\varepsilon_{\min}$;

  Define $\bar{\mathbf{I}}_{\varepsilon_i}$ such that $\bar{\mathbf{I}}_{\varepsilon_i}(u) = \begin{cases} \mathbf{I}(u), & \text{if } \Phi(u) \geq \varepsilon_i \\ 0, & \text{if otherwise} \end{cases}$

  $\widetilde{\mathrm{Sum}_i}(\mathbf{I}) = \mathrm{Sum}\left(\bar{\mathbf{I}}_{\varepsilon_i}\right) + \mathrm{Lap}\left(\frac{t}{\varepsilon_i}\right) - \frac{t}{\varepsilon_i} \ln \frac{t}{\beta}$;

end

return $\widetilde{\mathrm{Sum}}(\mathbf{I}) = \max_i \widetilde{\mathrm{Sum}_i}(\mathbf{I})$;

Proof. We first prove its privacy.

Let $t = \lceil \log \frac{\varepsilon_{\max}}{\varepsilon_{\min}} \rceil$. We'll show each iteration of the for-loop is $\frac{\Phi(\cdot)}{t}$-PDP. In the ith iteration where the threshold is ε_i, consider $\mathbf{I} \overset{u}{\sim} \mathbf{I}'$ for arbitrary u. If $\Phi(u) < \varepsilon_i$, then u will not appear in $\bar{\mathbf{I}}_{\varepsilon_i}$ nor $\bar{\mathbf{I}}'_{\varepsilon_i}$ thus for any y

$$\Pr\left(\widetilde{\mathrm{Sum}_i}(\mathbf{I}) = y\right) = \Pr\left(\widetilde{\mathrm{Sum}_i}(\mathbf{I}') = y\right).$$

On the other hand, if $\Phi(u) \geq \varepsilon_i$, then $\bar{\mathbf{I}}_{\varepsilon_i} \overset{u}{\sim} \bar{\mathbf{I}}'_{\varepsilon_i}$. Since the global sensitivity of $\mathrm{Sum}\left(\bar{\mathbf{I}}_{\varepsilon_i}\right)$ is 1, according to lemma 3.1, adding a Laplace noise with scale $\frac{t}{\varepsilon_i}$ preserves $\frac{t}{\varepsilon_i}$-DP, namely for any y we have:

$$\Pr\left(\widetilde{\mathrm{Sum}_i}(\mathbf{I}) = y\right) \leq e^{\frac{\varepsilon_i}{t}} \Pr\left(\widetilde{\mathrm{Sum}_i}(\mathbf{I}') = y\right) \leq e^{\frac{\Phi(u)}{t}} \Pr\left(\widetilde{\mathrm{Sum}_i}(\mathbf{I}') = y\right)$$

Since this holds for any u, according to the definition of PDP, this iteration will be $\frac{\Phi(\cdot)}{t}$-PDP. Then the whole process will be Φ-PDP according to basic composition.

Then we prove its utility. First we should note, according to the Laplace tail bound in lemma 3.2, with probability at least $1 - \frac{\beta}{t}$, for any i we have $\mathrm{Lap}\left(\frac{t}{\varepsilon_i}\right) - \frac{t}{\varepsilon_i} \ln \frac{t}{\beta} \leq 0$. Then according to a union bound, with probability at least $1 - \beta$, the above inequality holds for all i. That is, $\widetilde{\mathrm{Sum}_i}(\mathbf{I}) \leq \mathrm{Sum}\left(\bar{\mathbf{I}}_{\varepsilon_i}\right) \leq \mathrm{Sum}(\mathbf{I})$ for all i with high probability.

For any given $\varepsilon \in [\varepsilon_{\min}, \varepsilon_{\max}]$, assume the ε_i is the largest value we examine that is no greater than ε, obviously we have $\varepsilon_i \leq \varepsilon < 2\varepsilon_i$. Then we know $\bar{\mathbf{I}}_{\varepsilon} \subset \bar{\mathbf{I}}_{\varepsilon_i}$. Since Sum() is monotone, we have

$$|\mathrm{Sum}(\bar{\mathbf{I}}_{\varepsilon_i}) - \mathrm{Sum}(\mathbf{I})| \leq |\mathrm{Sum}(\bar{\mathbf{I}}_{\varepsilon}) - \mathrm{Sum}(\mathbf{I})|.$$

Which means the bias at ε_i will be no greater than ε. Thus with probability at least $1 - \beta$ we have:

$$\begin{aligned} |\widetilde{\mathrm{Sum}}(\mathbf{I}) - \mathrm{Sum}(\mathbf{I})| &\leq |\widetilde{\mathrm{Sum}_i}(\mathbf{I}) - \mathrm{Sum}(\mathbf{I})| \\ &= |\mathrm{Sum}\left(\bar{\mathbf{I}}_{\varepsilon_i}\right) + \mathrm{Lap}\left(\frac{t}{\varepsilon_i}\right) - \frac{t}{\varepsilon_i} \ln \frac{t}{\beta} - \mathrm{Sum}(\mathbf{I})| \\ &\leq |\mathrm{Sum}(\bar{\mathbf{I}}_{\varepsilon_i}) - \mathrm{Sum}(\mathbf{I})| + \frac{2t}{\varepsilon_i} \ln \frac{t}{\beta} \\ &\leq |\mathrm{Sum}(\bar{\mathbf{I}}_{\varepsilon}) - \mathrm{Sum}(\mathbf{I})| + \frac{4t}{\varepsilon} \ln \frac{t}{\beta} \end{aligned}$$

Here the first line is because $\widetilde{\mathrm{Sum}_i}(\mathbf{I})$ is always no greater than $\mathrm{Sum}(\mathbf{I})$ and the third line is due to Laplace tail bound. Since the above inequality holds for any $\varepsilon \in [\varepsilon_{\min}, \varepsilon_{\max}]$, the actual error of Algorithm 1 should be no greater than the minimum of them. □

As argued earlier, it suffices to consider $\boldsymbol{\tau}$ in the form of $(0, \ldots, 0, 1, \ldots, 1)$, so $\min_{\boldsymbol{\tau}} \text{Error}(\mathbf{I}, \boldsymbol{\tau}) = \min_{\varepsilon} \text{Error}(\mathbf{I}, \varepsilon)$. Thus, (7) implies the claimed error bound of (5) with $B = 1$ and β a constant, namely, Algorithm 1 achieves the optimal personalized truncation error within logarithmic factors. It can also be implemented efficiently: Let n be the number of users with bit 1; the other users are ignored. We partition these n users into $\lceil \log \frac{\varepsilon_{\max}}{\varepsilon_{\min}} \rceil$ buckets by the value of $\lceil \log \frac{\Phi(u)}{\varepsilon_{\min}} \rceil$. Then we scan these buckets in ascending order and compute $\text{Sum}\left(\bar{\mathbf{I}}_{\varepsilon_i}\right)$ for all $i \in \{1, 2, \ldots \lceil \log \frac{\varepsilon_{\max}}{\varepsilon_{\min}} \rceil\}$. Thus the total running time is $O(n)$.

Compare with the sampling mechanism. Astute readers would observe that, for the basic counting problem, the personalized truncation mechanism is very similar to the sampling mechanism combined with the Laplace mechanism: The latter is the same as the former except that $\bar{\mathbf{I}}_{\varepsilon}$ is replaced by $S(\mathbf{I}, \varepsilon)$. Specifically, instead of discarding users with $\Phi(u) < \varepsilon$, the sampling mechanism keeps them with probability $\min\left\{\frac{e^{\Phi(u)}-1}{e^{\varepsilon}-1}, 1\right\}$. Thus, the (expected) error of the sampling mechanism is

$$\text{Error}^{S}(\mathbf{I}, \varepsilon) = \text{Sum}(\mathbf{I}) - \mathbf{E}[\text{Sum}(S(\mathbf{I}, \varepsilon))] + 1/\varepsilon. \tag{8}$$

Since $\text{Sum}(\bar{\mathbf{I}}_{\varepsilon}) \le \mathbf{E}[\text{Sum}(S(\mathbf{I}, \varepsilon))] \le \text{Sum}(\mathbf{I})$, we see that (8) is better than (6). However, we show that the advantage is no more than a constant factor, when both are using an optimal ε.

Lemma 4.3. *There is a constant c such that for any Φ and any $\mathbf{I}$, $\min_{\varepsilon} \text{Error}(\mathbf{I}, \varepsilon) \le c \cdot \min_{\varepsilon} \text{Error}^{S}(\mathbf{I}, \varepsilon)$.*

Proof. We assume the users with non-zero values are denoted as $u_1, u_2, \ldots, u_n$ such that $\Phi(u_1) \le \ldots \le \Phi(u_n)$, i.e., they are ranked by their privacy level in ascending order. Then we have

$$\min_{\varepsilon} \text{Error}(\mathbf{I}, \varepsilon) = \min_{i \in \{0,1,\ldots,n-1\}} i + \frac{1}{\Phi(u_{i+1})}$$

and

$$\min_{\varepsilon} \text{Error}^{S}(\mathbf{I}, \varepsilon) = O\left(\min_{i \in \{0,1,\ldots,n-1\}} i + \frac{1}{\Phi(u_{i+1})} - \frac{\sum_{t=1}^{i} \Phi(u_t)}{\Phi(u_{i+1})}\right)$$

The latter is because when $\Phi(u), \varepsilon$ is not too large, the sampling probability $\frac{e^{\Phi(u)}-1}{e^{\varepsilon}-1} = O(\frac{\Phi(u)}{\varepsilon})$. Thus our goal is to show

$$\min_{i \in \{0,1,\ldots,n-1\}} i + \frac{1}{\Phi(u_{i+1})} = O\left(\min_{i \in \{0,1,\ldots,n-1\}} i + \frac{1}{\Phi(u_{i+1})} - \frac{\sum_{t=1}^{i} \Phi(u_t)}{\Phi(u_{i+1})}\right)$$

Denote $i^* = \text{argmin}_{i \in \{0,1,\ldots,n-1\}} i + \frac{1}{\Phi(u_{i+1})}$. Let $M = \max\{i^*, \frac{1}{\Phi(u_{i^*+1})}\}$, then $M \le \min_{i \in \{0,1,\ldots,n-1\}} i + \frac{1}{\Phi(u_{i+1})} \le 2M$. Consider $i' = \lfloor \frac{M}{2} \rfloor$.

Since i^* is optimal, the error at i' should be larger, that is

$$i' + \frac{1}{\Phi(u_{i'+1})} \ge M \ge 2i'$$

Which means $\Phi(u_{i'+1}) \le \frac{1}{i'}$. Then for $i \le i'$, we directly have

$$i + \frac{1}{\Phi(u_{i+1})} - \frac{\sum_{t=1}^{i} \Phi(u_t)}{\Phi(u_{i+1})} \ge \frac{1}{\Phi(u_{i+1})} \ge i'$$

And for $i > i'$, we have

$$\begin{aligned} i + \frac{1}{\Phi(u_{i+1})} - \frac{\sum_{t=1}^{i}\Phi(u_t)}{\Phi(u_{i+1})} &= i + \frac{1}{\Phi(u_{i+1})} - \frac{\sum_{t=1}^{i'}\Phi(u_t)}{\Phi(u_{i+1})} - \frac{\sum_{t=i'+1}^{i}\Phi(u_t)}{\Phi(u_{i+1})} \\ &\geq i + \frac{1}{\Phi(u_{i+1})} - \frac{1}{\Phi(u_{i+1})} - \frac{\sum_{t=i'+1}^{i}\Phi(u_t)}{\Phi(u_{i+1})} \\ &\geq i' \end{aligned}$$

Thus we have

$$\begin{aligned} \min_{i\in\{0,1,\ldots,n-1\}} i + \frac{1}{\Phi(u_{i+1})} - \frac{\sum_{t=1}^{i}\Phi(u_t)}{\Phi(u_{i+1})} &\geq i' \\ &\geq \frac{1}{8}\min_{i\in\{0,1,\ldots,n-1\}} i + \frac{1}{\Phi(u_{i+1})} \end{aligned}$$

□

Thus, there is no significant difference between the sampling mechanism and personalized truncation themselves — the key issue in both is how to find a good ε. While our Algorithm 1 finds a provably near-optimal ε for every Φ and $\mathbf{I}$, [22] suggests a simple heuristic of setting it to the average $\Phi(u)$, which can be far from optimal, as illustrated in the following example.

Example 4.4. Consider the same setup as in Example 4.1, where the n users have privacy parameters $\Phi(u_1) = \cdots = \Phi(u_{n/2}) = \frac{1}{\sqrt{n}}$, $\Phi(u_{n/2+1}) = \ldots = \Phi(u_n) = 1$, and all have bit 1. The optimal sampling threshold is also $\varepsilon^* = \frac{1}{\sqrt{n}}$, which has an error of $O(\sqrt{n})$, same as that of personalized truncation. On the other hand, the average $\Phi(u)$ is $\frac{1}{2} + \frac{1}{2\sqrt{n}}$. Setting ε to this value would sample each of the first $n/2$ users with probability $O(1/\sqrt{n})$, leading to a bias of $n/2 - O(\sqrt{n}) = \Omega(n)$. □

In fact, we can also apply Algorithm 1 to find a near-optimal ε for the sampling mechanism as well, by simply replacing $\bar{\mathbf{I}}_{\varepsilon_i}$ with $S(\mathbf{I}, \varepsilon_i)$. Both the privacy proof and utility analysis can be easily adapted to this version. Nevertheless, by Lemma 4.3, this version yields no asymptotic difference. In our experiments, we do see some constant-factor improvements when using this version of the algorithm.

Compare with the exponential mechanism. Jorgensen et al. [22] have also extended the (inverse sensitivity based) exponential mechanism [4, 9, 28] to PDP, namely when the output domain is finite and using the inverse path length as the utility function. Although not analyzed in [22], we show in Appendix B of the full version [33] that the error of the exponential mechanism, when applied to the basic counting problem, is

$$\mathrm{Error}^{\mathrm{exp}}(\mathbf{I}) = \max\{\kappa(\mathbf{I}, 0), \kappa(\mathbf{I}, 1)\}, \tag{9}$$

where

$$\kappa(\mathbf{I}, b) = \max\left\{|\boldsymbol{u}| : \boldsymbol{u} \subseteq \mathcal{U}, \mathbf{I}(u) = b \text{ for all } u \in \boldsymbol{u}, \sum_{u\in\boldsymbol{u}} \Phi(u) \leq \log(|\mathcal{U}|)\right\}.$$

In other words, we sort the users with bit b by $\Phi(u)$ in ascending order. Then $\kappa(\mathbf{I}, b)$ is the largest k such that the sum of $\Phi(u)$ of the first k users with bit b is no more than $\log(|\mathcal{U}|)$.

Example 4.5. Again consider the setup in Example 4.1, where the n users have privacy parameters $\Phi(u_1) = \cdots = \Phi(u_{n/2}) = \frac{1}{\sqrt{n}}$, $\Phi(u_{n/2+1}) = \ldots = \Phi(u_n) = 1$, and all have bit 1. On this instance, $\kappa(\mathbf{I}, 0) = 0$ and $\kappa(\mathbf{I}, 1) = \sqrt{n}\log n$, so $\mathrm{Error}^{\mathrm{exp}}(\mathbf{I}) = O(\sqrt{n}\log n)$. If we change the bits of the first $n/2$ users to 0, then $\kappa(\mathbf{I}, 0) = \sqrt{n}\log n$ and $\kappa(\mathbf{I}, 1) = \log n$, so $\mathrm{Error}^{\mathrm{exp}}(\mathbf{I})$ is still $O(\sqrt{n}\log n)$. On the other

hand, the error of Algorithm 1 on the first instance is $O(\sqrt{n}\log n \log\log n)$ and $O(\log n \log\log n)$ on the second. □

More generally, below we prove the following relationship:

Lemma 4.6. *There is a constant c such that for any Φ and any $\mathbf{I}$, $\min_\varepsilon \mathrm{Error}(\mathbf{I},\varepsilon) \le c \cdot \kappa(\mathbf{I},1)$.*

Proof. We assume the users with non-zero values are denoted as $u_1, u_2, ..., u_n$ such that $\Phi(u_1) \le ... \le \Phi(u_n)$, i.e., they are ranked by their privacy level in ascending order. Let $i^* = \mathrm{argmin}_i\{i + \frac{1}{\Phi(u_{i+1})}\}$, and let $M = \max\{i^*, \frac{1}{\Phi(u_{i^*+1})}\}$. We only need to show $M = O(\kappa(\mathbf{I},1))$.

Consider $i' = \lfloor \frac{M}{2} \rfloor$, such i' is valid since we assume $\varepsilon_{\min} \ge \frac{1}{n}$. We have by definition

$$i' + \frac{1}{\Phi(u_{i'+1})} \ge i^* + \frac{1}{\Phi(u_{i^*+1})} > M \ge 2i'$$

So $i'\Phi(u_{i'}) \le i'\Phi(u_{i'+1}) < 1$, which means $i' \le \kappa(\mathbf{I},1)$ and this further implies $M = O(\kappa(\mathbf{I},1))$. □

We thus conclude that Algorithm 1 is never worse than the exponential mechanism by more than an $O(\log \frac{\varepsilon_{\max}}{\varepsilon_{\min}} \log\log \frac{\varepsilon_{\max}}{\varepsilon_{\min}})$ factor, and can be much better on some instances, especially when there are many 0 bits owned by conservative users, as illustrated in Example 4.5.

5 SUM ESTIMATION

For the basic counting problem, we used a simple truncation strategy: For some ε, all users with $\Phi(u) \le \varepsilon$ are discarded, and an ε-DP mechanism is applied on the other users. However, as the following example illustrates, this strategy is sub-optimal for more general sum estimation problem with a large B:

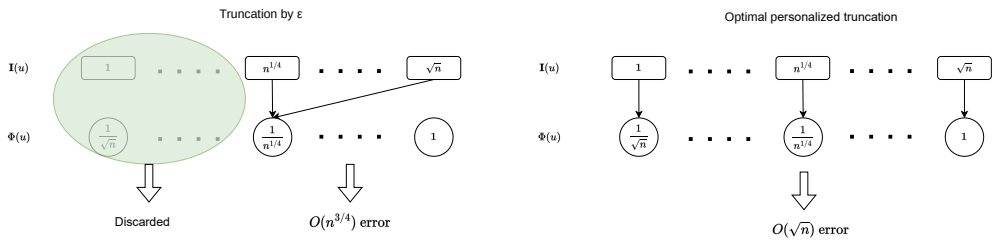

Fig. 1. Visualization of Example 5.1.

Example 5.1. Suppose $\mathcal{U} = \{u_1, \ldots, u_n\}$, and their privacy parameters are $\Phi(u_i) = \sqrt{i/n}$ for $i = 1, \ldots, n$. Consider the instance $\mathbf{I}$ in which $\mathbf{I}(u_i) = \sqrt{i}$ for $i = 1, \ldots, n$. Suppose we apply the truncation mechanism above with $\varepsilon = \sqrt{i/n}$ for some i. Then the error is (ignoring constant factors) $i^{1.5} + \sqrt{n}/\varepsilon = i^{1.5} + n/\sqrt{i}$. Thus, even using the optimal $i = \sqrt{n}$, the error is $\Omega(n^{3/4})$. On the other hand, the optimal personalized truncation would use the truncation function $\boldsymbol{\tau}(u_i) = \sqrt{i/n}$, which results in 0 bias and $\max_i \frac{\boldsymbol{\tau}(u_i)}{\Phi(u_i)} = \sqrt{n}$ noise. Please see Figure 1 for a graphical illustration. □

This example implies that, for the sum estimation problem with $B > 1$, we must exploit the full power of personalized truncation and we should try to find the optimal truncation function in the entire space of $[B]^{\mathcal{U}}$.

To reduce this exponentially large search space, our idea is to partition the space $[B]^{\mathcal{U}}$ into groups based on the value of $\max_u \frac{\boldsymbol{\tau}(u)}{\Phi(u)}$. Note that the smallest nonzero value of $\max_u \frac{\boldsymbol{\tau}(u)}{\Phi(u)}$ is $\frac{1}{\varepsilon_{\max}}$

while the largest is $\frac{B}{\varepsilon_{\min}}$. We hence divide $[B]^{\mathcal{U}}$ into $t = \lceil \log \frac{\varepsilon_{\max} B}{\varepsilon_{\min}} \rceil$ disjoint g roups $\mathbf{G}_1, \ldots, \mathbf{G}_t$, such that $\mathbf{G}_i = \{\boldsymbol{\tau} : \frac{2^{i-1}}{\varepsilon_{\max}} < \max_u \frac{\boldsymbol{\tau}(u)}{\Phi(u)} \le \frac{2^i}{\varepsilon_{\max}}\}$ for $i = 2, \ldots, t$. Specially, define $\mathbf{G}_1 = \{\boldsymbol{\tau} : \frac{1}{\varepsilon_{\max}} \le \max_u \frac{\boldsymbol{\tau}(u)}{\Phi(u)} \le \frac{2}{\varepsilon_{\max}}\}$ and $\mathbf{G}_0 = \{\mathbf{0}\}$, so that all possible values of $\max_u \frac{\boldsymbol{\tau}(u)}{\Phi(u)}$ are covered. Note that the values of $\max_u \frac{\boldsymbol{\tau}(u)}{\Phi(u)}$ in each group differ by no more than a factor of 2.

Algorithm 2: PDP Sum

Input: $\mathbf{I}$, Φ, B, β

1 $t = \lceil \log \frac{\varepsilon_{\max} B}{\varepsilon_{\min}} \rceil$;
2 **for** $i \leftarrow 1, 2, \ldots, t$ **do**
3 Define $\boldsymbol{\tau}^i$ such that $\boldsymbol{\tau}^i(u) = \lfloor \frac{2^i \Phi(u)}{\varepsilon_{\max}} \rfloor$;
4 Define $\mathbf{I}_{\boldsymbol{\tau}^i}$ such that $\mathbf{I}_{\boldsymbol{\tau}^i}(u) = \min(\mathbf{I}(u), \boldsymbol{\tau}^i(u))$;
5 $\widetilde{\mathrm{Sum}_i}(\mathbf{I}) = \mathrm{Sum}(\mathbf{I}_{\boldsymbol{\tau}^i}) + \mathrm{Lap}(\frac{2^i t}{\varepsilon_{\max}}) - \frac{2^i t}{\varepsilon_{\max}} \ln \frac{t}{\beta}$;
6 **end**
7 **return** $\widetilde{\mathrm{Sum}}(\mathbf{I}) = \max\{\max_i \widetilde{\mathrm{Sum}_i}(\mathbf{I}), 0\}$;

Our PDP sum estimation mechanism is given in Algorithm 2. It follows the same framework as in Algorithm 1, where we return the maximum privatized estimate from t candidates, each of which is an underestimate of the true sum. The crucial observation that makes Algorithm 2 work is that each group $\mathbf{G}_i$ only needs to elect one best candidate for this competition. More precisely, observe that $\mathrm{Sum}(\mathbf{I}_{\boldsymbol{\tau}})$ is non-decreasing in the coordinates of $\boldsymbol{\tau}$. Thus, in group $\mathbf{G}_i$, the $\boldsymbol{\tau}^i$ with $\boldsymbol{\tau}^i(u) = \lfloor \frac{2^i \Phi(u)}{\varepsilon_{\max}} \rfloor$ yields the smallest bias. Meanwhile, all the $\boldsymbol{\tau} \in \mathbf{G}_i$ have the same noise term up to a factor of 2. Thus, for each $\mathbf{G}_i$, it suffices to only consider this particular $\boldsymbol{\tau}$. Finally, for $\mathbf{G}_0$, its only member $\boldsymbol{\tau} = \mathbf{0}$ always returns 0, and this is taken into consideration in line 7 of Algorithm 2.

Below we prove its privacy and show that it achieves the optimal personalized truncation within an $O(\log \frac{\varepsilon_{\max} B}{\varepsilon_{\min}} \log\log \frac{\varepsilon_{\max} B}{\varepsilon_{\min}})$ factor.

Theorem 5.2. *For any Φ, β, Algorithm 2 satisfies Φ-PDP. Additionally, for any $\mathbf{I}$, its error is at most*

$$\min_{\boldsymbol{\tau}} \left(\mathrm{Sum}(\mathbf{I}) - \mathrm{Sum}(\mathbf{I}_{\boldsymbol{\tau}}) + 4t \ln \frac{t}{\beta} \max_u \frac{\boldsymbol{\tau}(u)}{\Phi(u)} \right)$$

with probability at least $1 - \beta$, where $t = \lceil \log \frac{\varepsilon_{\max} B}{\varepsilon_{\min}} \rceil$.

Proof. We start with the privacy part.

Let $t = \lceil \log \frac{\varepsilon_{\max}}{\varepsilon_{\min}} \rceil$. We will show each iteration of the for-loop is $\frac{\Phi(\cdot)}{t}$-PDP. Consider the ith iteration and $\mathbf{I} \overset{u}{\sim} \mathbf{I}'$ for arbitrary u. We first show $|\mathrm{Sum}(\mathbf{I}_{\boldsymbol{\tau}^i}) - \mathrm{Sum}(\mathbf{I}'_{\boldsymbol{\tau}^i})| \le \frac{2^i \Phi(u)}{\varepsilon_{\max}}$. Note the value of $\boldsymbol{\tau}^i$ is independent of $\mathbf{I}, \mathbf{I}'$. Without loss of generality, assume $\mathbf{I} \subseteq \mathbf{I}'$, then we have:

$$\begin{aligned}
\mathrm{Sum}(\mathbf{I}'_{\boldsymbol{\tau}^i}) - \mathrm{Sum}(\mathbf{I}_{\boldsymbol{\tau}^i}) &= \sum_{k \in \mathcal{U}} \min(\mathbf{I}(k), \tau^i(k)) - \sum_{k \in \mathcal{U}} \min(\mathbf{I}'(k), \tau^i(k)) \\
&= \min(\mathbf{I}'(u), \lfloor \frac{2^i \Phi(u)}{\varepsilon_{\max}} \rfloor) \\
&\le \frac{2^i \Phi(u)}{\varepsilon_{\max}}
\end{aligned}$$

Here the second line is because $\mathbf{I}, \mathbf{I}'$ only differ on user u. On the other hand, obviously $\mathrm{Sum}(\mathbf{I}_{\boldsymbol{\tau}^i}) \le \mathrm{Sum}(\mathbf{I}'_{\boldsymbol{\tau}^i})$. So $|\mathrm{Sum}(\mathbf{I}_{\boldsymbol{\tau}^i}) - \mathrm{Sum}(\mathbf{I}'_{\boldsymbol{\tau}^i})| \le \frac{2^i \Phi(u)}{\varepsilon_{\max}}$ holds.

Then according to lemma 3.1, adding a Laplace noise with scale $\frac{2^i t}{\varepsilon_{\max}}$ preserves $\frac{\Phi(u)}{t}$-DP, namely for any y we have:

$$\Pr\left(\widetilde{\mathrm{Sum}_i}(\mathbf{I}) = y\right) \le e^{\frac{\Phi(u)}{t}} \Pr\left(\widetilde{\mathrm{Sum}_i}(\mathbf{I}') = y\right)$$

Since this holds for any u, according to the definition of PDP, this iteration will be $\frac{\Phi(\cdot)}{t}$-PDP. Then the whole process will be Φ-PDP according to basic composition.

Then we prove its utility. As shown in the proof of Theorem 4.2, with probability at least $1-\beta$, for all i we have:

$$\widetilde{\mathrm{Sum}_i}(\mathbf{I}) \le \mathrm{Sum}(\mathbf{I}_{\boldsymbol{\tau}^i}) \le \mathrm{Sum}(\mathbf{I}).$$

For $\boldsymbol{\tau} = \mathbf{0}$, obviously we have

$$\begin{aligned}
&|\widetilde{\mathrm{Sum}}(\mathbf{I}) - \mathrm{Sum}(\mathbf{I})| \\
\le &|\mathrm{Sum}(\mathbf{I}_{\boldsymbol{\tau}}) - \mathrm{Sum}(\mathbf{I})| \\
\le &|\mathrm{Sum}(\mathbf{I}_{\boldsymbol{\tau}}) - \mathrm{Sum}(\mathbf{I})| + 4\lceil \log \frac{\varepsilon_{\max} B}{\varepsilon_{\min}} \rceil \ln \frac{\lceil \log \frac{\varepsilon_{\max} B}{\varepsilon_{\min}} \rceil}{\beta} \max_u \frac{\boldsymbol{\tau}(u)}{\Phi(u)}
\end{aligned}$$

For any given $\boldsymbol{\tau} \in \{0, 1, ..., B\}^{|\mathcal{U}|}$ and $\boldsymbol{\tau} \neq \mathbf{0}$, we can always find an i such that $\boldsymbol{\tau} \in \mathbf{G}_i$. Then we have:

$$|\mathrm{Sum}(\mathbf{I}_{\boldsymbol{\tau}^i}) - \mathrm{Sum}(\mathbf{I})| \le |\mathrm{Sum}(\mathbf{I}_{\boldsymbol{\tau}}) - \mathrm{Sum}(\mathbf{I})|.$$

Because the bias at the group containing $\boldsymbol{\tau}$ will be no greater than the bias at $\boldsymbol{\tau}$. Thus with probability at least $1-\beta$ we have:

$$\begin{aligned}
|\widetilde{\mathrm{Sum}}(\mathbf{I}) - \mathrm{Sum}(\mathbf{I})| &\le |\widetilde{\mathrm{Sum}_i}(\mathbf{I}) - \mathrm{Sum}(\mathbf{I})| \\
&= |\mathrm{Sum}(\mathbf{I}_{\boldsymbol{\tau}^i}) + \mathrm{Lap}(\frac{2^i t}{\varepsilon_{\max}}) - \frac{2^i t}{\varepsilon_{\max}} \ln \frac{t}{\beta} - \mathrm{Sum}(\mathbf{I})| \\
&\le |\mathrm{Sum}(\mathbf{I}_{\boldsymbol{\tau}^i}) - \mathrm{Sum}(\mathbf{I})| + \frac{2^{i+1} t}{\varepsilon_{\max}} \ln \frac{t}{\beta} \\
&\le |\mathrm{Sum}(\mathbf{I}_{\boldsymbol{\tau}}) - \mathrm{Sum}(\mathbf{I})| + 4t \ln \frac{t}{\beta} \max_u \frac{\boldsymbol{\tau}(u)}{\Phi(u)}
\end{aligned}$$

Here the last line is because $\boldsymbol{\tau}$ is contained in the ith group, then

$$\frac{2^{i-1}}{\varepsilon_{\max}} \le \max_u \frac{\boldsymbol{\tau}(u)}{\Phi(u)}$$

Since the above inequality holds for any $\boldsymbol{\tau} \in \{0, 1, ..., B\}^{|\mathcal{U}|}$, the actual error of Algorithm 2 should be no greater than the minimum of them. □

Algorithm 2 can also be implemented efficiently, even over an infinite user universe. This is because to compute $\mathrm{Sum}(\mathbf{I}_{\boldsymbol{\tau}^i})$, we just need to consider users that contribute nonzero values, which can be easily done in $O(n)$ time. Thus, the total running time is $O(n \log \frac{\varepsilon_{\max} B}{\varepsilon_{\min}})$. Additionally, when $B = 1$, Algorithm 2 reproduces the result of our basic counting algorithm, but with a slightly higher running time.

6 SUM ESTIMATION UNDER USER-LEVEL PDP

Finally, we consider the most general setting, where each user u may own (solely or jointly with other users) multiple values, and has a personalized privacy parameter $\Phi(u)$. As mentioned in Section 3.1, this captures select-join-aggregate (SJA) queries with numerical aggregations such as sum or count in relational database. Here we assume that the sum of all values owned by any user

is bounded by a parameter B. We still aim to achieve the error guarantee of (5). This allows us to set a large B, which will only affect the error guarantee logarithmically.

In this general model, the first issue we need to clarify is how the personalized truncation mechanism works under a given truncation function $\boldsymbol{\tau}$. For tuple-level PDP, we simply truncate $\mathbf{I}$ into $\mathbf{I}_{\boldsymbol{\tau}}$ such that $\mathbf{I}_{\boldsymbol{\tau}}(u) = \min(\mathbf{I}(u), \boldsymbol{\tau}(u))$, and then add a Laplace noise of scale $\max_u \frac{\boldsymbol{\tau}(u)}{\Phi(u)}$ to $\mathrm{Sum}(\mathbf{I}_{\boldsymbol{\tau}})$. However, under user-level PDP, each value $\mathbf{I}(\boldsymbol{u})$ is jointly owned by multiple users in $\boldsymbol{u}$, and it is not clear whose truncation threshold to use. One natural attempt is to use the smallest threshold, i.e., set $\mathbf{I}_{\boldsymbol{\tau}}(\boldsymbol{u}) = \min(\mathbf{I}(\boldsymbol{u}), \min_{u\in\boldsymbol{u}} \boldsymbol{\tau}(u))$. However, as the following example shows, this method does not satisfy PDP.

Example 6.1. Suppose $\mathcal{U} = \{u_1, \ldots, u_n\}$, and their privacy parameters are $\Phi(u_i) = 1$ for all i. Consider $\mathbf{I}$ such that $\mathbf{I}(u_1, u_i) = 1$ for $i = 2, \ldots, n$, and $\mathbf{I}(\boldsymbol{u}) = 0$ for all other $\boldsymbol{u}$. Let $\mathbf{I}'$ be an empty instance. It is easy to see that $\mathbf{I} \overset{u_1}{\sim} \mathbf{I}'$. Suppose we apply the truncation function $\boldsymbol{\tau}(u) = 1$ for all u. Then the simple truncation method above will not truncate any data for either $\mathbf{I}$ or $\mathbf{I}'$. However, $\mathrm{Sum}(\mathbf{I}_{\boldsymbol{\tau}}) - \mathrm{Sum}(\mathbf{I}'_{\boldsymbol{\tau}}) = n - 1$, so adding a noise of scale $\max_u \frac{\boldsymbol{\tau}(u)}{\Phi(u)} = 1$ is not enough to make the output distributions on these two instances $\Phi(u_1)$-indistinguishable. □

The issue exemplified by the example above is that we need to ensure that, in the truncated instance $\mathbf{I}_{\boldsymbol{\tau}}$, the *total* contributions from each user u is bounded by $\boldsymbol{\tau}(u)$, which can be formalized as

$$\sum_{\boldsymbol{u}\ni u} \mathbf{I}_{\boldsymbol{\tau}}(\boldsymbol{u}) \le \boldsymbol{\tau}(u)$$

for all $u \in \mathcal{U}$. In addition, we need to make sure that $\mathbf{I}_{\boldsymbol{\tau}}$ satisfy the following properties:

- $0 \le \mathbf{I}_{\tau}(\boldsymbol{u}) \le \mathbf{I}(\boldsymbol{u})$ for all $\boldsymbol{u}$, which means it is a valid truncation from the original dataset
- The truncated sum $\sum_{\boldsymbol{u}} \mathbf{I}_{\boldsymbol{\tau}}(\boldsymbol{u})$ should be maximized so that the truncation bias is minimized

Obtaining an $\mathbf{I}_{\boldsymbol{\tau}}$ satisfying all the above conditions is non-trivial. But if we treat each $\mathbf{I}_{\boldsymbol{\tau}}(\boldsymbol{u})$ as a variable and the conditions as constraints, the above can be formulated as the following linear program (LP):

$$\begin{aligned} \max \quad & \sum_{\boldsymbol{u}} \mathbf{I}_{\boldsymbol{\tau}}(\boldsymbol{u}) \\ \text{s.t.} \quad & \sum_{\boldsymbol{u}\ni u} \mathbf{I}_{\boldsymbol{\tau}}(\boldsymbol{u}) \le \boldsymbol{\tau}(u), \quad u \in \mathcal{U}, \\ & 0 \le \mathbf{I}_{\boldsymbol{\tau}}(\boldsymbol{u}) \le \mathbf{I}(\boldsymbol{u}), \quad \boldsymbol{u} \subseteq 2^{\mathcal{U}}. \end{aligned} \tag{10}$$

Although each $\mathbf{I}_{\boldsymbol{\tau}}(\boldsymbol{u})$ is a variable in the LP, but since $\mathbf{I}_{\boldsymbol{\tau}}(\boldsymbol{u})$ must be 0 if $\mathbf{I}(\boldsymbol{u}) = 0$, we actually only need to include all the $\mathbf{I}_{\boldsymbol{\tau}}(\boldsymbol{u})$ for which $\mathbf{I}(\boldsymbol{u}) > 0$. Thus, the size of the LP is linear in the input size of $\mathbf{I}$, so it can be solved in polynomial time.

We use the optimal solution of this LP to define the truncated instance $\mathbf{I}_{\boldsymbol{\tau}}$. We will show later that the global sensitivity of $\mathrm{Sum}(\mathbf{I}_{\boldsymbol{\tau}})$ is bounded so that the personalized truncation mechanism satisfies PDP. The remaining issue of how to find a near-optimal $\boldsymbol{\tau}$ can be solved using a similar idea from Algorithm 2: We divide all possible $\boldsymbol{\tau}$ into $t = \lceil \log \frac{\varepsilon_{\max} B}{\varepsilon_{\min}} \rceil$ disjoint groups $\mathbf{G}_1, \ldots, \mathbf{G}_t$, such that $\mathbf{G}_i = \{\boldsymbol{\tau} \in \{0, 1, ..., B\}^{|\mathcal{U}|} \mid \frac{2^{i-1}}{\varepsilon_{\max}} < \max_u \frac{\boldsymbol{\tau}(u)}{\Phi(u)} \le \frac{2^i}{\varepsilon_{\max}}\}$ and find the best estimation among all groups. We take the maximum between the best estimation and 0 to take $\mathbf{G}_0 = \{\mathbf{0}\}$ into consideration. Details are given in Algorithm 3.

It is worth noting that, in the special case where all $|\boldsymbol{u}| = 1$ for all $\boldsymbol{u}$, the optimal solution of the LP (10) exactly yields the simple truncation method $\mathbf{I}_{\boldsymbol{\tau}}(u) = \min(\mathbf{I}(u), \boldsymbol{\tau}(u))$. Namely, Algorithm 3 degenerates into Algorithm 2 under tuple-level PDP.

Algorithm 3: Sum estimation under User-level PDP

Input: $\mathbf{I}$, Φ, B, β

1 $t = \lceil \log \frac{\varepsilon_{\max} B}{\varepsilon_{\min}} \rceil$;

2 **for** $i \leftarrow 1, 2, \ldots, t$ **do**

3 Define $\boldsymbol{\tau}^i$ such that $\boldsymbol{\tau}^i(u) = \lfloor \frac{2^i \Phi(u)}{\varepsilon_{\max}} \rfloor$;

4 Let $\mathbf{I}_{\boldsymbol{\tau}^i}$ be the optimal solution of (10);

5 $\widetilde{\mathrm{Sum}_i}(\mathbf{I}) = \mathrm{Sum}(\mathbf{I}_{\boldsymbol{\tau}^i}) + \mathrm{Lap}(\frac{2^i t}{\varepsilon_{\max}}) - \frac{2^i t}{\varepsilon_{\max}} \ln \frac{t}{\beta}$;

6 **end**

7 **return** $\widetilde{\mathrm{Sum}}(\mathbf{I}) = \max\{\max_i \widetilde{\mathrm{Sum}_i}(\mathbf{I}), 0\}$;

Theorem 6.2. *Given any Φ, β, Algorithm 3 satisfies Φ-PDP. Additionally, its error is at most*

$$\min_{\boldsymbol{\tau}} \left(\mathrm{Sum}(\mathbf{I}) - \mathrm{Sum}(\mathbf{I}_{\boldsymbol{\tau}}) + 4t \ln \frac{t}{\beta} \max_u \frac{\boldsymbol{\tau}(u)}{\Phi(u)} \right)$$

with probability at least $1 - \beta$, where $t = \lceil \log \frac{\varepsilon_{\max} B}{\varepsilon_{\min}} \rceil$.

Proof. We only need to show its privacy, namely each iteration is $\frac{\Phi(\cdot)}{t}$-PDP. The proof of utility follows from Theorem 5.2.

Given any i and for any pair of neighbors $\mathbf{I} \overset{u}{\sim} \mathbf{I}'$ that differ by arbitrary user u, we show the sensitivity of $\mathrm{Sum}(\mathbf{I}_{\boldsymbol{\tau}^i})$ is bounded by $\frac{2^i \Phi(u)}{\varepsilon_{\max}}$, that is, $|\mathrm{Sum}(\mathbf{I}_{\boldsymbol{\tau}^i}) - \mathrm{Sum}(\mathbf{I}'_{\boldsymbol{\tau}^i})| \le \frac{2^i \Phi(u)}{\varepsilon_{\max}}$. Then according to a similar reasoning as in the proof of Theorem 5.2, adding $\mathrm{Lap}(\frac{2^i t}{\varepsilon_{\max}})$ in line 4 is enough to preserve $\frac{\Phi(\cdot)}{t}$-PDP.

Without loss of generality, assume $\mathbf{I} \subseteq \mathbf{I}'$. Let P denote the optimization problem described in Equation (10), when instantiated with constraints $\mathbf{I}$ and $\boldsymbol{\tau}^i$. And let P' be the same problem instantiated with constraints $\mathbf{I}'$ and $\boldsymbol{\tau}^i$. First of all, we should note any solution to P will also be a solution to P', since all constraints that appear in P are also satisfied in P'. Thus $\mathbf{I}_{\boldsymbol{\tau}^i}$ is also a solution of P', although it may not be optimal. As a result, we have

$$\mathrm{Sum}(\mathbf{I}_{\boldsymbol{\tau}^i}) \le \mathrm{Sum}(\mathbf{I}'_{\boldsymbol{\tau}^i})$$

On the other hand, if we define $\mathbf{I}''_{\boldsymbol{\tau}^i}$ such that

$$\mathbf{I}''_{\boldsymbol{\tau}^i}(\boldsymbol{u}) = \begin{cases} \mathbf{I}'_{\boldsymbol{\tau}^i}(\boldsymbol{u}), & \text{if } u \notin \boldsymbol{u}; \\ 0, & \text{if otherwise.} \end{cases}$$

Then $\mathbf{I}''_{\boldsymbol{\tau}^i}$ is a solution to P, which implies

$$\mathrm{Sum}(\mathbf{I}_{\boldsymbol{\tau}^i}) \ge \mathrm{Sum}(\mathbf{I}''_{\boldsymbol{\tau}^i})$$

In the meanwhile, from the definition of P', we should also have

$$\mathrm{Sum}(\mathbf{I}''_{\boldsymbol{\tau}^i}) = \sum_{\boldsymbol{u} \in \binom{\mathcal{U}}{\le l}} \mathbf{I}''_{\boldsymbol{\tau}^i}(\boldsymbol{u}) = \sum_{\boldsymbol{u} \in \binom{\mathcal{U}}{\le l}, u \notin \boldsymbol{u}} \mathbf{I}'_{\boldsymbol{\tau}^i}(\boldsymbol{u}) \ge \sum_{\boldsymbol{u} \in \binom{\mathcal{U}}{\le l}} \mathbf{I}'_{\boldsymbol{\tau}^i}(\boldsymbol{u}) - \boldsymbol{\tau}^i(u)$$

Combining the above arguments, we have

$$|\mathrm{Sum}(\mathbf{I}_{\boldsymbol{\tau}^i}) - \mathrm{Sum}(\mathbf{I}'_{\boldsymbol{\tau}^i})| \le \frac{2^i \Phi(u)}{\varepsilon_{\max}}.$$

□

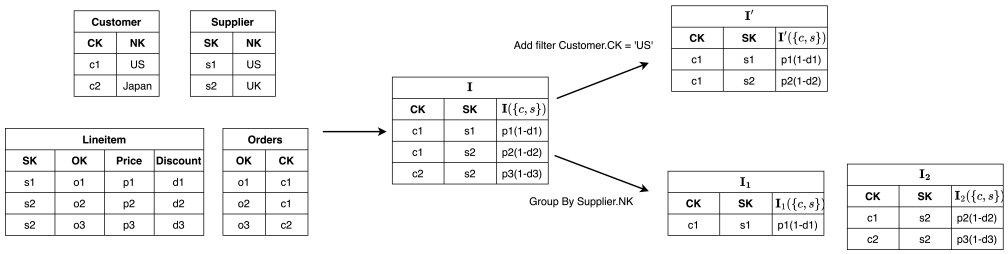

Fig. 2. An example showing the workflow from a query to $\mathbf{I}$. On the left is the TPC-H data, in the middle is the $\mathbf{I}$ obtained from executing the original query in Section 6.1. On the right are the instances for queries with filter predicate ($\mathbf{I}'$) and group-by clause ($\mathbf{I}_1, \mathbf{I}_2$); empty instances are omitted.

To summarize, the core intuition of our approach is to truncate users with high sensitivity/strong privacy requirements. This approach presents two primary challenges: (1) computing the truncated sum, and (2) determining the optimal truncation vector $\boldsymbol{\tau}$. To compute the truncated sum, we develop the linear program (10) such that its result has low sensitivity. Then we design Algorithm 3 to find a near-optimal $\boldsymbol{\tau}$, ensuring that the resulting error remains within a log factor of the optimal error.

6.1 Handling SQL Queries

Up to now, we have assumed that the input is given as a mapping $\mathbf{I} : 2^{\mathcal{U}} \to [B]$. For a SQL query consisting of joins, selections, and (group-by) aggregations, one should perform some preprocessing to convert the data into such a mapping, as described in [18]. We briefly review the process using the following query as an example, which finds the total discounted price in the TPC-H dataset:

```
SELECT SUM(L_price * (1 - L_discount))
FROM Customer, Supplier, Lineitem, Orders
WHERE Customer.CK = Order.CK
  AND Orders.OK = Lineitem.OK
  AND Lineitem.SK = Supplier.SK;
```

Suppose we want to protect the privacy of every customer and supplier, so the user universe $\mathcal{U}$ consists of all suppliers and customers. We first compute the join results without the `SUM` aggregation. This yields the mapping $\mathbf{I}$ as shown in the middle table of Figure 2, as required by our algorithms.

Selection filters can also be handled easily. Consider the query above with a filter `Customer.NK='US'`. We simply set $\mathbf{I}(\{c, s\}) = 0$ for all $\{c, s\}$ pairs where the nation of c is not US. Effectively, this removes all such pairs from $\mathbf{I}$, since all 0 mappings need not be represented explicitly. The resulting mapping is shown on the top-right corner of Figure 2.

A group-by query can be decomposed into multiple sub-queries, each constrained to one of the groups using a filter. For instance, consider the previous query with a `GROUP BY Supplier.NK` clause. We divide the query into 2 subqueries, with filters `Supplier.NK='US'` and `Supplier.NK='UK'`, respectively. This results in mapping $\mathbf{I}_1, \mathbf{I}_2$ shown on the bottom-right corner of Figure 2. Then we can apply Algorithm 3 on each sub-query separately, with the privacy budget divided according to the composition rule in Lemma 3.4.

7 EXTENSIONS

7.1 Dealing with the Real Domain

Actually, Algorithm 2 and 3 can handle the real domain (where $\mathbf{I}(\boldsymbol{u})$ is allowed to take any real value in $[0, B]$) naively by still searching for a best $\boldsymbol{\tau}$ in the integer domain. But in that case the error guarantees in Theorem 5.2 and 6.2 no longer hold, as illustrated in the following example.

Example 7.1. Suppose $\mathcal{U} = \{u_1, \ldots, u_n\}$. Their privacy parameters are $\Phi(u_i) = 1/\sqrt{n}$ for all i, and each owns a value $\mathbf{I}(u_i) = 1/\sqrt{n}$. The error of Algorithm 2 or 3 on this instance is $\Omega(\sqrt{n})$. On the other hand, using the truncation function $\boldsymbol{\tau}(u) = \frac{1}{\sqrt{n}}$ for all u would lead to an error of $O(1)$. □

The reason why the error guarantees in Theorem 5.2 and 6.2 do not hold in the real domain is that they benchmark themselves against the optimal integer-valued $\boldsymbol{\tau}$. However, as seen in the example above, the optimal $\boldsymbol{\tau}$ might take real values when $\mathbf{I}(\boldsymbol{\tau})$ are real numbers.

On the other hand, Algorithm 2, 3 cannot search for $\boldsymbol{\tau}$ on the real domain directly since the real domain has infinite cardinality, whereas our algorithms require a finite value to divide the privacy budget and also to provide the error bound.

To fix this issue, we discretize the real domain $[0, B]$ into buckets of size δ. Then we run Algorithm 5.2 or 6.2 over the discretized domain $\{0, \delta, 2\delta, ..., \delta\lfloor \frac{B}{\delta} \rfloor\}$. This allows us to reach the optimal truncation error within logarithmic factor where the optimal truncation function takes values in this discretized domain. More precisely, the error will be

$$\min_{\boldsymbol{\tau} \in \{0,\delta,\ldots,\delta\lfloor \frac{B}{\delta} \rfloor\}^{|\mathcal{U}|}} \left(\mathrm{Sum}(\mathbf{I}) - \mathrm{Sum}(\mathbf{I}_{\boldsymbol{\tau}}) + 4t_\delta \ln \frac{t_\delta}{\beta} \max_u \frac{\boldsymbol{\tau}(u)}{\Phi(u)} \right), \tag{11}$$

where $t_\delta = \lceil \log \frac{\varepsilon_{\max} B}{\varepsilon_{\min} \delta} \rceil$. One can easily show that (11) is very close to the optimal error when no restriction is imposed on $\boldsymbol{\tau}$: Suppose $\boldsymbol{\tau}$ is the optimal real-valued truncation function. Then we can round it down to a discretized $\boldsymbol{\tau}$. This rounding down incurs at most $n\delta$ additional bias and no increase in the noise term. Thus, we arrive at the following conclusion:

Corollary 7.2. *Over the real domain, for any $\delta > 0$, Algorithm 5.2 and 6.2 incur an error at most*

$$\min_{\boldsymbol{\tau}} \left(\mathrm{Sum}(\mathbf{I}) - \mathrm{Sum}(\mathbf{I}_{\boldsymbol{\tau}}) + 4t_\delta \ln \frac{t_\delta}{\beta} \max_u \frac{\boldsymbol{\tau}(u)}{\Phi(u)} \right) + n\delta$$

with probability at least $1 - \beta$, where $t_\delta = \lceil \log \frac{\varepsilon_{\max} B}{\varepsilon_{\min} \delta} \rceil$.

The value of δ should be given as an input parameter depending on pre-knowledge of the data scale and also the desired accuracy. Due to the logarithmic dependency on $1/\delta$, one can set δ small so that the additive error term is insignificant. For instance, setting $\delta = 1/n$ reduces the additive error to 1, while the logarithmic factor is just $\log \frac{\varepsilon_{\max} Bn}{\varepsilon_{\min}}$. Such an additive error is insignificant in most real applications, whereas the DP error is usually the dominant term.

7.2 Dealing with the Negative Domain

So far we have assumed that all the values in $\mathbf{I}$ are non-negative, which is an essential assumption to find the best sum estimation by taking the maximum among a series of possible values. When the domain of $\mathbf{I}(\boldsymbol{u})$ contains negative values, we can split the problem into two parts.

Now assume $\mathbf{I} : 2^{\mathcal{U}} \to \{-B, ..., B\}$, namely the domain contains all integers whose absolute value is bounded by B, we construct the following two mappings $\mathbf{I}^+$ and $\mathbf{I}^-$, both from $2^{\mathcal{U}}$ to $[B]$:

$$\mathbf{I}^+(\boldsymbol{u}) = \begin{cases} \mathbf{I}(\boldsymbol{u}), & \text{if } \mathbf{I}(\boldsymbol{u}) \geq 0; \\ 0, & \text{if otherwise.} \end{cases}$$

and

$$\mathbf{I}^-(\boldsymbol{u}) = \begin{cases} 0, & \text{if } \mathbf{I}(\boldsymbol{u}) > 0; \\ -\mathbf{I}(\boldsymbol{u}), & \text{if otherwise.} \end{cases}$$

$\mathbf{I}^+$ and $-\mathbf{I}^-$ can be considered as the positive (negative) part of the original $\mathbf{I}$. We can see both $\mathbf{I}^+$ and $\mathbf{I}^-$ only contains non-negative values, while $\mathrm{Sum}(\mathbf{I}) = \mathrm{Sum}(\mathbf{I}^+) - \mathrm{Sum}(\mathbf{I}^-)$. Additionally, the neighboring relationship is preserved in the sense that $\mathbf{I} \overset{u}{\sim} \mathbf{I}'$ implies either $\mathbf{I}^+ \overset{u}{\sim} \mathbf{I}'^+, \mathbf{I}^- = \mathbf{I}'^-$ or $\mathbf{I}^- \overset{u}{\sim} \mathbf{I}'^-, \mathbf{I}^+ = \mathbf{I}'^+$, but not both. Thus we can run our algorithms on both $\mathbf{I}^+$ and $\mathbf{I}^-$ without splitting the privacy budget, and return $\widetilde{\mathrm{Sum}}(\mathbf{I}^+) - \widetilde{\mathrm{Sum}}(\mathbf{I}^-)$. The error will be at most 2 times larger.

One should note the split of $\mathbf{I}$ into $\mathbf{I}^+$ and $\mathbf{I}^-$ is done as long as the domain of $\mathbf{I}$ contains negative numbers, and does not depend on the actual values in $\mathbf{I}$. Thus the process is data-independent and does not violate DP.

8 EXPERIMENTS

We conduct experiments on each algorithm proposed in this paper. For the counting problem, we compare our algorithm with the sampling mechanism and PDP exponential mechanism (PDP EM) in [22], together with the naive algorithm which adds a Laplace noise with scale $1/\varepsilon_{\min}$. As mentioned in Section 4, our PDP count algorithm can be combined with sampling, so we also add it into comparison. For the sum and query problem, we compare our algorithms with the sampling mechanism as well as the naive algorithm by instantiating the best standard DP mechanism with privacy level $\varepsilon_{\min}$. Note that except for our PDP Sum/Query mechanism, all other mechanisms work by transforming a black box standard DP algorithm into a PDP version. The black box mechanism we used for sum is the universal private estimator [12] and for query processing we use R2T [10]. The threshold ε we use for the sampling mechanism is set to the average of all users' privacy levels as advised in [22].

We summarize the results under the default setting in Table 1, 2. Additionally, we also try different dataset/user universe sizes, different global sensitivity bounds B as well as different privacy specifications to demonstrate the performance of our mechanisms. These results are shown in Figure 3, 4, 5, 6, 7.

Problem Type	Data	Query Result	Technique	Relative Error(%)	Time(s)
Count	Synthetic	100,000	Naive	12.0	0.000002
			Sampling	16.0	0.03
			PDP EM	0.4	0.5
			PDP Count	1.2	0.04
			PDP Count (with sampling)	**0.5**	**0.2**
Sum	Synthetic	99,933,209	Naive	100	0.08
			Sampling	16.0	1.0
			PDP Sum	**1.1**	**2.0**
	Bank	61,589,682	Naive	100	0.08
			Sampling	19.5	0.19
			PDP Sum	**8.6**	**0.96**

Table 1. Summary of results for count and sum under default setting where $|\mathcal{U}| = 10^6$, the performance of our best PDP mechanism is boldfaced.

8.1 Setup

Count and Sum. We evaluate mechanisms for count and sum on synthetic data. We try different user universe size $|\mathcal{U}| = 10^4, 10^5, 10^6, 10^7, 10^8$ and the default value is 10^6. We always keep the ratio of users with non-zero contribution to 0.1 (from now on we call them effective users). Thus the effective number of users are $10^3, 10^4, 10^5, 10^6, 10^7$ accordingly. For the count problem, all effective users have a value $\mathbf{I}(u) = 1$ by definition. And for the sum problem, the values of effective users are

Problem Type	Data	Query Result	Query Time	Technique	Relative Error(%)	Time(s)
q_{1-}	Deezer	846,915	1.22	Naive	100	293.4
				Sampling	30.3	220.9
				PDP Query	**4.1**	**131.7**
$q_{\triangle}$		794,210	4.66	Naive	100	6,201
				Sampling	48.2	6,585
				PDP Query	**16.2**	**174.5**
Q_5	TPC-H	240,000	2.55	Naive	100	30.2
				Sampling	30.3	30.5
				PDP Query	**5.5**	**27.3**
Q_7		218,000,000	3.28	Naive	100	1,108
				Sampling	30.4	1,043
				PDP Query	**5.7**	**1,325**

Table 2. Summary of results for PDP query answering under default setting, the performance of our PDP mechanism is boldfaced.

sampled from a normal distribution with mean $1,000$ and standard deviation 200 independently. The sampled values are rounded to the nearest integer for convenience. We also tried varying values of $B = 10^5, 10^6, 10^7, 10^8, 10^9$ to demonstrate the effect of different global sensitivity bounds, and the default value shown in the table is 10^7.

Additionally, for the sum problem, we also examine a real dataset, namely the Bank Marketing dataset (Bank) [30] which contains 45,211 records. Each record represents a client's data and we treat each client as an effective user. We calculate the sum of clients' account balances, which ranges from -8, 109 to 102, 127, and $B = 10^7$ is the same as the default setting for synthetic data. The negative domain is dealt with as described in Section 7.2.

Queries. We examine two types of queries: graph pattern counting queries and SJA queries with foreign key constraints, where the former is a special subclass of the later. For graph pattern counting query, we use the Deezer dataset obtained from SNAP [25], which collects the friendships of users from the music streaming service provider Deezer. The Deezer dataset contains 144,000 nodes and 846,915 edges. We treat nodes as private users and run both edge counting query (q_{1-}) and triangle counting query ($q_{\triangle}$). For both queries, we set $B = |\mathcal{U}| = 10^6$.

For SJA query, we use the TPC-H dataset with scale factor 0.125, 0.25, 0.5, 1, 2, 4 and the default value is 1. TPC-H is a synthetic dataset consisting of eight relations: Region, Nation, Customer, Orders, Supplier, Part, PartSupp, and Lineitem, and the one with scale factor 1 contains about 7.5 million tuples. We treat suppliers and customers as private users and the user universe size is $160,000\times$ the scale factor.

The queries we test are analogs of Q5 and Q7 from the benchmark, here we only consider the SELECT-FROM-WHERE clause while dropping the GROUP BY clauses, which can also be handled as described in Section 6.1. Q5 is a counting query that counts the number of lineitems where supplier and customer are from the same nation, whereas Q7 is doing the sum aggregation over `extendedprice` and `discount` attributes in Lineitem relation, i.e., it calculates the total revenue from lineitems. Filter conditions are omitted for clarity of the query. For both queries, we set $B = 10^6$.

The aggregation values in Q7 are real numbers so we use the discretization approach as described in Section 7.1 with $\delta = 1$. The other three queries are all counting queries with integer values.

Privacy Specification. We try two sets of privacy specifications. The default setting is referred to as *Gaussian*, where we assume the users' privacy levels are i.i.d samples from a normal distribution

with mean 1 and standard deviation 0.3. To ensure a valid privacy specification, we set values smaller than $\frac{1}{\sqrt{|\mathcal{U}|}}$ to $\frac{1}{\sqrt{|\mathcal{U}|}}$ and set values greater than 2 to 2.

Another setting is referred from [2, 22], where we randomly divide the users into three groups: conservative, representing users with high privacy concern; moderate, representing users with medium concern; and liberal, representing users with low concern. The fraction of users in the conservative and moderate groups are set to 0.54 and 0.37, respectively and the remaining users are in the liberal group. The privacy level for the users in the conservative and moderate groups are drawn uniformly at random from the ranges $[\frac{1}{\sqrt{|\mathcal{U}|}}, 0.5]$and $[0.5, 2]$, respectively. The users in the liberal group have a fixed privacy level equal to 2. We call this setting as *Uniform*.

Experimental Parameters. All experiments are done on a desktop PC equipped with an M2 Pro CPU and 16GB memory. We always set the probability $\beta = 0.1$. Each experiment is repeated 20 times and we record its average running time and relative error compared to the real query result. Since the errors are unstable, we discard the top/lower 10% errors and compute the average of the remaining values, which reflects the failure probability $\beta = 0.1$.

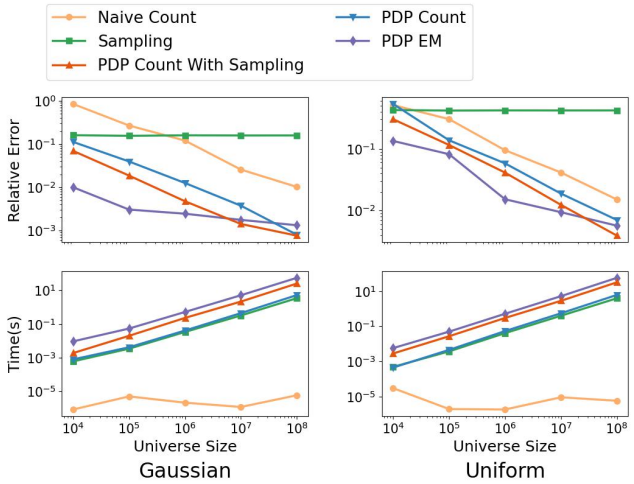

Fig. 3. Different methods for the count problem under varying universe sizes and privacy specifications, both axes are in log scale.

8.2 Results

Count and Sum. Under the default setting, all mechanisms can be implemented efficiently and the results are shown in Table 1. For the count problem, PDP EM [22], PDP Count, and PDP Count with sampling all achieve good performance, with only around 1% relative error. PDP EM performs the best among all methods, however, as shown in Figure 3 later, when the universe is large enough, the performance of PDP Count/with sampling will dominate. On the other hand, PDP EM can only be applied to this specific problem thus limiting its practicability. Additionally, we can see the improvement of adding a sampling procedure to our PDP count is around 2.4×, which is consistent with our result in Section 4.

For the sum problem, the naive method which calls the universal private estimator [12] with ε_{min} loses utility and leads to a 100% error, while our PDP Sum mechanism achieves good performance.

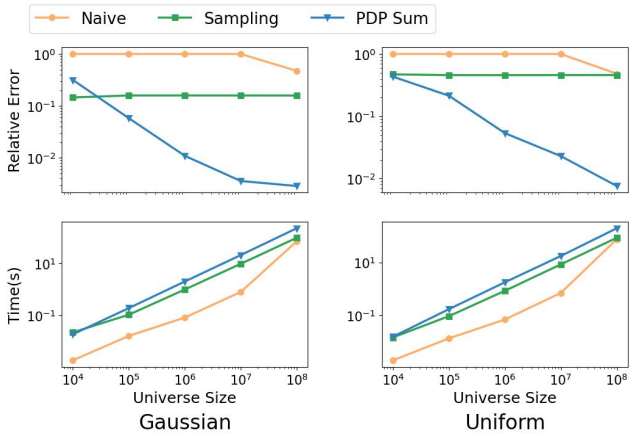

Fig. 4. Different methods for the sum problem under varying universe sizes and privacy specifications on the synthetic data, both axes are in log scale.

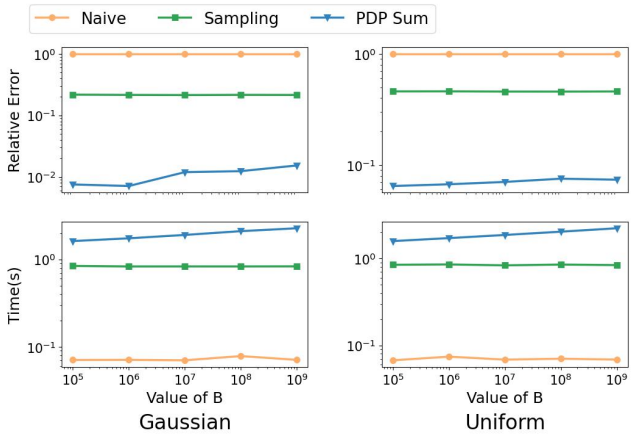

Fig. 5. Different methods for the sum problem under varying global sensitivity B and privacy specifications on the synthetic data, both axes are in log scale.

In comparison to the sampling mechanism, PDP Sum exhibits a utility improvement exceeding 10× on the synthetic dataset, whereas the enhancement is limited to 2× on the Bank dataset, which includes negative values. This discrepancy can be attributed to two primary factors. Firstly, PDP Sum partitions $\mathbf{I}$ into two sub-datasets, resulting in the final error being the aggregate of the errors from both segments. Secondly, the Bank dataset contains outliers (i.e., a small number of individuals with substantial wealth), which diminishes the accuracy of PDP Sum.

Figure 3 and 4 show the results under varying universe sizes and privacy specifications on the synthetic data. Since Bank has a fixed data size, we do not include it in this comparison. We can see the running time of all mechanisms grows roughly linearly with universe size, while the errors have different trends. For the count problem, the relative error of PDP EM [22], PDP Count, and PDP Count with sampling all decrease as $|\mathcal{U}|$ increases, with the latter two decreasing more rapidly. This is because the error of PDP EM is proportional to $\log |\mathcal{U}|$, while the other two methods have

errors independent of universe size. Thus when the universe size is large enough (which is likely the case in reality), our PDP Count/with sampling will provide the best performance. For the sum problem, PDP Sum performs the best for a large enough user universe, and its performance also has the best dependency on $|\mathcal{U}|$.

Figure 5 shows the effect of varying global sensitivity B. As analyzed in the main paragraphs, when B grows, the error/running time of PDP Sum grows roughly linearly in $\log B$. On the other hand, the naive/sampling approach is unaffected since their black-box standard DP estimator is independent of B.

Under different privacy specifications, the trend of error/running time for each mechanism is preserved. However, the accuracies under the *Uniform* setting are generally lower than that under the *Gaussian* setting, except for the naive mechanism. And the gap between them is around 3×. Since in the *Uniform* setting, the privacy levels are less concentrated thus more users are discarded/truncated. On the other hand, the running times under two different settings do not have significant differences.

One should note that, given a privacy specification, the error of the sampling mechanism roughly remains unchanged as the universe grows. This is because for a given threshold ε, the percentage of data that is discarded is roughly fixed. So as $|\mathcal{U}|$ increases, the number of discarded users also grows at the same speed. On the other hand, when the privacy specification changes from *Gaussian* to *Uniform* (which is less concentrated), the percentage of discarded users grows, leading to a greater relative error.

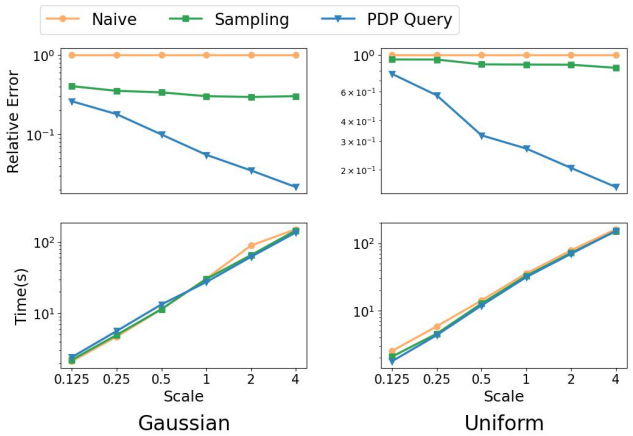

Fig. 6. Different methods for Q5 under varying universe sizes and privacy specifications.

Queries. The gap between our PDP methods and previous ones becomes more clear in the query answering problem. Here the naive mechanism is to call R2T [10] with privacy parameter ε_{min}.

According to Table 2 and Figure 6, 7, for both graph pattern counting queries and TPC-H queries, PDP Query achieves the best result under all data scales/privacy specifications while the naive method provides no utility in almost all cases. For instance, under the *Gaussian* setting and for TPC-H data with scale 4, our mechanism achieves a relative error of around 1%, whereas the sampling mechanism leads to around 30% error and the naive method has 100% error. In the meanwhile, the relative error of PDP Query decreases rapidly as the data scale increases. From Figure 6 and 7 we can see its dependency on $|\mathcal{U}|$ is roughly linear, whereas the other two methods have almost

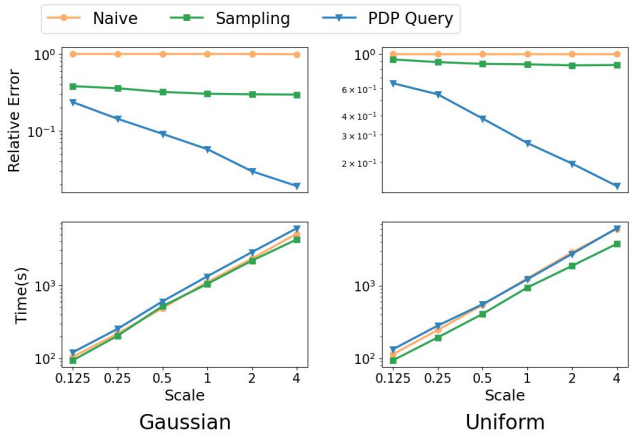

Fig. 7. Different methods for Q7 under varying universe sizes and privacy specifications.

fixed relative errors. The reason is the same as before, that is, the sampling mechanism discards a roughly fixed percentage of users.

Similar as before, trend of error/running time for each mechanism is preserved, while the accuracies under the *Uniform* setting are generally lower than that under the *Gaussian* setting. This time the gap can be as large as 7×, greater than that for the count/sum problem. For instance, consider Q5 at scale 4, under *Gaussian* setting, the error is around 2% while under *Uniform* setting the error is 15%. This is possibly because, in the query each user is related to more than one tuple, thus the result is more sensitive to the concentration of users' privacy levels.

There is a gap between the query time without DP and the execution time of our PDP Query algorithm, and the gap can be as large as two orders of magnitude. However, such an overhead is reasonable for PDP since the state-of-the-art standard DP algorithm [10] also incurs a similar overhead. Moreover, PDP offers a strictly stronger privacy protection compared to standard DP.

9 CONCLUSION AND FUTURE DIRECTIONS

In this paper, we study the fundamental sum estimation problem under personalized differential privacy (PDP). We develop the first PDP mechanisms with explicit error guarantees and conduct a series of experiments to demonstrate the advantages of our proposed mechanisms.

Although not concretely pointed out in the paper, it is noticeable that the PDP Count mechanism described in Section 4 can be extended to a general mechanism that transforms any sensitivity-based black box standard DP mechanism into a PDP version, by replacing the Laplace mechanism in line 5 of Algorithm 1 with the black box mechanism. For instance, when applied to the sum problem and using those truncation-based standard DP sum algorithms [10, 12, 20, 23, 24] as the black box, it achieves an error of roughly

$$\widetilde{O}\left(\min_{\varepsilon\in[\varepsilon_{\min},\varepsilon_{\max}]}\left(|\mathrm{Sum}(\bar{\mathbf{I}}_\varepsilon)-\mathrm{Sum}(\mathbf{I})|+\frac{\max_{u\in\mathcal{U}}\bar{\mathbf{I}}_\varepsilon(u)}{\varepsilon}\right)\right)$$

This is worse than the result of PDP Sum in Theorem 5.2. However, it is still much better than the sampling mechanism when the data scale is large. Thus one possible direction is to extend current standard DP results into PDP by inserting them into PDP Count.

ACKNOWLEDGMENTS

This work has been supported by HKRGC under grants 16205422, 16204223, and 16203924; by NTU-NAP start up grant. We would also like to thank the anonymous reviewers who have made valuable suggestions on improving the presentation of the paper.

REFERENCES

[1] Martin Abadi, Andy Chu, Ian Goodfellow, H Brendan McMahan, Ilya Mironov, Kunal Talwar, and Li Zhang. Deep learning with differential privacy. In *Proceedings of the 2016 ACM SIGSAC conference on computer and communications security*, pages 308–318, 2016.

[2] Alessandro Acquisti and Jens Grossklags. Privacy and rationality in individual decision making. *IEEE security & privacy*, 3(1):26–33, 2005.

[3] Mohammad Alaggan, Sébastien Gambs, and Anne-Marie Kermarrec. Heterogeneous differential privacy. *arXiv preprint arXiv:1504.06998*, 2015.

[4] Hilal Asi and John C Duchi. Instance-optimality in differential privacy via approximate inverse sensitivity mechanisms. *Advances in neural information processing systems*, 33:14106–14117, 2020.

[5] Ting Bao, Lei Xu, Liehuang Zhu, Lihong Wang, and Tielei Li. Successive point-of-interest recommendation with personalized local differential privacy. *IEEE Transactions on Vehicular Technology*, 70(10):10477–10488, 2021.

[6] Jeremiah Blocki, Avrim Blum, Anupam Datta, and Or Sheffet. Differentially private data analysis of social networks via restricted sensitivity. In *Proceedings of the 4th conference on Innovations in Theoretical Computer Science*, pages 87–96, 2013.

[7] Rui Chen, Haoran Li, A Kai Qin, Shiva Prasad Kasiviswanathan, and Hongxia Jin. Private spatial data aggregation in the local setting. In *2016 IEEE 32nd International Conference on Data Engineering (ICDE)*, pages 289–300. IEEE, 2016.

[8] Shixi Chen and Shuigeng Zhou. Recursive mechanism: towards node differential privacy and unrestricted joins. In *Proceedings of the 2013 ACM SIGMOD International Conference on Management of Data*, pages 653–664, 2013.

[9] Graham Cormode, Cecilia Procopiuc, Divesh Srivastava, Entong Shen, and Ting Yu. Differentially private spatial decompositions. In *2012 IEEE 28th International Conference on Data Engineering*, pages 20–31. IEEE, 2012.

[10] Wei Dong, Juanru Fang, Ke Yi, Yuchao Tao, and Ashwin Machanavajjhala. R2t: Instance-optimal truncation for differentially private query evaluation with foreign keys. In *Proceedings of the 2022 International Conference on Management of Data*, pages 759–772, 2022.

[11] Wei Dong, Dajun Sun, and Ke Yi. Better than composition: How to answer multiple relational queries under differential privacy. *Proceedings of the ACM on Management of Data*, 1(2):1–26, 2023.

[12] Wei Dong and Ke Yi. Universal private estimators. In *Proceedings of the 42nd ACM SIGMOD-SIGACT-SIGAI Symposium on Principles of Database Systems*, pages 195–206, 2023.

[13] Cynthia Dwork and Jing Lei. Differential privacy and robust statistics. In *Proceedings of the Forty-First Annual ACM Symposium on Theory of Computing*, page 371–380, 2009.

[14] Cynthia Dwork, Frank McSherry, Kobbi Nissim, and Adam Smith. Calibrating noise to sensitivity in private data analysis. In *Theory of cryptography conference*, pages 265–284. Springer, 2006.

[15] Cynthia Dwork, Aaron Roth, et al. The algorithmic foundations of differential privacy. *Foundations and Trends® in Theoretical Computer Science*, 9(3–4):211–407, 2014.

[16] Cynthia Dwork and Adam Smith. Differential privacy for statistics: What we know and what we want to learn. *Journal of Privacy and Confidentiality*, 1(2), 2010.

[17] Hamid Ebadi, David Sands, and Gerardo Schneider. Differential privacy: Now it's getting personal. *Acm Sigplan Notices*, 50(1):69–81, 2015.

[18] Juanru Fang, Wei Dong, and Ke Yi. Shifted inverse: A general mechanism for monotonic functions under user differential privacy. In *Proceedings of the 2022 ACM SIGSAC Conference on Computer and Communications Security*, pages 1009–1022, 2022.

[19] Bugra Gedik and Ling Liu. Protecting location privacy with personalized k-anonymity: Architecture and algorithms. *IEEE Transactions on Mobile Computing*, 7(1):1–18, 2007.

[20] Ziyue Huang, Yuting Liang, and Ke Yi. Instance-optimal mean estimation under differential privacy. *Advances in Neural Information Processing Systems*, 34:25993–26004, 2021.

[21] Noah Johnson, Joseph P Near, and Dawn Song. Towards practical differential privacy for sql queries. *Proceedings of the VLDB Endowment*, 11(5):526–539, 2018.

[22] Zach Jorgensen, Ting Yu, and Graham Cormode. Conservative or liberal? personalized differential privacy. In *2015 IEEE 31St international conference on data engineering*, pages 1023–1034. IEEE, 2015.

[23] Shiva Prasad Kasiviswanathan, Kobbi Nissim, Sofya Raskhodnikova, and Adam Smith. Analyzing graphs with node differential privacy. In *Theory of Cryptography: 10th Theory of Cryptography Conference, TCC 2013, Tokyo, Japan, March*

3-6, 2013. Proceedings, pages 457–476. Springer, 2013.
[24] Ios Kotsogiannis, Yuchao Tao, Xi He, Maryam Fanaeepour, Ashwin Machanavajjhala, Michael Hay, and Gerome Miklau. Privatesql: a differentially private sql query engine. *Proceedings of the VLDB Endowment*, 12(11):1371–1384, 2019.
[25] Jure Leskovec and Andrej Krevl. SNAP Datasets: Stanford large network dataset collection. http://snap.stanford.edu/data, 2014.
[26] Haoran Li, Li Xiong, Zhanglong Ji, and Xiaoqian Jiang. Partitioning-based mechanisms under personalized differential privacy. In *Advances in Knowledge Discovery and Data Mining: 21st Pacific-Asia Conference, PAKDD 2017, Jeju, South Korea, May 23-26, 2017, Proceedings, Part I 21*, pages 615–627. Springer, 2017.
[27] Ashwin Machanavajjhala, Daniel Kifer, Johannes Gehrke, and Muthuramakrishnan Venkitasubramaniam. l-diversity: Privacy beyond k-anonymity. *ACM Transactions on Knowledge Discovery from Data (TKDD)*, 1(1):3–es, 2007.
[28] Frank McSherry and Kunal Talwar. Mechanism design via differential privacy. In *48th Annual IEEE Symposium on Foundations of Computer Science (FOCS'07)*, pages 94–103. IEEE, 2007.
[29] Xuying Meng, Suhang Wang, Kai Shu, Jundong Li, Bo Chen, Huan Liu, and Yujun Zhang. Towards privacy preserving social recommendation under personalized privacy settings. *World Wide Web*, 22:2853–2881, 2019.
[30] Sérgio Moro, Paulo Cortez, and Paulo Rita. A data-driven approach to predict the success of bank telemarketing. *Decision Support Systems*, 62:22–31, 2014.
[31] Ben Niu, Yahong Chen, Boyang Wang, Jin Cao, and Fenghua Li. Utility-aware exponential mechanism for personalized differential privacy. In *2020 IEEE Wireless Communications and Networking Conference (WCNC)*, pages 1–6. IEEE, 2020.
[32] Yanguang Shen, Hui Shao, and Yan Li. Research on the personalized privacy preserving distributed data mining. In *2009 Second International Conference on Future Information Technology and Management Engineering*, pages 436–439. IEEE, 2009.
[33] Dajun Sun, Wei Dong, Yuan Qiu, and Ke Yi. Personalized truncation for personalized privacy. In *https://drive.google.com/file/d/1gsgBTSeXCQLiBsSce1KtwyrGm-8blVuq/view?usp=sharing*.
[34] Wei Wang, Lei Chen, and Qian Zhang. Outsourcing high-dimensional healthcare data to cloud with personalized privacy preservation. *Computer Networks*, 88:136–148, 2015.
[35] Xiaokui Xiao and Yufei Tao. Personalized privacy preservation. In *Proceedings of the 2006 ACM SIGMOD international conference on Management of data*, pages 229–240, 2006.
[36] Qiao Xue, Youwen Zhu, and Jian Wang. Mean estimation over numeric data with personalized local differential privacy. *Frontiers of Computer Science*, 16:1–10, 2022.
[37] Xiaojun Ye, Yawei Zhang, and Ming Liu. A personalized (a, k)-anonymity model. In *2008 The Ninth International Conference on Web-Age Information Management*, pages 341–348. IEEE, 2008.
[38] NIE Yiwen, Wei Yang, Liusheng Huang, Xike Xie, Zhenhua Zhao, and Shaowei Wang. A utility-optimized framework for personalized private histogram estimation. *IEEE Transactions on Knowledge and Data Engineering*, 31(4):655–669, 2018.
[39] Mingxuan Yuan, Lei Chen, and Philip S Yu. Personalized privacy protection in social networks. *Proceedings of the VLDB Endowment*, 4(2):141–150, 2010.
[40] Shufan Zhang and Xi He. Dprovdb: Differentially private query processing with multi-analyst provenance. *arXiv preprint arXiv:2309.10240*, 2023.

Received April 2024; revised July 2024; accepted August 2024