Shifted Inverse: A General Mechanism for Monotonic Functions under User Differential Privacy

Juanru Fang Wei Dong Ke Yi
Hong Kong University of Science and Technology
Hong Kong, China
{jfangad,wdongac,yike}@cse.ust.hk

ABSTRACT

While most work on differential privacy has focused on protecting the privacy of tuples, it has been realized that such a simple model cannot capture the complex user-tuple relationships in many real-world applications. Thus, user differential privacy (user-DP) has recently gained more attention, which includes node-DP for graph data as a special case. Most existing work on user-DP has only studied the sum estimation problem. In this work, we design a general DP mechanism for any monotonic function under user-DP with strong optimality guarantees. While our general mechanism may run in super-polynomial time, we show how to instantiate an approximate version in polynomial time on some common monotonic functions, including sum, k-selection, maximum frequency, and distinct count. Finally, we conduct experiments on all these functions and the results show that our framework is more general and obtains better results in many cases.

CCS CONCEPTS

• **Security and privacy → Database and storage security**.

KEYWORDS

Differential privacy; Monotonic function

ACM Reference Format:
Juanru Fang Wei Dong Ke Yi. 2022. 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 (CCS '22), November 7–11, 2022, Los Angeles, CA, USA.* ACM, New York, NY, USA, 14 pages. https://doi.org/10.1145/3548606.3560567

1 INTRODUCTION

Differential privacy (DP) has become the mainstream privacy standard due to its strong protection of individual users' information. It guarantees that the outputs of a DP mechanism on *neighboring* datasets, i.e., two datasets that differ only in one user's data, are statistically indistinguishable. Most existing work on differential privacy has considered the simple case where the dataset is given in a tabular form and each user possesses exactly one tuple in the table, thus two neighboring datasets differ by one tuple. In this case, there is a bijection between $\mathcal{U}$, the space of users, and $\mathcal{T}$, the space of tuples, so there is no distinction between user-level differential privacy (or simply, user-DP) and tuple-level differential privacy (tuple-DP).

In many applications, however, the relationship between the users and the tuples is more complicated than a simple bijection; in these cases user-DP becomes a more general notion than tuple-DP. First, a user may have multiple tuples in the dataset, which is a common scenario in practice as there is often significant skewness in data possession [2, 25]. In this case, the bijection between $\mathcal{U}$ and $\mathcal{T}$ becomes a one-to-many relationship. Note that DP should protect all the tuples belonging to any one user, i.e., two datasets are neighboring if one can be obtained from the other by deleting any subset of tuples of any one user. One can further generalize this relationship to be many-to-many, which models the case where each tuple is jointly contributed by a group of users and a user may contribute to multiple tuples. The canonical instantiation of this setting is counting the number of edges (or some graph pattern like triangles) of a graph under node-DP [6, 7, 15]. For this problem, the users are the nodes while the tuples are the edges (resp. triangles), and each edge (resp. triangle) is contributed by its two (resp. three) endpoints. Similarly, DP should protect all tuples contributed by any one user (solely or jointly with other users); indeed, node-DP defines neighboring instances as two graphs where one can be obtained from the other by deleting any subset of edges incident to one node. Finally, each group of users may jointly contribute more than one tuple; for graph data, this models the situation where multiple edges exist between a pair of nodes, each possibly associated with a different attribute value.

1.1 A General Model for User-DP

We use the following formalization to capture all the cases above. Let $\mathcal{U}$ be the universe of all users and $\mathcal{T}$ the universe of all tuples, both of which are public and possibly infinite. For an integer $\ell \geq 1$, we use $\binom{\mathcal{U}}{\leq \ell}$ to denote the set of all subsets of $\mathcal{U}$ of size at most ℓ. A dataset, or an instance, is a mapping $V : \binom{\mathcal{U}}{\leq \ell} \to \mathcal{T}^{\mathbb{N}}$, i.e., it maps each subset of no more than ℓ users to a multiset of tuples, representing the data jointly contributed by these users. Any instance V should have finite size, which is defined as

$$|V| = \sum_{x \in \binom{\mathcal{U}}{\leq \ell}} |V(x)|,$$

where $|V(x)|$ denotes the cardinality of the multiset $V(x)$. We also use $\mathcal{X}_V \subseteq \binom{\mathcal{U}}{\leq \ell}$ to denote the set of subsets of users mapped to

CCS '22, November 7–11, 2022, Los Angeles, CA, USA.

ACM ISBN 978-1-4503-9450-5/22/11…$15.00
https://doi.org/10.1145/3548606.3560567

non-empty multisets, i.e.,

$$\mathcal{X}_V = \left\{x \in \binom{\mathcal{U}}{\leq \ell} : V(x) \neq \varnothing\right\},$$

and use $\mathcal{U}_V \subseteq \mathcal{U}$ for the set of users with nonzero contributions, i.e.,

$$\mathcal{U}_V = \{u \in \mathcal{U} : \exists x \in \mathcal{X}_V, u \in x\}.$$

Neighboring instances are defined as follows. For any two non-identical instances V and V', we say that V *contains* V', denoted $V' \preceq V$, if $V'(x) \subseteq V(x)$ for any $x \in \binom{\mathcal{U}}{\leq \ell}$. We say that V' is a *down neighbor* of V if $V' \preceq V$ and there exists a user $u \in \mathcal{U}$, called the *witness*, such that all the differences between V and V' are contributed by u, i.e., for any $x \in \binom{\mathcal{U}}{\leq \ell}$,

$$V'(x) \subsetneq V(x) \Rightarrow u \in x.$$

Then, V and V' are neighboring instances if one is a down neighbor of the other, denoted as $V \sim V'$ or $V \sim_u V'$, where u is the witness. The distance, $d(V, V')$, between two instances is defined as the length of the shortest sequence $(V_0 = V, V_1, \ldots, V_d = V')$ such that $V_{i-1} \sim V_i$ for all $i = 1, \ldots, d$.

After neighboring instances are defined, user-DP follows the standard definition of differential privacy, i.e., a mechanism M is ε-DP if

$$\Pr[M(V) = y] \leq e^{\varepsilon} \cdot \Pr[M(V') = y],$$

for any $V \sim V'$ and any output y.

Our user-DP model captures all the aforementioned scenarios as special cases:

- If we set $\ell = 1$ and restrict $|V(x)| \in \{0, 1\}$ for every $u \in \mathcal{U}$, then the model degenerates into tuple-DP. In this case, $|\mathcal{U}_V| = |\mathcal{X}_V| = |V|$.
- The one-to-many case corresponds to $\ell = 1$ while $|V(x)|$ can be arbitrarily large (but still finite). For this case, $|\mathcal{U}_V| = |\mathcal{X}_V| \leq |V|$.
- Edge counting under node-DP can be incorporated by setting $\ell = 2$ and defining $V(\{u, v\}) := \{1\}$ if there is an edge between u and v, while $V(x) := \emptyset$ for all other x. For this case, $|\mathcal{U}_V|$ is the number of nodes and $|\mathcal{X}_V| = |V|$ is the number of edges in the graph.
- For triangle (or any other graph pattern) counting, we define $V(\{u, v, w\}) := \{1\}$ if u, v, w form a triangle, and $V(x) := \emptyset$ for all other x. For this case, $|\mathcal{U}_V|$ is the number of nodes and $|\mathcal{X}_V| = |V|$ is the number of triangles in the graph. Note that $|V|$ is not really the same as the size of the input graph. Nevertheless, as we will only differentiate between polynomial and super-polynomial running times, this polynomial difference in input size is not a concern.
- Many real problems on graph data rely on not only the topology of the graph, but also some additional attributes. For example, an edge (u, v) may represent a transaction between the two users, and multiple transactions involving different products may exist between the same pair of users. This can be captured in our model by defining, e.g., $V(\{u, v\}) := \{\text{product A, product B, product C}\}$. In this more general case, we have $|\mathcal{X}_V| \leq |V|$.
- Our model can also capture data structures more complicated than a graph. In general, the user-tuple relationship can take the form of a hypergraph: Each group x of up to ℓ nodes (users) corresponds to a hyperedge, and $V(x)$ specifies the multiset of attribute values contributed jointly by x. Examples include transactions that involve more than two parties (e.g., buyer, seller, and intermediary), or documents with multiple co-authors, etc.

Please see Section 6 for more concrete problems that can be formulated in our user-DP model. In fact, in Appendix A, we show that our model is equivalent to the user-DP model defined for relational databases with foreign key constraints [10, 18]. However, our user-DP model is more mathematically concise without needing to introduce the jargon from relational databases.

1.2 Monotonic Functions

The user-DP model above only concerns about privacy; for utility, we need to define what one wants to compute from V. For this purpose, it is convenient to take $\mathcal{T} = \mathbb{N}$, i.e., each $V(x)$ is a multiset of natural numbers. This is without loss of generality, since any tuple can be encoded as a number. We use $[n]$ to denote the set $\{0, 1, ..., n\}$. Let $S(V) = \biguplus_x V(x)$ be the multiset of all data, and $S(V, u) = \biguplus_{x \ni u} V(x)$ the data contributed (solely or in part) by user u. Note that for $\ell = 1$, $S(V) = \biguplus_u S(V, u)$; for $\ell \geq 2$, we have $\mathrm{Supp}(S(V)) = \mathrm{Supp}(\biguplus_u S(V, u))$ but the multiplicities in the latter are generally higher.

For a given $f : \mathbb{N}^{\mathbb{N}} \to [D]$, we wish to design an ε-DP mechanism M to approximate $f(V) := f(S(V))$. The assumption on such a D given *a priori* is a mild one, as all our utility bounds will depend on D logarithmically. Also, we consider a running time to be polynomial if it is a polynomial in $|V|, \frac{1}{\varepsilon}, \log D$. We require f to be *monotonic*, i.e., for any two multisets $S' \subseteq S$, we have $f(S') \leq f(S)$. Monotonicity is a basic property of many commonly encountered functions; all functions covered in Section 5 are monotonic. Exceptions exist, such as the mean of $S(V)$, but it can be computed from the sum $\sum_{t \in S(V)} t$ and the count $|S(V)| = |V|$, both of which are monotonic. Finally, by symmetry, all our results hold if monotonicity is defined as $S' \subseteq S$ implying $f(S') \geq f(S)$.

1.3 Down Neighborhood Optimality

The main challenge under user-DP is that nearly every non-trivial f has large sensitivity, which is defined as $\Delta f := \max_{V \sim V'} |f(V) - f(V')|$. Even for the simple counting function $f(V) = |V|$ with $\ell = 1$, we already have $\Delta f = D$, by taking $V(\cdot) \equiv \emptyset$, and setting V' to be the same as V except $V'(\{u\}) = \{1, 2, \ldots, D\}$ for some user u. This makes worst-case optimality meaningless: Simply invoking the standard Laplace mechanism with noise calibrated to $\Delta f/\varepsilon = D/\varepsilon$ is already worst-case optimal, since this amount of noise is necessary to protect this particular pair of neighboring instances. However, this noise magnitude clearly eliminates any utility.

Thus, most mechanisms under user-DP add instance-specific noise [8, 17, 18, 22, 25] with magnitude much smaller than Δf on most typical instances. However, the theoretical justification of their superiority over the naive worst-case optimality had remained an open problem. The strongest notion of optimality is *instance optimality*. Unfortunately, as pointed out by Asi and Duchi [3], instance optimality is impossible to achieve (see Section 3.2 for details), who then proposed a natural relaxation, where we do not

compare M against the best mechanism tailor-designed for each instance V itself (which can always be perfect), but against the best mechanism M' that must do well in a small neighborhood of V within distance ρ. This *neighborhood optimality* avoids the issue of a large Δf for certain problems, but does not address the issue caused by user-DP. Still consider the counting function above. As long as $f(V) \le D/2$ (which is the typical case as we often set D conservatively large), then V will have a neighbor V' such that $f(V') = D$ and $|f(V) - f(V')| \ge D/2$. Then, even the best M' designed just for V and V' has to inject noise $\Omega(D)$, so the trivial Laplace mechanism above is also neighborhood optimal. Equivalently speaking, on most instances under user-DP, neighborhood optimality is the same as worst-case optimality, both of which are meaningless.

Recently, Dong et al. [10] proposed to restrain the neighborhood in order to obtain a meaningful notion of optimality under user-DP, referred as *down neighborhood optimality*. The formal definition is given in Section 3.2, but we give the intuition below. Again consider the counting function. The problem with neighborhood optimality manifested above is that almost every typical instance V has a bad neighbor V' with $|f(V) - f(V')| = \Omega(D)$, which makes achieving neighborhood optimality too easy, hence meaningless. It is observed that this bad neighbor V' must contain V, namely, it is an up neighbor. The idea is thus to exclude all such up neighbors when considering the best M', which now only needs to work well on V and its down neighborhood of distance ρ. For the counting function, it can be shown that such a best M' can inject noise proportional to $\max_u |S(V,u)|$ instead of $O(D)$. Thus, a *down neighborhood optimal* M must match this, possibly up to some ratio c, known as the *optimality ratio*. This is much more meaningful, as $\max_u |S(V,u)| \ll D$ on most typical V. But it is also more challenging, since we use the same M to compare with all different M', each designed specifically for each different instance V (and its down neighborhood).

1.4 Our Results

The user-DP mechanism of [10] only works for the counting function and the sum function, which are actually equivalent under user-DP. In this paper, we design a general user-DP mechanism (Section 4) based on a shifted version of the inverse sensitivity mechanism [3, 4]. It works with any monotonic function f, and achieves down neighborhood optimality with both ρ and c being at most $O(\frac{1}{\varepsilon}\log D)$.

Without any assumptions on f other than monotonicity, our general mechanism runs in $|V|^{O(\frac{1}{\varepsilon}\log D)}$ time. Next, we investigate a number of fundamental functions of interest, and show how to instantiate the general framework so that it runs in polynomial time. In doing so, we not only need to exploit the special properties of each function, in some cases we also have to work with an approximate version of the general mechanism. Nevertheless, we show that the approximation does not degrade the down neighborhood optimality by more than a constant factor (for constant ℓ); in fact, for certain functions, we are able to tighten up the analysis and improve either ρ or c to $O(1)$ or even 1. Specifically, we obtain the following results (details given in Section 5):

(1) Sum/count: Our mechanism runs in polynomial time and achieves $\rho = 1$ and $c = O(\frac{1}{\varepsilon}\log D)$. The previous mechanism [10] achieves $(1, O(\frac{1}{\varepsilon}\log R \log\log R))$-down neighborhood optimality, where R is a given upper bound on $\sum_{t\in S(V,u)} t$. Since $R \le D$, these two results are incomparable.
(2) k-selection: For a given k, the function f returns the k-th largest value in $S(V)$. An approximate version of our mechanism can run in polynomial time and achieve $\rho = O(\frac{\ell}{\varepsilon}\log D), c = O(\frac{\ell}{\varepsilon}\log D)$. For the special case $k = 1$, i.e., finding the maximum value, c can be reduced to $O(1)$. For this problem, the previous mechanism [27] only works for the case $\ell = 1$ and it does not offer any optimality guarantee. By symmetry, the mechanism also works for finding the k-th smallest value.
(3) Frequency moments: Let $\psi_i(V)$ be the multiplicity of i in $S(V)$. For some integer $k \ge 1$, the k-th frequency moment is defined as $F_k(V) = \sum_i \psi_i(V)^k$, which is an important statistic on the frequency distribution, and extensively studied in the literature [1]. Note that $F_1(V) = |V|$. We show how to achieve $\rho = 1, c = O(\frac{1}{\varepsilon}\log D)$ in polynomial time for any integer $k \ge 1$.
(4) Distinct count: The function returns $|\mathrm{Supp}(S(V))|$, which can also be considered as $F_0(V)$. Our general mechanism has $\rho = O(\frac{1}{\varepsilon}\log D), c = O(\frac{1}{\varepsilon}\log D)$, but it runs in super-polynomial time. We design a more efficient version that runs in polynomial time and show that it satisfies DP. Although we cannot give an optimality guarantee, it performs well in the experiments.
(5) Maximum frequency: The function returns $\max_i \psi_i(V)$, which can also be considered as $F_\infty(V)$. For this function, our general mechanism has optimality guarantee but it runs in super-polynomial time. We also show how to implement its approximate version in polynomial time but without an optimality guarantee. As a further application, the maximum degree of a graph under node-DP can be formulated as a maximum frequency problem, by setting $V(\{u,v\}) := \{u,v\}$ for each edge (u,v).

2 RELATED WORK

The general user-DP model described in Section 1.1 has previously been adopted by [7, 10, 18, 25], but they frame their model as SPJA queries in a relational database with foreign-key constraints. Our formulation is equivalent to theirs, but is more mathematically concise without relying on any background knowledge on relational databases. They have mostly focused on the sum/count function, which is a very special case of monotonic functions. A few other specific monotonic functions, such as distinct count, have been studied in [7, 10, 27], but they do not generalize to arbitrary monotonic functions.

Various special cases of the general user-DP model have been studied in the literature. The case $\ell = 1$ has drawn particular attention. Recall that $\ell = 1$ implies that different users' data are disjoint, but users may contribute different amounts of data. A common technique for this case is the *truncation* mechanism. For count, each $V(x)$ is truncated to have size $\min\{\tau, |V(x)|\}$ for some truncation threshold τ; for sum, $V(x)$ is truncated to have sum $\min\{\tau, \sum_{t\in V(x)} t\}$. After truncation, Δf can be bounded, then a standard mechanism, such as Laplace or Gaussian, can be applied.

A key technical issue is thus how to pick an optimal truncation threshold. Various threshold-selection techniques have been proposed [2, 13, 18, 25, 27], but they only work for the sum/count function. Similar problems have also been studied in statistical machine learning on private data [5, 12, 19, 20]. They typically assume that each $|V(x)|$ is already bounded, and data in the $V(x)$'s are i.i.d. samples from some distribution. We do not make these assumptions.

Moving from $\ell = 1$ to $\ell \geq 2$ presents another challenge that is caused by the interdependence of the users. In particular, the (naive) truncation mechanism above no longer works, as the users' data cannot be truncated independently. Node-DP, which is a special case of user-DP where each $V(x)$ is restricted to contain only one value, has received a lot of attention in private graph analytics [6, 9, 15, 24]. Note that we always have $\ell \geq 2$ for node-DP problems, although the specific value of ℓ depends on the particular graph problem. For example, $\ell = 2$ for edge counting, maximum degree, and degree distribution; $\ell = 3$ for triangle counting, which also generalizes to counting any graph pattern with ℓ nodes. As we have seen, all these problems are special cases of our results, except degree distribution. Our current framework only supports a monotonic f with a scalar output. It would be interesting to see if it can be extended to vector-valued functions such as the degree distribution.

3 PRELIMINARIES

3.1 Differential Privacy

The canonical DP-mechanism for a scalar-valued function f is the Laplace mechanism. Let Δf denote the sensitivity of the function f, i.e.,

$$\Delta f = \max_{V \sim V'} |f(V) - f(V')|,$$

the Laplace mechanism adds noise sampled from the Laplace distribution with scale $\frac{\Delta f}{\varepsilon}$.

Theorem 3.1 (The Laplace Mechanism [11]). *The Laplace mechanism outputs $M(V) = f(V) + \mathrm{Lap}(\frac{\Delta f}{\varepsilon})$ on instance V. It preserves ε-differential privacy.*

A more general DP mechanism that allows an arbitrary output range $\mathcal{R}$ is the exponential mechanism:

Definition 1 (The Exponential Mechanism [21])**.** Given an instance V and an output range $\mathcal{R}$, the exponential mechanism assigns a utility score $s(V, r)$ for each possible output $r \in \mathcal{R}$, and samples the output with probability proportional to $\exp(\frac{\varepsilon s(V,r)}{2\Delta s})$, where Δs is the sensitivity of the utility score, i.e.,

$$\Delta s = \max_{r \in \mathcal{R}} \max_{V \sim V'} |s(V, r) - s(V', r)|.$$

Theorem 3.2. *The exponential mechanism preserves ε-differential privacy, and with probability at least $1 - \beta$, the output $M(V)$ satisfies*

$$s(V, M(V)) \geq \mathrm{OPT}_s(V) - \frac{2\Delta s}{\varepsilon} \ln\left(\frac{|\mathcal{R}|}{\beta}\right),$$

where $\mathrm{OPT}_s(V) = \max_{r \in \mathcal{R}} s(V, r)$.

To instantiate the exponential mechanism for a given f, a popular choice is to set the utility score as the *inverse sensitivity* of f:

Definition 2 (Inverse Sensitivity [3])**.** Given an output $r \in \mathcal{R}$, the inverse sensitivity $\mathrm{len}(V, r)$ measures the shortest distance from V to an instance $\bar{V}$ such that $f(\bar{V}) = r$, i.e.,

$$\mathrm{len}(V, r) = \min_{\bar{V} \in \mathcal{V}} \{d(V, \bar{V}) : f(\bar{V}) = r\}.$$

It should be clear that as long as $d(\cdot, \cdot)$ takes non-negative integer values and forms a metric, then $\Delta\mathrm{len}(\cdot, r) \leq 1$ for any r. This yields the *inverse sensitivity mechanism* (Algorithm 1). The privacy and utility guarantee follow from Theorem 3.2.

Algorithm 1: `Inverse`

Input : The instance V, a function f, an output range $\mathcal{R}$, and the privacy budget ε
Output : A privatized $f(V)$
Invoke the exponential mechanism with $s(V, r) = -\mathrm{len}(V, r)$.

One important property of differential privacy is the composition theorem, which allows one to design DP mechanisms in a modular fashion.

Theorem 3.3 (Basic Composition Theorem [11]). *Let M be an adaptive composition of $M_1, ..., M_k$, where each M_i is ε_i-differential private, then mechanism M is $(\sum_{i=1}^{k} \varepsilon_i)$-differential private.*

3.2 Neighborhood Optimality and Down Neighborhood Optimality

Let $\mathcal{M}$ be the class of all ε-differential private mechanisms. For a given function f, the ρ-neighborhood lower bound [3] at V is defined as[1]

$$\mathcal{L}(V, \rho) := \inf_{M' \in \mathcal{M}} \max_{\bar{V}, d(V,\bar{V}) \leq \rho} \inf \left\{\xi : \Pr[|M'(\bar{V}) - f(\bar{V})| \leq \xi] \geq \frac{2}{3}\right\}, \tag{1}$$

i.e., it is the minimax constant-probability error bound achievable by any ε-DP mechanism M' in the ρ-neighborhood of V. Note that the $\frac{2}{3}$ probability in (1) is not important; it can be replaced by any constant between $\frac{1}{2}$ and 1.

Definition 3 (Neighborhood Optimality)**.** An ε-differential private mechanism M is (ρ, c)*-neighborhood optimal* if for any instance V,

$$\Pr[|M(V) - f(V)| \leq c \cdot \mathcal{L}(V, \rho)] \geq \frac{2}{3},$$

where c is called the *optimality ratio*.

It should be clear that smaller values of ρ, c correspond to stronger notions of optimality. The two extreme cases $\rho = 0$ and $\rho = \infty$ correspond to instance optimality and worst-case optimality, respectively. However, if $\rho = 0$, then for any V there is a trivial, perfectly private mechanism $M'(\cdot) \equiv f(V)$. This means that $\mathcal{L}(V, 0) = 0$ for all V, but it is impossible for any M to match this lower bound on all V. So instance-optimality is unattainable, and one can only hope for a small (constant or logarithmic) ρ.

[1] Asi and Duchi [3] defined $\mathcal{L}$ using the expected error $\mathbf{E}[|M'(\bar{V}) - f(\bar{V})|]$; here we use a constant-probability version to be consistent with the upper bounds. These two versions of $\mathcal{L}$ only differ by a constant factor.

User-DP, however, poses a problem for neighborhood optimality. Recall the argument in Section 1.3 on the simple counting function and $\ell = 1$. Because a user may contribute an arbitrarily large $V(x)$, we have $\mathcal{L}(V, 1) = \Omega(D)$ for any V with $f(V) \le D/2$. This means that even $(1, c)$-neighborhood optimality already degenerates into worst-case optimality, i.e., neighborhood optimality does not offer a smooth spectrum between instance optimality and worst-case optimality.

To address the issue, Dong et al. [10] proposed a simple fix: restrict the neighborhood in (1) so that only instances contained in V are considered, i.e., $\mathcal{L}$ is redefined as

$$\mathcal{L}(V, \rho) := \inf_{M' \in \mathcal{M}} \max_{\bar{V}, \bar{V} \le V, d(V, \bar{V}) \le \rho} \inf \left\{ \xi : \Pr[|M'(\bar{V}) - f(\bar{V})| \le \xi] \ge \frac{2}{3} \right\}. \tag{2}$$

Down neighborhood optimality is the same as in Definition 3 except that (2) is used for $\mathcal{L}$, which will be used as the default definition of $\mathcal{L}$ from now on. Down neighborhood optimality offers a more meaningful instance-specific optimality under user-DP. For the counting function under $\ell = 1$, it can be shown that $\mathcal{L}(V, 1) = \Theta(\max_u |S(V, u)|)$, which exactly captures the hardness of each instance V: If V contains a user with a large $|S(V, u)|$, then it is necessary to inject more noise to protect the privacy of this user; if $|S(V, u)| \in \{0, 1\}$, then user-DP degenerates into tuple-DP and $O(1)$ noise suffices. The challenge, of course, is to match this optimal noise level in a DP fashion. In particular, $\max_u |S(V, u)|$ cannot be used directly as it is sensitive information.

3.3 Downward Local Sensitivities

The definition of $\mathcal{L}$ is not easy to work with. Below we introduce a lower bound on $\mathcal{L}$ as a proxy; later we will design mechanisms to match this lower bound instead.

Definition 4 (Downward Local Sensitivity)**.** Given an f and an instance V, the *downward local sensitivity* of f at V is

$$\mathrm{DS}(V) = \max_{V', V \sim V', V' \le V} |f(V) - f(V')|,$$

and its *downward local sensitivity at distance k* is

$$\mathrm{DS}^{(k)}(V) = \max_{\bar{V}, \bar{V} \le V, d(V, \bar{V}) \le k} \mathrm{DS}(\bar{V}).$$

In particular, $\mathrm{DS}^{(0)}(V) = \mathrm{DS}(V)$.

A lower bound on $\mathcal{L}$ is given below:

Theorem 3.4. *For any $\varepsilon \le \ln 2$, any f, ρ, and V, $\mathcal{L}(V, \rho) \ge \frac{1}{2}\mathrm{DS}^{(\rho-1)}(V)$.*

Proof. The proof is based on the idea in [26]. Let

$$V_1 = \operatorname*{argmax}_{\bar{V}, \bar{V} \le V, d(V, \bar{V}) \le \rho - 1} \mathrm{DS}(\bar{V}),$$

and

$$V_2 = \operatorname*{argmax}_{V', V_1 \sim V', V' \le V_1} |f(V_1) - f(V')|,$$

so that $\mathrm{DS}^{(\rho-1)}(V) = \mathrm{DS}(V_1) = |f(V_1) - f(V_2)|$. Let $M(\cdot)$ be an ε-differential private mechanism, and for $b = 1$ or 2, define

$$\mathcal{G}_b = \{r \in \mathcal{R} : |r - f(V_b)| < \frac{1}{2}\mathrm{DS}(V_1)\}.$$

Algorithm 2: `ShiftedInverse`

Input : The instance V, a monotonic function $f : \mathcal{V} \to [D]$, the privacy budget ε, and failure probability β

Output : A privatized $f(V)$

$\tau \leftarrow \lceil \frac{2}{\varepsilon} \ln(\frac{D+1}{\beta}) \rceil$;

for $j \in [2\tau]$ **do**

  Compute $\check{f}(V, j) = \min_{\bar{V}, \bar{V} \le V, d(V, \bar{V}) \le j} f(\bar{V})$;

end

For $r \in [D]$, set $s(V, r) =$

$$\begin{cases} \tau - j, & \text{if } r \in [\check{f}(V, j), \check{f}(V, j-1)) \text{ for some } \tau < j \le 2\tau \\ 0, & \text{if } r = \check{f}(V, \tau) \\ -\tau + j - 1, & \text{if } r \in (\check{f}(V, j), \check{f}(V, j-1)] \text{ for some } 0 < j \le \tau \\ -\tau - 1, & \text{otherwise} \end{cases}$$

Sample an r from $[D]$ with probability proportional to $\exp(\frac{\varepsilon}{2} s(V, r))$, denoted by $\tilde{r}$;

return $M(V) = \tilde{r}$;

Let

$$p = \min(\Pr(M(V_1) \in \mathcal{G}_1), \Pr(M(V_2) \in \mathcal{G}_2)).$$

We have

$$1 - p \ge \Pr(M(V_1) \notin \mathcal{G}_1) \ge \Pr(M(V_1) \in \mathcal{G}_2) \ge e^{-\varepsilon} \Pr(M(V_2) \in \mathcal{G}_2) \ge e^{-\varepsilon} p$$

Therefore, $p \le \frac{1}{1+e^{-\varepsilon}}$. When $\varepsilon \le \ln 2$, we have $p \le \frac{2}{3}$ and $\mathcal{L}(V, \rho) \ge \frac{1}{2}\mathrm{DS}^{(\rho-1)}(V)$. □

For the sum/count function, it can be shown that $\mathrm{DS}^{(\rho-1)}(V) = \mathrm{DS}(V)$ for all ρ, i.e., any $\bar{V}$ contained in V is no harder than V, which is determined by the largest $S(V, u)$. This means that for sum/count, down neighborhood optimality is not sensitive to ρ. For other functions, as we will see, this is not the case.

4 A GENERAL MECHANISM FOR MONOTONIC FUNCTIONS

4.1 The Mechanism

The inverse sensitivity mechanism can be used to handle a general f. However, there are two issues when applying it under user-DP. First, the utility can be arbitrarily bad. Still consider the counting function $f(V) = |V|$ for $\ell = 1$. For any output $r \in (|V|, D]$, one can always add a user to V, obtaining a $\bar{V}$ such that $|\bar{V}| = r$, namely, $\mathrm{len}(V, r) = 1$. This makes all such r equally likely to be sampled by the exponential mechanism, resulting in an error of $\Omega(D)$ with constant probability. This is actually the main reason why the inverse sensitivity mechanism has only been used for tuple-DP problems, for which adding a user can only change $|V|$ by 1, so that $r > |V|$ have large $\mathrm{len}(V, r)$, hence exponentially less likely to be sampled.

Since adding users is too "powerful", one idea is to consider only deleting users when defining $\mathrm{len}(V, r)$. This will set $\mathrm{len}(V, r) = \infty$ for all $r \in (|V|, D]$, which avoids the issue above, but now there exists neighboring instance $V' \sim V$ such that $|V'| > |V|$ and $\Delta\mathrm{len}$ becomes unbounded for any $r \in (|V|, |V'|]$, thus the exponential

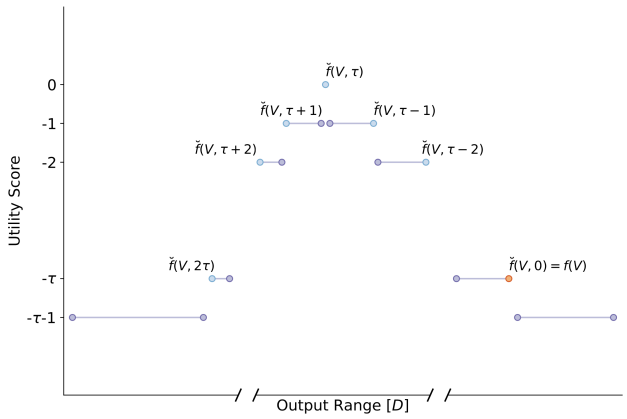

Figure 1: An illustration of the score function $s(V,r)$ used in `ShiftedInverse`. The optimal answer $\check{f}(V,0)=f(V)$ has a score of $-\tau$, while $\check{f}(V,\tau)$ has the highest score 0.

mechanism still has no utility. Our idea to fix this issue is a rather counter-intuitive one: In the exponential mechanism, one usually assigns the highest score, e.g., 0, to the best answer $r=f(V)$. In our mechanism, we intentionally shift our target down, assigning the highest score to an $r=f(\bar{V})$ such that $\bar{V}\preceq V$ and $d(V,\bar{V})=\tau$, for an appropriately chosen τ. Then with $r=f(\bar{V})$ at the center, we reduce the scores gradually in both directions (see Figure 1). While this makes sense for $r<f(\bar{V})$, for $r\in(f(\bar{V}),f(V)]$, it actually assigns smaller scores to better answers, and the best answer $r=f(V)$ has (almost) the lowest score. This counter-intuitive scoring mechanism is needed to ensure $\Delta\text{len}\le 1$ and all $r\in(f(V),D]$ have low scores (we can just set their scores equal to that of $r=f(V)$ minus 1) so that the exponential mechanism has good utility. Indeed, our analysis shows that $\tau=O(\frac{1}{\varepsilon}\log\frac{D}{\beta})$ is sufficient. This ensures that we hit a close-to-optimal target with high probability, although the probability of hitting the optimal target is low.

The second issue is computational. To apply the inverse sensitivity mechanism, one generally needs to compute the inverse sensitivity for every $r\in[D]$, which takes super-polynomial time. Instead, we only compute $O(\log D)$ inverse sensitivities, and "interpolate" others. More precisely, for each $j\in[2\tau]$, we compute

$$\check{f}(V,j)=\min_{\bar{V},\bar{V}\preceq V,d(V,\bar{V})\le j} f(\bar{V}).$$

Note that $\check{f}(V,0)=f(V)$. By definition, $\text{len}(V,\check{f}(V,j))=j$. For $r\in(\check{f}(V,j),\check{f}(V,j-1))$, we set $\text{len}(V,r)$ to j or $j-1$, depending on whether it is on the left or right side of $\check{f}(V,\tau)$. The details are given in Algorithm 2; also see Figure 1. Note that the true $\text{len}(V,r)$ may be different for the r's, but we show that it does not affect the privacy or utility of our mechanism due to the monotonicity of f.

Given the $\check{f}(V,j)$'s, `ShiftedInverse` can be implemented in time $O(\tau)=O(\frac{1}{\varepsilon}\log\frac{D}{\beta})$: We do not really need to compute $s(V,r)=-\text{len}(V,r)$ for every r. Since all r's in the same interval delineated by the $\check{f}(V,j)$'s have the same score, we can implement the mechanism by first sampling an interval, followed by sampling an r uniformly within the interval.

4.2 Analysis

For down neighborhood optimality, it suffices to use $\beta=\frac{1}{3}$. Let $\tau_0=\frac{2}{\varepsilon}\ln(3D+3)$ by setting $\beta=\frac{1}{3}$ in τ. The analysis below also needs the corresponding instance which attains $\check{f}(V,j)$, where ties are broken by taking the instance with the smallest size, i.e., define

$$\check{V}_j=\operatorname*{argmin}_{\bar{V}}\{|\bar{V}|:\bar{V}\preceq V,d(V,\bar{V})\le j,f(\bar{V})=\check{f}(V,j)\}.$$

Lemma 4.1. *For any monotonic f, any $V\sim V'$, and any r, $|s(V,r)-s(V',r)|\le 1$.*

Proof. Given two neighboring instances $V\sim_{u^*}V'$, without loss of generality, assume $V'\preceq V$. We prove that $\check{f}(\cdot,j)$'s are smooth for any neighboring instances, i.e., for any j,

$$\check{f}(V,j+1)\le\check{f}(V',j)\le\check{f}(V,j). \tag{3}$$

Thus, $s(V,r)-1\le s(V',r)\le s(V,r)+1$ for all $r\le f(V)$. Moreover, when $r>f(V)$, $s(V,r)=s(V',r)=-\tau-1$. Therefore, the sensitivity of $s(\cdot,r)$ is 1.

We now prove (3). It is trivial for $j>2\tau$. Consider any $j\in[2\tau]$. On the one hand, given the instance $\check{V}'_j$, we have $\check{V}'_j\preceq V'\preceq V$ and $d(V,\check{V}'_j)\le d(V,V')+d(V',\check{V}'_j)\le j+1$. Therefore, $\check{f}(V,j+1)\le f(\check{V}'_j)=\check{f}(V',j)$. On the other hand, given the instance $\check{V}_j$, denote U as the subset of users such that $d(V,\check{V}_j)=|U|\le j$. If $u^*\in U$, all the multisets contributed by u^* are empty in $\check{V}_j$ due to the monotonicity and the fact that we break the tie by choosing the instance with the smallest size, i.e., for any $x\in\binom{\mathcal{U}}{\le\ell}$ such that there exists $u\in U,u\in x$, we have $V(x)=\emptyset$. Therefore, $\check{V}_j\preceq V'$. Moreover, $d(V',\check{V}_j)\le|U|\le j$, thus $\check{f}(V',j)\le f(\check{V}_j)=\check{f}(V,j)$. If $u^*\notin U$, construct $\tilde{V}'_j$ from V' by setting all the multisets contributed by any user $u\in U$ as empty set, i.e., set $V(x)=\emptyset$ for all $x\in\binom{\mathcal{U}}{\le\ell}$ such that there exists $u\in U,u\in x$. We have $\tilde{V}'_j\preceq\check{V}_j$, $\tilde{V}'_j\preceq V'$ and $d(V',\tilde{V}'_j)\le|U|\le j$. Therefore, $\check{f}(V',j)\le f(\tilde{V}'_j)\le f(\check{V}_j)=\check{f}(V,j)$. □

Lemma 4.2. *Given an instance V, `ShiftedInverse` preserves ε-differential privacy, and with probability at least $1-\beta$, the output $M(V)$ satisfies $s(V,M(V))\ge-\tau$, i.e.,*

$$\check{f}(V,2\tau)\le M(V)\le f(V).$$

Proof. Follows from Theorem 3.2 and Lemma 4.1. □

Lemma 4.3. *If for any instance V, there exists an instance $\bar{V}\preceq V$ such that $d(V,\bar{V})+1\le\rho$ and*

$$f(V)-\check{f}(V,2\tau_0)\le c\cdot\text{DS}(\bar{V})$$

for some c, `ShiftedInverse` is $(\rho,2c)$-down neighborhood optimal.

Proof. Let $\bar{V}$ be the instance. On the one hand, by Theorem 3.4,

$$\mathcal{L}(V,\rho)\ge\frac{1}{2}\text{DS}^{(\rho-1)}(V)\ge\frac{1}{2}\text{DS}(\bar{V}).$$

On the other hand, by Lemma 4.2, with probability at least $\frac{2}{3}$,

$$|M(V)-f(V)|\le f(V)-\check{f}(V,2\tau_0)\le c\cdot\text{DS}(\bar{V}).$$

Therefore, $|M(V)-f(V)|\le 2c\cdot\mathcal{L}(V,\rho)$ and the algorithm is $(\rho,2c)$-down neighborhood optimal. □

Theorem 4.4. *For any monotonic f,* `ShiftedInverse` *is $(2\tau_0, 4\tau_0)$-down neighborhood optimal.*

Proof. Given the instance $\check{V}_{2\tau_0}$, denote U as the set of users such that $d(V, \check{V}_{2\tau_0}) = |U|$. Assume an arbitrary permutation is given for the users in U, denoted by $\{u_1, ..., u_{|U|}\}$. A path of neighboring instances from V to $\check{V}_{2\tau_0}$ can be constructed as follows. Let $V_0 = V$. For $i = 1, ..., |U|$, V_i is the instance constructed from V_{i-1} by setting all the multisets contributed by u_i as empty set, so that $V_{|U|} = \check{V}_{2\tau_0}$.

For any neighboring instance $V_{i-1} \sim V_i, i = 1, ..., |U|$, we have

$$\mathrm{DS}(V_{i-1}) \geq f(V_{i-1}) - f(V_i).$$

Therefore,

$$f(V) - \check{f}(V, 2\tau_0) = \sum_{i=1}^{|U|} f(V_{i-1}) - f(V_i) \leq |U| \cdot \max_{i=1,...,|U|} \mathrm{DS}(V_{i-1}).$$

Let $\bar{V} \in \{V_0, ..., V_{|U|-1}\}$ be the one with the largest downward local sensitivity, we have $d(V, \bar{V}) + 1 \leq |U| \leq 2\tau_0$ and $f(V) - \check{f}(V, 2\tau_0) \leq |U| \cdot \mathrm{DS}(\bar{V}) \leq 2\tau_0 \cdot \mathrm{DS}(\bar{V})$. The algorithm thus achieves $(2\tau_0, 4\tau_0)$-down neighborhood optimality by Lemma 4.3. □

4.3 Approximate Shifted Inverse

The bottleneck of `ShiftedInverse` is the computation of the $\check{f}(V, j)$'s. In fact, as we will see, even for the counting function with $\ell = 2$, computing $\check{f}(V, j)$ is NP-hard, so implementing `ShiftedInverse` exactly in general takes $|V|^{O(\log D/\varepsilon)}$ time. By digesting the analysis in Section 4.2, we observe that Lemma 4.2 and 4.3 still hold even with some approximate $\check{f}(V, j)$, denoted by $\tilde{f}(V, j)$, as long as the following two conditions are met:

(1) (Smoothness) For any j and any neighboring instances $V \sim V'$ where $V' \preceq V$, $\tilde{f}(V, j+1) \leq \tilde{f}(V', j) \leq \tilde{f}(V, j)$.
(2) (Neighborhood approximability) There exist ρ, c such that for any instance V, there exists an instance $\bar{V} \preceq V$ such that $d(V, \bar{V}) + 1 \leq \rho$ and $f(V) - \tilde{f}(V, 2\tau_0) \leq c \cdot \mathrm{DS}(\bar{V})$.

In particular, smoothness ensures that the sensitivity of $s(\cdot, r)$ is 1 (Lemma 4.1), and neighborhood approximability leads to down neighborhood optimality (Lemma 4.3). We denote the approximate version of our mechanism `ApproxShiftedInverse`, which uses some $\tilde{f}(V, j)$ in place of $\check{f}(V, j)$.

Lemma 4.5. *For any monotonic f and any $\tilde{f}(V, j)$ satisfying the smoothness and neighborhood approximability conditions,* `ApproxShiftedInverse` *preserves ε-DP and is $(\rho, 2c)$-down neighborhood optimal.*

Proof. Follows the same proofs as for Lemma 4.2 and 4.3. □

5 SPECIFIC FUNCTIONS

For an arbitrary monotonic f, `ShiftedInverse` runs in $|V|^{O(\tau)}$ time and is $(O(\tau_0), O(\tau_0))$-down neighborhood optimal. In this section, we consider a number of specific monotonic functions. For each case, we show how to reduce the running time to polynomial and/or further improve the optimality.

5.1 Count/Sum

The sum function is $f(V) = \sum_{t \in S(V)} t$. Note that count is a special case of sum where $t = 1$ for all t. Writing $\psi(x) = \sum_{t \in V(x)} t$, then $f(V) = \sum_x \psi(x)$.

The case $\ell = 1$. For $\ell = 1$, all $x \in \mathcal{X}_V$ are singleton sets, so removing any user u just reduces $f(V)$ by $\psi(\{u\})$. Then $\check{f}(V, j)$ is simply $f(V)$ minus the sum of the j largest $\psi(x)$'s.

Theorem 5.1. *For the sum function with $\ell = 1$,* `ShiftedInverse` *runs in polynomial time and is $(1, 4\tau_0)$-down neighborhood optimal.*

Proof. Running time is obvious. To see that the optimality improves over the generic guarantee in Theorem 4.4, we use Lemma 4.3 directly with $\rho = 1$. Consider the instance $\bar{V} = V$. We have $d(V, \bar{V}) = 0$ and $\mathrm{DS}(\bar{V}) = \max_x \psi(x)$. Moreover, according to the definitions of $\check{f}(V, 2\tau_0)$ and $\mathrm{DS}(\bar{V})$, we have

$$f(V) - \check{f}(V, 2\tau_0) \leq 2\tau_0 \cdot \mathrm{DS}(\bar{V}).$$

Therefore, the algorithm achieves $(1, 4\tau_0)$-down neighborhood optimality. □

The case $\ell \geq 2$. Computing $\check{f}(V, j)$ is NP-hard for the edge counting problem, i.e., even for the special $\ell = 2$ and $\psi(x) = 1$ for all $x \in \mathcal{X}_V$ (so $f(V) = |\mathcal{X}_V| = |V|$), by a reduction from the vertex cover problem: Note that $\check{f}(V, j)$ is $|\mathcal{X}_V|$ minus the largest number of edges that can be covered by j vertices. If we can compute $\check{f}(V, j)$ in polynomial time, then we can find the minimum vertex cover by finding the smallest j such that $\check{f}(V, j) = 0$.

Thus, for $\ell \geq 2$, instead of trying to find $\check{f}(V, j)$ exactly, we solve for an approximate $\tilde{f}(V, j)$ using linear programming. For $j > 2\tau$, set $\tilde{f}(V, j) = 0$. For $j \in [2\tau]$, we assign a weight w_u for each user $u \in \mathcal{U}_V$, and a weight w_x for each subset of users $x \in \mathcal{X}_V$. Then the linear program $\mathrm{LP}(V, j)$ is defined as follows:

$$\begin{aligned}
\text{minimize} \quad & \tilde{f}(V, j) = \sum_{x \in \mathcal{X}_V} (1 - w_x)\psi(x) \\
\text{subject to} \quad & w_x \leq \sum_{u \in x} w_u, \ \forall x \in \mathcal{X}_V, \\
& \sum_{u \in \mathcal{U}_V} w_u \leq j, \\
& 0 \leq w_u \leq 1, \ \forall u \in \mathcal{U}_V, \\
& 0 \leq w_x \leq 1, \ \forall x \in \mathcal{X}_V.
\end{aligned}$$

This LP is similar to those used in LP-based approximation algorithms for vertex cover and set cover problems [16], but for this to work under `ApproxShiftedInverse`, the key is to show it satisfies the smoothness and neighborhood approximability conditions.

Lemma 5.2. *The $\tilde{f}(V, j)$ defined by the LP above is smooth.*

Proof. Let $u^* \in \mathcal{U}$ be the user such that $V \sim_{u^*} V'$. Consider any integer j. On the one hand, given the optimal solution to $\mathrm{LP}(V', j)$, we can construct a feasible solution to $\mathrm{LP}(V, j+1)$ by setting $w_{u^*} = 1$ and $w_x = 1$ for all $x \in \mathcal{X}_V$ such that $u^* \in x$, and copying all other weights. The objective value is no more than $\tilde{f}(V', j)$, thus, $\tilde{f}(V, j+1) \leq \tilde{f}(V', j)$. On the other hand, given the optimal solution to $\mathrm{LP}(V, j)$, we can construct a feasible solution to $\mathrm{LP}(V', j)$ by copying the weights w_u's for the users $u \in \mathcal{U}_{V'}$ and w_x's for the

Algorithm 3: Computing $\check{f}(V, j)$'s for k-selection with $\ell = 1$

Input : The instance V, and the parameters k and τ
Output: $\check{f}_k(V, j)$'s for $j \in [2\tau]$
count$(u) \leftarrow 0$ for all $u \in \mathcal{U}_V$, $\check{f}_k(V, j) \leftarrow 0$ for $j \in [2\tau]$;
$j \leftarrow 0$;
for $i \leftarrow 1, \ldots, |V|$ **do**
  if *sum of all but the largest j counters* count$(u) \le k-1$ **then**
    $\check{f}_k(V, j) \leftarrow t_{(i)}$;
  else
    $j \leftarrow j + 1$;
    if $j > 2\tau$ **then**
      return $\check{f}_k(V, j), j \in [2\tau]$
    else
      $\check{f}_k(V, j) \leftarrow t_{(i)}$;
    end
  end
  Increment count(u) for the user u contributing $t_{(i)}$;
end

subsets of users $x \in \mathcal{X}_{V'}$. The objective value is no more than $\tilde{f}(V, j)$. Therefore, $\tilde{f}(V', j) \le \tilde{f}(V, j)$. □

For neighborhood approximability, we prove a slightly more general result:

Lemma 5.3. *For any instance V and any integer $j \in [2\tau_0]$, we have $f(V) - \tilde{f}(V, j) \le j \cdot \mathrm{DS}(V)$.*

Proof. Given a feasible solution to LP(V, j), we have

$$f(V) - \tilde{f}(V, j) = \sum_{x \in \mathcal{X}_V} w_x \cdot \psi(x) \le \sum_{x \in \mathcal{X}_V} \sum_{u \in x} w_u \cdot \psi(x)$$
$$= \sum_{u \in \mathcal{U}_V} w_u \cdot \sum_{x \ni u} \psi(x) \le j \cdot \mathrm{DS}(V).$$

□

The running time of solving a linear program is polynomial in the number of variables, which is $O(\ell \cdot |V|)$ in our case, so the algorithm runs in polynomial time. Then down neighborhood optimality follows by setting $j = 2\tau_0$ in Lemma 5.3:

Theorem 5.4. *For the sum function with any $\ell \ge 2$,* `Approx-ShiftedInverse` *runs in polynomial time and is $(1, 4\tau_0)$-down neighborhood optimal.*

5.2 k-Selection

Sort $S(V)$ in a descending order as $t_{(1)}, \ldots, t_{(|V|)}$, then the k-selection problem is to return $f_k(V) = t_{(k)}$. Define $t_{(k)} := 0$ for $k > |V|$. Let $S_i(V) = \{t_{(1)}, \ldots, t_{(i)}\}$. As we go from $i = 1$ to $i = |V|$, $\check{f}_k(V, j)$ corresponds to the last i such that all but $k-1$ tuples in $S_i(V)$ are contributed by at most j users. Flipping the question around, it boils down to the following decision problem: Can we cover $S_i(V)$ with j users while leaving at most $k-1$ tuples uncovered?

The case $\ell = 1$. For $\ell = 1$, the contributions of the users are disjoint. Then above decision problem can be answered easily by simply picking the j users with the largest contributions in $S_i(V)$. In fact, we can compute all $\check{f}_k(V, j)$, $j = 0, 1, \ldots, 2\tau$ incrementally while making a single pass over $S(V)$, as shown in Algorithm 3. The algorithm maintains a counter for each user $u \in \mathcal{U}_V$ for the number of tuples in $S_i(V)$ contributed by u. For each i, we check if the sum of all but the largest j counters is at most $k-1$. If yes, namely, the largest $i-1$ tuples can be covered by at most j users, then $\check{f}_k(V, j)$ is at most $t_{(i)}$. Otherwise, we have found $\check{f}_k(V, j)$ and continue onto $\check{f}_k(V, j+1)$. By maintaining all the counters in a binary search tree, each iteration of Algorithm 3 can be implemented in $O(\log |V|)$ time, so the total running time is $O(|V| \log |V|)$.

Theorem 5.5. *For k-selection with $\ell = 1$,* `ShiftedInverse` *runs in polynomial time and is $(2\tau_0, 4\tau_0)$-down neighborhood optimal. For $k = 1$ (i.e., finding the maximum), it is $(2\tau_0 + 1, 2)$-down neighborhood optimal.*

Proof. The first claim follows Theorem 4.4. For the second claim, consider the instance $\bar{V} \le V$ constructed as follows. Given the instance $\check{V}_{2\tau_0}$, $\bar{V}$ is constructed from $\check{V}_{2\tau_0}$ by adding back the largest element in $S(V)$ to the original multiset. We have $d(V, \bar{V}) \le d(V, \check{V}_{2\tau_0}) \le 2\tau_0$ and

$$f_1(V) - \check{f}_1(V, 2\tau_0) \le \mathrm{DS}(\bar{V}).$$

Therefore, the algorithm achieves $(2\tau_0 + 1, 2)$-down neighborhood optimality. □

Remark. The k-selection problem has been studied in the tuple-DP model, which is the special case of user-DP with $\ell = 1$ and $|V(x)| \in \{0, 1\}$. A number of techniques, including inverse sensitivity [3, 23], smooth sensitivity [17], and binary search [13], can achieve an optimal *rank error* (which is the difference between k and the rank of the returned element) of $O(\frac{1}{\varepsilon} \log D)$. Note that for the k-selection problem, ρ under tuple-DP is exactly the rank error. So our result can be considered as a generalization of these results to user-DP. A worst-case rank error guarantee is impossible to achieve under user-DP, since removing one user's data may affect the rank of an element arbitrarily. Down neighborhood optimality is thus a more appropriate, instance-specific measure of optimality.

The case $\ell \ge 2$. For $\ell \ge 2$, the decision problem above is at least as difficult as vertex cover. Instead of trying to solve it exactly, we use the following LP relaxation, where we assign a weight w_u to each user u and w_t to each tuple t:

$$\begin{aligned}
\text{minimize} \quad & \mathrm{VC}_i(V) = \sum_{u \in \mathcal{U}_V} w_u \\
\text{subject to} \quad & \sum_{u \in V^{-1}(t)} w_u \ge w_t, \ \forall t \in S(V), \\
& \sum_{t \in S_i(V)} w_t \ge i - k + 1 \\
& 0 \le w_u \le 1, \ \forall u \in \mathcal{U}_V, \\
& 0 \le w_t \le 1, \ \forall t \in S(V).
\end{aligned}$$

Then we compute

$$\tilde{f}_k(V, j) = \min\{t_{(i+1)} \mid \mathrm{VC}_i(V) \le j\},$$

for $j \in [2\tau]$, and set $\tilde{f}_k(V, j) = 0$ for $j > 2\tau$. The total running time is clearly polynomial; to make it more efficient, we can do a binary search over i to find each $\tilde{f}_k(V, j)$.

The smoothness proof is similar to the LP for sum/count.

Lemma 5.6. *For any k, j, and any neighboring instance $V \sim V'$ where $V' \preceq V$, $\tilde{f}_k(V, j+1) \le \tilde{f}_k(V', j) \le \tilde{f}_k(V, j)$.*

Proof. Let $u^* \in \mathcal{U}$ be the user such that $V \sim_{u^*} V'$. The lemma is trivial for $j > 2\tau$. Consider any integer $j \in [2\tau]$. On the one hand, given we can cover $i-k+1$ elements in $S_i(V)$ with no more than j users where $\tilde{f}_k(V, j) = t_{(i+1)}$, we can cover a subset of these elements with no more than j users, therefore, $\tilde{f}_k(V', j) \le \tilde{f}_k(V, j)$. On the other hand, given we can cover $i-k+1$ elements in $S_i(V')$ with no more than j users where $\tilde{f}_k(V', j) = t'_{(i+1)}$. By setting $w_{u^*} = 1$, we can ensure all tuples t contributed by u are covered, and $\tilde{f}_k(V, j+1) \le \tilde{f}_k(V', j)$. □

The proof of neighborhood approximability is more technical, since now $\tilde{f}_k(V, j)$ is not the direct solution of the LP. It turns out that we lose a factor of ℓ compared with `ShiftedInverse`, which is the price for polynomial time.

Lemma 5.7. *For any instance V, we have $f_k(V) - \tilde{f}_k(V, 2\tau_0) \le d(V, \bar{V}) \cdot \mathrm{DS}(\bar{V})$ for some instance $\bar{V} \preceq V, d(V, \bar{V}) \le 2\ell\tau_0$. When $k = 1$, we have $f_1(V) - \tilde{f}_1(V, 2\tau_0) \le \mathrm{DS}(\bar{V})$ for some instance $\bar{V} \preceq V, d(V, \bar{V}) \le 2\ell\tau_0$.*

Proof. We first prove $\tilde{f}_k(V, 2\tau_0) \ge \check{f}_k(V, 2\ell\tau_0 + 1)$, so that the first claim can be derived using a similar proof as Theorem 4.4.

For any $\mathrm{LP}_i(V)$, let $\{w_t^*\}_t$ and $\{w_u^*\}_u$ be the optimal solutions. To prove the target claim, it suffices to show that for any i, we can find $\ell \cdot \sum w_u^* + 1$ users to cover at least $\sum w_t^* \ge i - k + 1$ tuples in $S_i(V)$. We separate the users into two subsets according to whether its weight is heavy ($\ge \frac{1}{\ell}$) or not, i.e., let

$$U_1(V) = \{u \in \mathcal{U}_V : w_u^* \ge \frac{1}{\ell}\},$$

$$U_2(V) = \mathcal{U}_V - U_1(V).$$

Similarly, we separate the elements according to whether it is contributed by some heavy user or not:

$$S_i^1(V) = \{t \in S_i(V) : \exists u \in U_1(V), u \in V^{-1}(t)\},$$

$$S_i^2(V) = S_i(V) - S_i^1(V).$$

By definition, it is trivial to see that

$$|U_1(V)| \le \ell \cdot \sum_{u \in U_1(V)} w_u^*, \tag{4}$$

$$w_t^* < 1, \forall t \in S_i^2(V) \tag{5}$$

Therefore, we first select all the users in $U_1(V)$ so that all the elements in S_i^1 are well covered. By (4), the remaining problem is to select a subset of users $U^* \subseteq U_2(V)$ satisfying the following two properties:

(1) $|U^*| \le \ell \cdot \sum_{u \in U_2(V)} w_u^* + 1$.
(2) $|\bigcup_{u \in U^*} S(V, u)| \ge \sum_{t \in S_i^2(V)} w_t^*$.

We specially define

$$\mathrm{Count}(u, S_i^2(V)) = |S(V, u) \bigcap S_i^2(V)|,$$

representing the number of elements in $S_i^2(V)$ contributed by u, and select the first $\lceil \ell \cdot \sum_{u \in U_2(V)} w_u^* \rceil$ users with maximum $\mathrm{Count}(u, S_i^2(V))$ as U^*. The first property is satisfied naturally. For the second one, by (5), we have

$$\sum_{t \in S_i^2(V)} w_t^* = \sum_{u \in U_2(V)} \left(w_u^* \cdot \mathrm{Count}(u, S_i^2(V)) \right), \tag{6}$$

so that

$$\left| \bigcup_{u \in U^*} S(V, u) \right| \ge \sum_{u \in U^*} \left(\frac{1}{\ell} \cdot \mathrm{Count}(u, S_i^2(V)) \right) \tag{7}$$

$$\ge \sum_{u \in U_2(V)} \left(w_u^* \cdot \mathrm{Count}(u, S_i^2(V)) \right). \tag{8}$$

The first inequality is because each element is contributed by at most ℓ users. The second one is because $|U^*| \ge \ell \cdot \sum_{u \in U_2(v)} w_u^*$ and each w_u^* is at most $\frac{1}{\ell}$. Combine (6) and (8), we show the second property is satisfied.

For the second sentence, when $k = 1$, given the instance $\check{V}(2\ell\tau_0)$, we can construct the instance $\bar{V} \preceq V$ by adding back the largest element in $S(V)$, so that

$$f_1(V) - \tilde{f}_1(V, 2\tau_0) \le f_1(V) - \check{f}_1(V, 2\ell\tau_0) \le \mathrm{DS}(\bar{V}),$$

and $d(V, \bar{V}) \le d(V, \check{V}(2\ell\tau_0)) \le 2\ell\tau_0$. □

Theorem 5.8. *For k-selection with $\ell \ge 2$, `ApproxShiftedInverse` runs in polynomial time and is $(2\ell\tau_0+1, 4\ell\tau_0+2)$-down neighborhood optimal. For $k = 1$, it is $(2\ell\tau_0 + 1, 2)$-down neighborhood optimal.*

5.3 Frequency Moments

In this and the next two subsections, we consider various frequency moments. For these problems, the ordering of the tuples in $S(V)$ is not important, i.e., they are categorical data. For each i, let $\psi_i(x)$ be the frequency (multiplicity) of i in $V(x)$ and $\psi_i(V) = \sum_{x \in \mathcal{X}_V} \psi_i(x)$ be the frequency of i in $S(V)$. For an integer k, the k-th frequency moment is

$$F_k(V) = \sum_{i \in S(V)} \psi_i(V)^k.$$

Note that $F_1(V) = |V|$, which has been discussed in Section 5.1. Below, we show how to reduce $F_k(V)$ to a sum problem for any $2 \le k < \infty$. Consider $F_2(V)$ first. We expand each $\psi_i(V)^2$ as

$$\psi_i(V)^2 = \left(\sum_{x \in \mathcal{X}_V} \psi_i(x) \right)^2 = \sum_{x \in \mathcal{X}_V} \psi_i(x)^2 + \sum_{x_1, x_2 \in \mathcal{X}_V, x_1 \ne x_2} 2\psi_i(x_1)\psi_i(x_2).$$

Then we construct an instance V_{sum} for the sum problem as follows. For every $x \in \mathcal{X}_V$ and each $i \in V(x)$, we put $\psi_i(x)^2$ into $V_{\mathrm{sum}}(x)$; for every $x_1, x_2 \in \mathcal{X}_V, x_1 \ne x_2$, and each $i \in V(x_1) \cap V(x_2)$, we put $2\psi_i(x_1) \cdot \psi_i(x_2)$ into $V_{\mathrm{sum}}(x_1 \cup x_2)$. It can be verified that (1) $F_2(V) = \sum_x V_{\mathrm{sum}}(x)$, and (2) $V \sim V'$ iff $V_{\mathrm{sum}} \sim V'_{\mathrm{sum}}$. Therefore, both privacy and optimality carry over from the sum problem to

computing $F_2(V)$. Note that, however, ℓ doubles after this reduction, but ρ and c are independent of ℓ anyway.

This technique easily generalizes to any constant $k \geq 2$. V_{sum} will be defined over all the unions of up to k subsets of users $x \in \mathcal{X}_V$. Its size is still polynomial in $|V|$ for constant k.

5.4 Distinct Count

Distinct count is a special frequency moment $F_0(V) = |\text{Supp}(S(V))|$. It is monotonic so `ShiftedInverse` is $(2\tau_0, 4\tau_0)$-down neighborhood optimal, but it runs in super-polynomial time. Below we describe an instantiation of `ApproxShiftedInverse` for this problem.

For the distinct count problem, $\check{f}(V, j)$ is the smallest number of distinct values in $S(V)$ after removing j users from V. We will design a linear program to compute an approximate $\tilde{f}(V, j)$. We could ignore the distinct requirement and use the same LP as that for the count problem, but this leads to a badly underestimated $\tilde{f}(V, j)$. Observing that the distinct count only decreases when all the copies of a distinct value have been removed, we introduce a weight w_i for each distinct value $i \in S(V)$, representing whether any copy of i exists. The LP is defined as follows:

$$\begin{aligned}
\text{minimize} \quad & \tilde{f}(V, j) = \sum_{i \in [k]} 1 - w_i \\
\text{subject to} \quad & w_i \leq w_t, \ \forall t = i, \\
& w_t \leq \sum_{u \in x} w_u, \ \forall t \in V(x), x \in \mathcal{X}_V, \\
& \sum_{u \in \mathcal{U}_V} w_u \leq j, \\
& 0 \leq w_u \leq 1, \ \forall u \in \mathcal{U}_V, \\
& 0 \leq w_t \leq 1, \ \forall t \in V(x), x \in \mathcal{X}_V, \\
& 0 \leq w_i \leq 1, \ \forall i \in S(V).
\end{aligned}$$

We can still show that this LP is smooth, which ensures the privacy of `ApproxShiftedInverse`.

Lemma 5.9. *For any j and any neighboring instance $V \sim V'$ where $V' \leq V$, $\tilde{f}(V, j+1) \leq \tilde{f}(V', j) \leq \tilde{f}(V, j)$.*

Proof. Similar to that of Lemma 5.2. □

Unfortunately, we do not have a neighborhood optimality result for this instantiation of `ApproxShiftedInverse` for the distinct count problem. Nevertheless, it performs very well in the experiments (see Section 6).

5.5 Maximum Frequency

The maximum frequency is also a special frequency moment $F_\infty(V) = \max_i \psi_i(V)$. It is a monotonic function so `ShiftedInverse` can achieve $(2\tau_0, 4\tau_0)$-down neighborhood optimal, but it runs in super-polynomial time. To reduce the running time, we use the following linear program to compute $\tilde{f}(V, j)$ and apply `ApproxShiftedInverse`:

$$\begin{aligned}
\text{minimize} \quad & \tilde{f}(V, j) = y \\
\text{subject to} \quad & \sum_{x \in \mathcal{X}_V} (1 - w_x)\psi_i(x) \leq y, \ \forall i \in S(V), \\
& w_x \leq \sum_{u \in x} w_u, \ \forall x \in \mathcal{X}_V, \\
& \sum_{u \in \mathcal{U}_V} w_u \leq j, \\
& 0 \leq w_u \leq 1, \ \forall u \in \mathcal{U}_V, \\
& 0 \leq w_x \leq 1, \ \forall x \in \mathcal{X}_V.
\end{aligned}$$

We can show that $\tilde{f}(V, j)$ is smooth, so `ApproxShiftedInverse` satisfies ε-DP:

Lemma 5.10. *For any j and any neighboring instance $V \sim V'$, $V' \leq V$, $\tilde{f}(V, j+1) \leq \tilde{f}(V', j) \leq \tilde{f}(V, j)$.*

Proof. Similar to that of Lemma 5.2. □

We do not have a neighborhood optimality result for this instantiation of `ApproxShiftedInverse` for the maximum frequency problem.

The (ρ, c)-down neighborhood optimality for various functions is summarized in Table 1.

6 EXPERIMENTS

We conducted experiments on the TPC-H benchmark and real-world network data from SNAP [14]. We tested `(Approx)ShiftedInverse` on all the functions covered in Section 5. For comparison, we also tested ZetaSQL [27], R2T [10], and the Recursive Mechanism (RM) [7]. These mechanisms do not support all the functions and all ℓ's, which will be pointed out as we present the experimental results for each function.

6.1 Datasets and Functions

The TPC-H benchmark consists on eight relations: `Region(RK)`, `Nation(RK,NK)`, `Customer(NK,CK)`, `Orders(CK,OK)`, `Supplier(NK,SK)`, `Part(PK)`, `PartSupp(SK,PK)`, `Lineitem(SK,PK,OK,LN)`. We used datasets with scale factors ranging from 1/8 to 4. The one with scale factor 1 has 1×10^4 suppliers, 1.5×10^5 customers, and around 6×10^6 lineitems. We also used 4 real-world graph datasets: **Amazon**, **Bitcoin**, **Gnutella**, and **RoadnetTX**. **Amazon** is an Amazon product co-purchasing network. **Bitcoin** is a who-trusts-whom network of people trading with Bitcoin, where each edge is associated with a rating score in the range $[-10, 10]$. **Gnutella** is a peer-to-peer file sharing network, and **RoadnetTX** is a road network of Texas. The basic information of the networks are shown in Table 2. For these graph data, the user-DP model degenerates into node-DP.

Sum, count, and F_2. For graph data, we used edge counting q_{1-} and triangle counting $q_\triangle$, which have $\ell = 2$ and 3, respectively. ZetaSQL and R2T require an upper bound R on the count that each user can contribute, while we require D, an upper bound on the total count. To set these parameters appropriately, we choose a degree upper bound D_{deg} for each graph as shown in Table 2. Then, for edge counting, we set $R_{1-} = D_{\text{deg}}$ and $D_{1-} = \frac{|\mathcal{U}_V| \cdot D_{\text{deg}}}{2}$; for triangle counting, $R_\triangle = \frac{D_{\text{deg}}^2}{2}$ and $D_\triangle = \frac{|\mathcal{U}_V| \cdot D_{\text{deg}}^2}{6}$, where $|\mathcal{U}_V|$ is the number of nodes in the graph. Note we set D for the worst-case scenario where each node has the maximum number of edges/triangles. If some prior knowledge about the degree distribution is known

Table 1: Summary of (ρ, c)-down neighborhood optimality. The ones with $*$ can only be achieved in super-poly time.

Function f	Sum/count	Maximum	k-selection	$F_k, 1 \le k < \infty$	F_0	F_∞
$\ell = 1$	$(O(1), O(\tau))$	$(O(\tau), O(1))$	$(O(\tau), O(\tau))$	$(O(1), O(\tau))$	$(O(\tau), O(\tau))^*$	$(O(\tau), O(\tau))^*$
$\ell \ge 2$	$(O(1), O(\tau))$	$(O(\ell\tau), O(1))$	$(O(\ell\tau), O(\ell\tau))$	$(O(1), O(\tau))$	$(O(\tau), O(\tau))^*$	$(O(\tau), O(\tau))^*$

Table 2: Basic information of graph data.

Dataset	**Amazon**	**Bitcoin**	**Gnutella**	**RoadnetTX**
Nodes	262,000	5,880	62,600	1,380,000
Edges	900,000	35,600	148,000	1,922,000
Maximum degree	420	795	95	12
Degree bound D_{deg}	1,024	2,048	256	32

(e.g., Zipfian), D can be made smaller and our mechanisms would work better.

For TPC-H data, we chose the following five queries:

- Q12 counts the number of lineitems. For this query, we only consider the customers as users, hence $\ell = 1$. Then this query can be instantiated in our model by setting $V(\{u\})$ as the set of lineitems belonging to u, with $f(V) = |V|$.
- Q18 returns the total quantity of lineitems purchased by all customers. This is similar to Q12, but $V(\{u\})$ is a multiset of quantities, and $f(V) = \sum_{t \in S(V)} t$.
- Q5 counts the number of lineitems where the customer and supplier are from the same nation. For this query, both customers and suppliers are considered users, who jointly contribute the lineitems, hence $\ell = 2$. For each customer u and supplier v, if they are from the same nation, then $V(\{u, v\})$ is the multiset of lineitems purchased by u and supplied by v, otherwise $\emptyset$. The function is count.
- Q7 returns the total revenue from lineitems purchased by a customer in nation A and supplied by a supplier from nation B. We also have $\ell = 2$ for this query. For each customer u from nation A and each supplier from nation B, $V(\{u, v\})$ is the multiset of revenues of lineitems purchased by u and supplied by v. The function is sum.
- We adapted Q11 to compute its second moment F_2. For this query, $V(\{v\})$ is the brands of parts satisfying a certain condition supplied by v, and we compute $F_2(S(V))$, which represents the skewness of the distribution of the brands. Note that although $\ell = 1$ for the frequency moment problem, our reduction turns it into a sum problem with $\ell = 2$.

We set R to 10^4 for counting queries and 10^6 for sum queries, and set $D = R \cdot |\mathcal{U}_V|$; for F_2, we set $R = 10^8$ and $D = R \cdot |\mathcal{U}_V|$.

k-selection. Using the same definition of V from Q18 and Q7 above, we queried for the minimum, median[2], 75%-percentile, and maximum in $S(V)$. In addition, we used Q9, which is interested in the profit of each part of each supplier, i.e., $V(\{v\})$ is the multiset of profits of the parts supplied by v. Note that the profits can take negative values, but they can easily be taken care of by setting the output domain to $\mathcal{R} = [-D, D]$. We set $D = 10^5$ for these queries. We also ran these k-selection queries on the rating scores in the **Bitcoin** dataset. This query has $\ell = 2$ and features a small D, as all ratings are between -10 and 10.

Distinct count and maximum frequency. We computed the distinct count and maximum frequency on various attributes with various predicates on the TPC-H data. The users are defined as the suppliers or the customers ($\ell = 1$), or both ($\ell = 2$). For distinct count, the upper bound D is set based on the TPC-H specification. For example, a date attribute takes values within a 7-year period, so $D = 7 \times 365$. For maximum frequency, D is simply set to be the number of tuples, e.g., 6×10^6 for lineitems.

Experimental environment. We conducted all the experiments on a machine with a 2.2GHz Intel Xeon CPU and 256GB memory. We repeated each experiment 100 times. For accuracy, we removed the best 20 and the worst 20 runs and reported the average error of the remaining runs. For efficiency, the time limit is set to 6 hours for each run and we reported the average running time of all runs. The default privacy budget is $\varepsilon = 1$ and the failure probability is $\beta = 0.1$.

6.2 Experimental Results

6.2.1 Sum, count, and F_2. In Table 3, we reported the errors and running times for sum, count, and F_2. Recall that they are all linear functions (F_2 can be reduced to a linear function), which are a special case of monotonic functions. On these queries, all mechanisms yield fairly accurate results, except for triangle counting on the **Gnutella** graph and the F_2 query. On these two queries, RM gives good results. However, a serious issue with RM is its computational cost. It can only complete about half of the test cases within the 6-hour time limit. We can also see that ZetaSQL performs best in the 2 cases where $\ell = 1$. However, it can does not support $\ell \ge 2$.

6.2.2 Quantile queries. For quantile queries, the only prior user-DP mechanism is ZetaSQL, and it only supports the $\ell = 1$ case. The errors (both absolute errors and rank errors) and running times of `(Approx)ShiftedInverse` and ZetaSQL are shown in Table 4. The results indicate that `(Approx)ShiftedInverse` outperforms ZetaSQL quite significantly in almost all test cases for $\ell = 1$, due to the down neighborhood optimality of our mechanism. Moreover, when the answers are stable, i.e., changing a few users does not change the quantile, our framework can even achieve zero error, as shown in Q18 and **Bitcoin**. We also see that the running time for maximum and minimum selection is short because of the small size of the linear programs.

6.2.3 Distinct count. For distinct count, the only prior user-DP mechanism is R2T. We reported the errors and running times in Table 5. We can see that `(Approx)ShiftedInverse` outperforms R2T in terms of accuracy by quite some margin, although it spends more time in solving the LPs.

6.2.4 Maximum frequency. For maximum frequency, the only prior user-DP mechanism is ZetaSQL but it only supports the $\ell = 1$

[2]Technically, the median function is not monotonic, but we can first find a privatized $|V|$, and then solve the $\frac{|V|}{2}$-selection problem, each using a privacy budget of $\varepsilon/2$.

Table 3: Comparison of (Approx)ShiftedInverse, R2T, RM and ZetaSQL on sum, count, and F_2.

Dataset		**Amazon**		**Gnutella**		**RoadnetTX**		TPC-H				
Function type		Count		Count		Count		Count		Sum		F_2
Query		Edge	Triangle	Edge	Triangle	Edge	Triangle	Q12	Q5	Q18	Q7	Q11
Query result	Value	900,000	718,000	148,000	2,020	1,920,000	82,900	6,000,000	240,000	153,000,000	66,500,000	39,600,000
	Time(s)	0.589	6.04	0.0902	0.410	1.30	9.38	1.49	2.10	1.86	1.63	0.757
(Approx)ShiftedInverse	Relative error(%)	0.884	1.55	1.08	13.7	0.0138	0.216	0.00538	0.599	0.113	0.514	15.7
	Time(s)	262	1,360	41.4	1.81	549	29.5	20.3	51.4	61.2	487	142
R2T	Relative error(%)	0.448	0.993	1.12	10.2	0.0112	0.0972	0.00923	1.270	0.105	1.51	40.8
	Time(s)	20.4	26.0	3.05	1.08	37.2	11.6	22.3	8.00	64.5	58.1	3.61
RM	Relative error(%)	Over time limit			1.49	Over time limit	0.0321	0.000504	0.0726	0.0206	Over time limit	1.41
	Time(s)				4.49		302	23.3	3,140	61.1		118
ZetaSQL	Relative error(%)	Not supported						0.000286	Not supported	0.00271	Not supported	
	Time(s)							19.4		59.9		

Table 4: Comparison of (Approx)ShiftedInverse and ZetaSQL on quantile queries on TPC-H benchmark.

Query		Q9				Q18				Q7				Rating Score			
α-Quantile(%)		100	75	50	0	100	75	50	0	100	75	50	0	100	75	50	0
Query result	Value	104	34.7	19.9	-8.04	50.0	38.0	26.0	1.00	104	52.4	34.9	0.814	10.0	2.0	1.0	-10.0
	Time(s)	3.23	12.7	12.5	3.23	1.86	8.29	8.09	1.85	1.62	5.31	5.28	1.62	0.0134	0.0159	0.0158	0.0100
(Approx)ShiftedInverse	Error	5.15	0.0434	0.0171	1.94	0.00	0.00	0.00	0.00	2.23	0.167	0.243	0.0364	0.00	0.00	0.00	0.00
	Rank error	40.0	3,490	2,030	45.0	0.00	0.00	0.00	0.00	28.0	3,850	6,450	34.0	0.00	0.00	0.00	0.00
	Time(s)	85.2	162	209	85.1	74.3	188	271	74.4	38.9	2,970	5,240	38.9	1.46	56.4	232	1.61
ZetaSQL	Error	15.1	15.0	6.58	10.3	1.97	0.0422	0.233	2.01	Not supported							
	Rank error	Not applicable	888,000	719,000	382,000	239,000	60,200	61,800	360,000								
	Time(s)	152	152	151	154	127	126	128	1286								

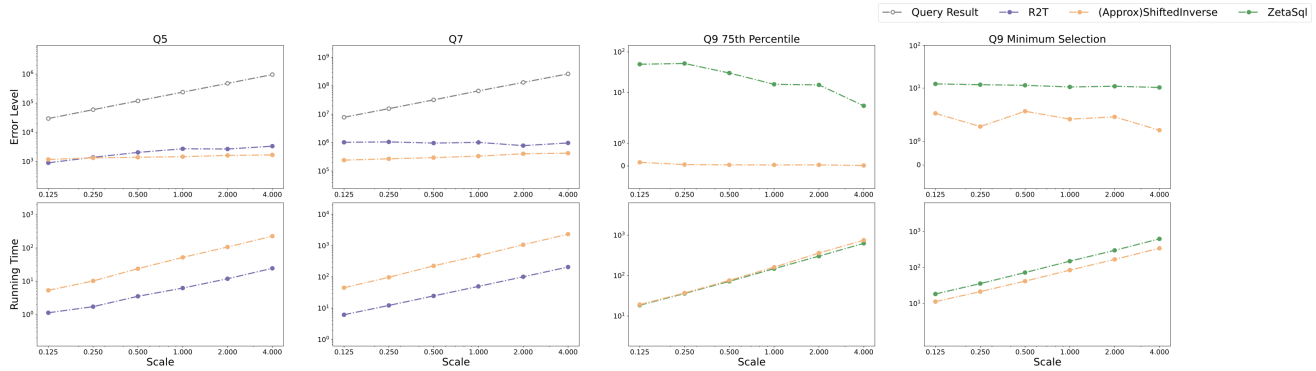

Figure 2: Error and running time of (Approx)ShiftedInverse, R2T and ZetaSQL given various scales.

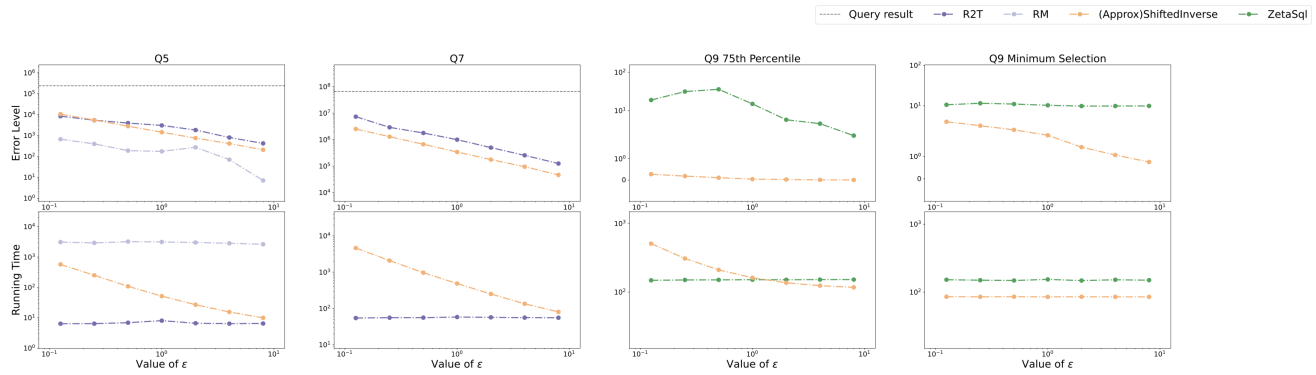

Figure 3: Error and running time of (Approx)ShiftedInverse, R2T, RM and ZetaSQL given various ε.

Table 5: Comparison of (Approx)ShiftedInverse and R2T on distinct count.

User		Supplier		Customer		Supplier and Customer
Queried attribute		PS.AQ	L.EP	O.OD	L.RD	Q7
Query result	Value	9,570	194,000	2,410	2,540	1,610,000
	Time(s)	0.297	4.43	0.600	3.64	5.42
(Approx)ShiftedInverse	Relative error(%)	5.54	5.85	0.527	0.611	0.0672
	Time(s)	23.1	51.7	114	288	4,590
R2T	Relative error(%)	6.58	17.9	0.613	15.0	0.313
	Time(s)	3.95	6.20	1.52	4.74	68.1

Table 6: Comparison of (Approx)ShiftedInverse and ZetaSQL on maximum frequency.

User		Supplier	Customer	Supplier and Customer		
Queried attribute		L.T	O.OP	L.Q	L.D	L.SD
Query result	Value	667,000	301,000	37,000	167,000	2,670
	Time(s)	2.02	0.294	1.39	1.38	1.32
(Approx)ShiftedInverse	Relative error(%)	0.098	0.00620	0.629	0.157	1.60
	Time(s)	59.9	14.5	2,590	5,550	989
ZetaSQL	Relative error(%)	0.0471	0.0106	Not supported		
	Time(s)	57.4	6.88			

case. Table 6 shows the results for both mechanisms. For $\ell = 1$, (Approx)ShiftedInverse and ZetaSQL have similar accuracy and running times, but the former also extends to $\ell \geq 2$.

6.2.5 Scalability. Next, we examined the effects of the scale of the dataset using the TPC-H benchmark. The scale factor ranges from 2^{-3} to 2^2, and the queries include Q5 and Q7 for sum estimation, and Q9 for quantile queries. The results are shown in Figure 2. We can see that the scale has a small impact on the errors, except maybe for ZetaSQL on the 75-th percentile of Q9. This is because in the TPC-H benchmark, the number of tuples that each user contributes to does not change much as the scale increases. Thus, when each leaf in the quantile tree contains more tuples, it is easier for ZetaSQL to find the correct leaf and return an output close to the actual answer. All the mechanisms have the running time polynomial to the scale.

6.2.6 Privacy budget ε. We also examined the effects of the privacy budget ε using the TPC-H benchmark. The privacy budget ε ranges from 2^{-3} to 2^3 and the results are shown in Figure 3. As expected, the error decreases as the privacy budget increases. Again, we can see that the accuracy of our framework and R2T is similar, and (Approx)ShiftedInverse performs much better than ZetaSQL on quantile queries. In terms of running time, R2T, RM and ZetaSQL are not affected by the value of privacy budget, while the running time of (Approx)ShiftedInverse decreases as ε increases. This is because τ is dependent on ε, and as ε increases, the number of linear programs that we have to solve also decreases.

7 FUTURE WORK

An obvious direction for future work is how to achieve neighborhood optimality for distinct count and maximum frequency within polynomial time. Another interesting open question is non-monotonic functions, such as mean or median. Mean and median can be both computed by composing two monotonic functions, but whether this composition preserves neighborhood optimality is unclear, although it works well in practice.

ACKNOWLEDGEMENTS

This work has been supported by HKRGC under grants 16201819, 16205420, and 16205422. We appreciate the anonymous reviewers for their helpful comments on the manuscript.

REFERENCES

[1] Noga Alon, Yossi Matias, and Mario Szegedy. 1999. The Space Complexity of Approximating the Frequency Moments. *J. Comput. System Sci.* 58 (1999), 137–147.
[2] Kareem Amin, Alex Kulesza, Andres Muñoz Medina, and Sergei Vassilvitskii. 2019. Bounding user contributions: A bias-variance trade-off in differential privacy. In *Proceedings of the 36th International Conference on Machine Learning*, Vol. 97. PMLR, 263–271.
[3] Hilal Asi and John C. Duchi. 2020. Instance-optimality in differential privacy via approximate inverse sensitivity mechanisms. In *Advances in Neural Information Processing Systems*, Vol. 33. Curran Associates, Inc., 14106–14117.
[4] Hilal Asi and John C. Duchi. 2020. Near instance-optimality in differential privacy. *ArXiv e-prints* (2020).
[5] Ghazi Badih, Kumar Ravi, and Manurangsi Pasin. 2021. User-level private learning via correlated sampling. In *Advances in Neural Information Processing Systems*.
[6] Jeremiah Blocki, Avrim Blum, Anupam Datta, and Or Sheffet. 2013. Differentially private data analysis of social networks via restricted sensitivity. In *Proceedings of the 4th Conference on Innovations in Theoretical Computer Science*. Association for Computing Machinery, New York, NY, USA, 87–96.
[7] Shixi Chen and Shuigeng Zhou. 2013. Recursive mechanism: towards node differential privacy and unrestricted joins. In *Proceedings of the 2013 ACM SIGMOD International Conference on Management of Data*. Association for Computing Machinery, New York, NY, USA, 653–664.
[8] Dwork Cynthia and Lei Jing. 2009. Differential privacy and robust statistics. In *Proceedings of the Forty-First Annual ACM Symposium on Theory of Computing*. Association for Computing Machinery, New York, NY, USA, 371–380.
[9] Wei Yen Day, Ninghui Li, and Min Lyu. 2016. Publishing graph degree distribution with node differential privacy. In *Proceedings of the 2016 International Conference on Management of Data*. Association for Computing Machinery, New York, NY, USA, 123–138.
[10] Wei Dong, Juanru Fang, Ke Yi, Yuchao Tao, and Ashwin Machanavajjhala. 2022. R2T: Instance-optimal truncation for differentially private query evaluation with foreign keys. In *Proc. ACM SIGMOD International Conference on Management of Data*.
[11] Cynthia Dwork and Aaron Roth. 2014. The algorithmic foundations of differential privacy. *Foundations and Trends in Theoretical Computer Science* 9 (2014).
[12] Alessandro Epasto, Mohammad Mahdian, Jieming Mao, Vahab Mirrokni, and Lijie Ren. 2020. Smoothly bounding user contributions in differential privacy. In *Advances in Neural Information Processing Systems*.
[13] Ziyue Huang, Yuting Liang, and Ke Yi. 2021. Instance-optimal mean estimation under differential privacy. In *Advances in Neural Information Processing Systems*.
[14] Leskovec Jure and Krevl Andrej. 2016. SNAP datasets: Stanford large network dataset collection (2014). *URL http://snap. stanford. edu/data* (2016), 49.
[15] Shiva Prasad Kasiviswanathan, Kobbi Nissim, Sofya Raskhodnikova, and Adam Smith. 2013. Analyzing graphs with node differential privacy. *Lecture Notes in Computer Science* 7785 LNCS.
[16] Jon Kleinberg and Eva Tardos. 2005. *Algorithm design*. Addison-Wesley Longman Publishing Co., Inc., USA.
[17] Nissim Kobbi, Raskhodnikova Sofya, and Smith Adam. 2007. Smooth sensitivity and sampling in private data analysis. In *Proceedings of the Thirty-Ninth Annual ACM Symposium on Theory of Computing*. Association for Computing Machinery, New York, NY, USA, 75–84.
[18] Ios Kotsogiannis, Yuchao Tao, Xi He, Maryam Fanaeepour, Ashwin Machanavajjhala, Michael Hay, and Gerome Miklau. 2018. PrivateSQL: a differentially private SQL query engine. In *Proceedings of the VLDB Endowment*.
[19] Daniel Levy, Ziteng Sun, Kareem Amin, Satyen Kale, Alex Kulesza, Mehryar Mohri, and Ananda Suresh. 2021. Learning with user-level privacy. In *Advances in Neural Information Processing Systems*.
[20] Yuhan Liu, Ananda Theertha Suresh, Felix Yu, Sanjiv Kumar, and Michael Riley. 2020. Learning discrete distributions: user vs item-level privacy. In *Advances in Neural Information Processing Systems*.
[21] Frank McSherry and Kunal Talwar. 2008. Mechanism design via differential privacy. In *48th Annual IEEE Symposium on Foundations of Computer Science*. 94–103.
[22] Johnson Noah, Near Joseph P., and Song Dawn. 2018. Towards practical differential privacy for SQL queries. (2018), 526–539.
[23] Adam Smith. 2011. Privacy-preserving statistical estimation with optimal convergence rates. In *Proceedings of the Forty-Third Annual ACM Symposium on Theory of Computing*. Association for Computing Machinery, New York, NY, USA, 813–822.

[24] Raskhodnikova Sofya and Smith Adam. 2016. Lipschitz extensions for node-private graph statistics and the generalized exponential mechanism. In *2016 IEEE 57th Annual Symposium on Foundations of Computer Science*. 495–504.
[25] Yuchao Tao, Xi He, Ashwin MacHanavajjhala, and Sudeepa Roy. 2020. Computing local sensitivities of counting queries with joins. In *Proceedings of the 2020 ACM SIGMOD International Conference on Management of Data*. Association for Computing Machinery, New York, NY, USA, 479–494.
[26] Salil Vadhan. 2017. *The complexity of differential privacy*. Springer International Publishing, Cham. 347–450 pages.
[27] Royce J Wilson, Celia Yuxin Zhang, William Lam, Damien Desfontaines, Daniel Simmons-Marengo, and Bryant Gipson. 2020. Differentially private SQL with bounded user contribution. In *Proceedings on Privacy Enhancing Technologies Symposium*.

A EQUIVALENCE WITH THE DP MODEL FOR RELATIONAL DATABASES WITH FOREIGN KEY CONSTRAINTS

In this section, we show that our user-DP model is equivalent to the one for relational databases with foreign key constraints [10, 18]. This section assumes familiarity with relational algebra.

The DP model in [10, 18]. Given a database schema $\mathbf{R}$ and an instance $\mathbf{I}$ over $\mathbf{R}$, let $\mathbf{I}(R)$ denote the relation instance of each $R \in \mathbf{R}$ in $\mathbf{I}$. There is a designated relation $R_u \in \mathbf{R}$ that stores all the users. The DP model in [10, 18] aims at protecting all information associated with each user, which is defined based on the referencing relationship, defined recursively as follows. First, any user $t_u \in \mathbf{I}(R_u)$ is said to references itself. Then for any user $t_u \in \mathbf{I}(R_u)$, and any tuples $t_i \in \mathbf{I}(R_i)$, $t_j \in \mathbf{I}(R_j)$, if t_i references t_u, R_j has a foreign key (FK) referencing the primary key (PK) of R_i, and the FK of t_j equals to the PK of t_i, then t_j references the user t_u. Two instances $\mathbf{I}$ and $\mathbf{I}'$ are neighbors if one can be obtained from another by deleting a set of tuples, all referencing the same user $t_u^* \in \mathbf{I}(R_u)$, which is called the *witness*.

The following class of queries are considered. We start with a multi-way (natural) join J

$$J := R_1(\mathbf{x}_1) \bowtie \cdots \bowtie R_k(\mathbf{x}_k),$$

where $\mathbf{x}_i$ is the variables of R_i. There can be self-joins in J, i.e., it is possible that $R_i = R_j$ for $i \neq j$; in this case, we must have $\mathbf{x}_i \neq \mathbf{x}_j$. The join is required to be *complete*, i.e., if any R_i in J has an FK referencing the PK of some R', then R' must also appear in J, with its PK given a variable that is the same as the variable given to the FK of R_i. Note that if R_i has multiple FKs, or it appears multiple times with its FK given different variables, then this will require R' to appear multiple times with its PK given different variables. Eventually, R_u may appear multiple times in J.

Next, the query specifies two functions $\psi : \mathbf{dom}(var(J)) \to \mathbb{N}$ and $f : \mathbb{N}^{\mathbb{N}} \to [D]$, and the query output is defined as

$$f\left(\biguplus_{q \in J(\mathbf{I})} \{\psi(q)\}\right),$$

where $J(\mathbf{I})$ denotes the join results on instance $\mathbf{I}$.

The equivalence. First, consider any database instance $\mathbf{I}$ in the DP model above. We first remove all the dangling tuples that contribute nothing to the join result $J(\mathbf{I})$. Suppose R_u appears ℓ times in the join. Then we construct an instances V in our user-DP model as follows. We group the join results $J(\mathbf{I})$ according to the contributing users. For each set x of ℓ users, we set $V(x) := \{\psi(q) \mid q \in J(\mathbf{I}), q \text{ contains } x\}$. It is clear that $\biguplus_{q \in J(\mathbf{I})} \{\psi(q)\} = \biguplus_x V(x)$, so the query results are the same. Next, consider two neighboring database instances $\mathbf{I} \sim \mathbf{I}'$. Without loss of generality, assume $\mathbf{I}$ contains the witness t_u^*. Any different join result in $J(\mathbf{I}) \setminus J(\mathbf{I}')$ must be associated to some different tuple referencing t_u^*. Since the join J is complete, the different join result must be contributed by the user t_u^*. Thus, the distance between the neighboring instances under our setting is 1.

For the other direction, we will use a simple schema with just two relations. The first relation, R_u, has only one column containing the user id's. The second relation, R_V, has $\ell + 1$ columns, the first ℓ of which are all FKs referencing the only column in R_u. Given any instance V in our user-DP model, we construct a database instance $\mathbf{I}$ as follows. For each subset of users $x = \{u_1, \ldots, u_\ell\} \in \binom{\mathcal{U}}{\leq \ell}$ and and any $t \in V(x)$, we add a tuple $\langle u_1, \ldots, u_\ell, t\rangle$ into the relation $\mathbf{I}(R_V)$. In case x contains less than ℓ users, we simply duplicate some user, say, u_1. Then we define the join

$$J := R_u(\mathbf{u}_1) \bowtie \cdots \bowtie R_u(\mathbf{u}_\ell) \bowtie R_V(\mathbf{u}_1, \ldots, \mathbf{u}_\ell, \mathbf{y})$$

and set $\psi(q) = \pi_{\mathbf{y}} q$ for each $q \in J(\mathbf{I})$. Thus, it is still the case that $\biguplus_{q \in J(\mathbf{I})} \{\psi(q)\} = \biguplus_x V(x)$. Now consider any neighboring instances $V \sim_{u^*} V'$ in our user-DP model (assuming $V' \preceq V$ without loss of generality), as all the different tuples in the mappings are contributed by the witness u^*, the distance between the corresponding instances $\mathbf{I}$ and $\mathbf{I}'$ is 1.