Differential Privacy of Hierarchical Census Data: An Optimization Approach

Ferdinando Fioretto and Pascal Van Hentenryck

Georgia Institute of Technology
fioretto@gatech.edu pvh@isye.gatech.edu

Abstract. This paper is motivated by applications of a Census Bureau interested in releasing aggregate socio-economic data about a large population without revealing sensitive information. The released information can be the number of individuals living alone, the number of cars they own, or their salary brackets. Recent events have identified some of the privacy challenges faced by these organizations. To address them, this paper presents a novel differential-privacy mechanism for releasing hierarchical counts of individuals satisfying a given property. The counts are reported at multiple granularities (e.g., the national, state, and county levels) and must be consistent across levels. The core of the mechanism is an optimization model that redistributes the noise introduced to attain privacy in order to meet the consistency constraints between the hierarchical levels. The key technical contribution of the paper shows that *this optimization problem can be solved in polynomial time by exploiting the structure of its cost functions*. Experimental results on very large, real datasets show that the proposed mechanism provides improvements up to two orders of magnitude in terms of computational efficiency and accuracy with respect to other state-of-the-art techniques.

1 Introduction

The release of datasets containing sensitive information about a large number of individuals is central to a number of statistical analysis and machine learning tasks. For instance, the US Census Bureau publishes socio-economic information about individuals, which is then used as input to train classifiers/predictors and release important statistics about the US population.

One of the fundamental roles of a Census Bureau is to report *group size* queries, which are especially useful to study the skewness of a distribution. For instance, in 2010, the US Census Bureau released 33 datasets of such queries [24]. Group size queries partition a dataset in *groups* and evaluate the size of each group. For instance, a group may be the households that are families of four members, or the households owning three cars.

The challenge is to release these datasets without disclosing sensitive information about any individual in the dataset. Various techniques for limiting a-priori the disclosed information have been investigated in the past, including anonymization [22] and aggregations [25]. However, these techniques have been

consistently shown ineffective in protecting sensitive data [14,22], For instance, the US Census Bureau confirmed [2] that the disclosure limitations used for the 2000 and 2010 censuses had serious vulnerabilities which were exposed by the Dinur and Nissim's reconstruction attack [6]. Additionally, the 2010 Census group sizes were truncated due to the lack of privacy methods for protecting these particular groups [4].

This paper addresses these limitations through the framework of *Differential Privacy* [7], that offers a formal approach to guarantee data privacy by bounding the disclosure risk of any individual participating in a dataset. Differential privacy is considered the de-facto standard for privacy protection and has been adopted by various corporations [9,23] and governmental agencies [1]. It works by injecting carefully calibrated noise to the data before release. However, while this process guarantees privacy, it also affects the fidelity of the released data. In particular, the injected noise often produces datasets that violate consistency constraints of the application domain. In particular, group size queries must be consistent in a geographical hierarchy, e.g., the national, state, and county levels. Unfortunately, the traditional injection of independent noise to the group sizes cannot ensure the consistency of hierarchical constraints.

To overcome this limitation, this paper casts the problem of privately releasing group size data as a *constraint optimization problem* that ensures consistency of the hierarchical dependencies. However, the optimization problem that redistributes noise optimally is intractable for real datasets involving hundreds of millions of individuals. In fact, even its convex relaxation, which does not guarantee consistency, is challenging computationally. This paper addresses these challenges by proposing mechanisms based on a dynamic programming scheme that leverages both the hierarchical nature of the problem and the structure of the objective function. The contributions of the paper are summarized as follows:

1. The paper introduces the *Privacy-preserving Group Size Release* (PGSR) problem, for releasing differentially private group sizes that preserves hierarchical consistency.
2. It proposes a differentially private mechanism that uses an optimization approach to release both accurate and consistent group sizes.
3. It shows that the differentially private mechanism can be implemented in polynomial time, using a dynamic program that exploits both the hierarchical nature of group size queries and the structure of the objective function.
4. Finally, it evaluates the mechanisms on very large datasets containing over 300,000,000 individuals. The results demonstrate the effectiveness and scalability of the proposed mechanisms that bring several orders of magnitude improvements over the state of the art.

2 Problem Specification

This paper is motivated by applications from the US Census Bureau, whose goal is to release socio-demographic features of the population grouped by census

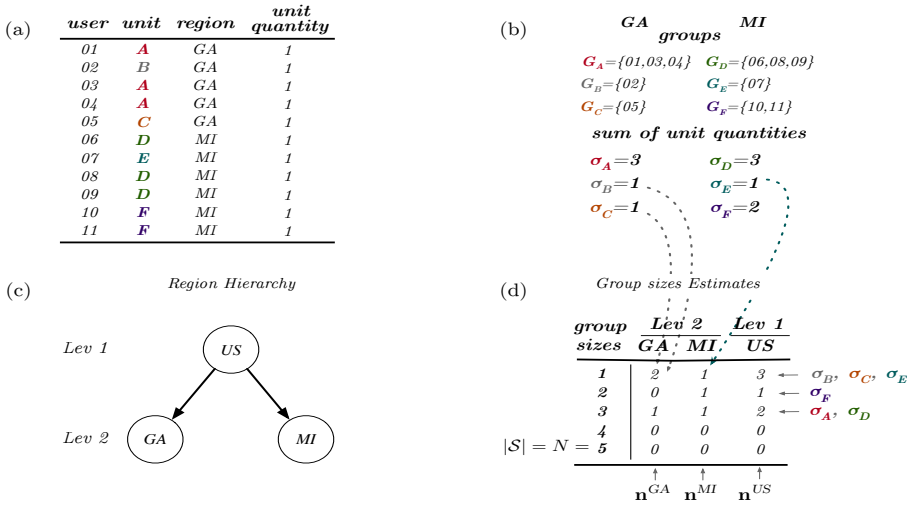


Fig. 1. A dataset D associating users to units and regions (a) and its associated groups and sum of units(b), region hierarchy (c), and the final hierarchical group-sizes (d).

blocks, counties, and states. For instance, the bureau is interested in releasing information such as the number of people in a household and how many cars they own. This section provides a generic formalization of this release problem.

Consider a dataset $D = \{(p_i, u_i, r_i, z_i)\}_{i=1}^{n}$ containing n tuples $(p_i, u_i, r_i, z_i) \in \mathcal{P} \times \mathcal{U} \times \mathcal{R} \times \mathcal{Z}$ denoting, respectively, a (randomly generated) identifier for user $i \in [n]$, its *unit identifier* (e.g., the home address where she lives), the *region* in which she lives, (e.g., a census block), and a *unit quantity* describing a socio-demographic feature, e.g., the number of cars she owns, or her salary bracket. The set of users sharing the same unit forms a *group* and $G_u = \{p_i \in \mathcal{P} \mid u_i = u\}$ denotes the group of unit u. The socio-demographic feature of interest is the sum of the *unit quantities* of a group G_u, i.e., $\sigma_u = \sum_{p_i \in G_u} z_i$. The set of all unit sizes, i.e., $\mathcal{S} \supseteq \{\sigma_u | u \in \mathcal{U}\}$, also plays an important role. Indeed, the bureau is interested in releasing, for every unit size $\sigma \in \mathcal{S}$, the quantity $n_\sigma = |\{G_u \mid u \in \mathcal{U}, \sigma_u = \sigma\}|$, i.e., the number of groups of size σ.

These concepts are illustrated in Figure 1(a), which shows a dataset containing $n = 11$ users with their home addresses (units), their states (regions), and a 0/1 quantity denoting a feature of interest. In the running example, the feature is always 1, since the application is interested is the composition of the household, i.e., how many people live at the same address. Hence the *groups* identify households and the *sums of unit quantities* represent household sizes. For instance, G_A is the group of 3 users living in unit A and $\sigma_A = 3$ as shown in Figure 1(b). The example also uses $\mathcal{S} = \{1, \ldots, 5\}$.

In addition to the dataset, the census bureau works with a *region hierarchy* that is formalized by a tree $\mathcal{T}_L$ of L levels. Each level $\ell \in [L]$ is associated with a set of regions $\mathcal{R}_\ell \subseteq \mathcal{R}$, forming a partition on D. Region r' is a subregion of region r, which is denoted by $r' \prec r$, if r' is contained in r and $\text{lev}(r') = \text{lev}(r)+1$,

where lev(r) denotes the level of r. The root level contains a single region $r^\top$. The children of r, i.e., $ch(r) = \{r' \in \mathcal{R} | r' \prec r\}$ is the set of regions that partition r in the next level of the hierarchy and $pa(r)$ denotes the parent of region r ($r \neq r^\top$). Figure 1(c) provides an illustration of a hierarchy of 2 levels. Each node represents a region. The regions *GA* and *MI* form a partition of region *US*.

The number of groups with size $\sigma \in \mathcal{S}$ and region $r \in \mathcal{R}$ is denoted by $n^r_\sigma = |\{u \in \mathcal{U} \mid \sigma_u = \sigma \wedge u \in r\}|$ and $\boldsymbol{n}^r = (n^r_1, \ldots, n^r_N)$ denotes the vector of *group sizes* for region r, where $N = |\mathcal{S}|$. Figure 1(d) illustrates the group sizes for each group size $s \in [N = 5]$ with: $\boldsymbol{n}^{GA} = (2, 0, 1, 0, 0)$, $\boldsymbol{n}^{MI} = (1, 1, 1, 0, 0)$, and $\boldsymbol{n}^{US} = (3, 1, 2, 0, 0)$.

It is now possible to define the problem of interest to the bureau: *The goal is to release, for every group size $s \in [N]$ and region $r \in \mathcal{R}$, the numbers n^r_s of groups of size s in region r, while preserving individual privacy.* The region hierarchy and the group sizes $\mathcal{S}$ are considered public *non-sensitive* information. The entries associating users with groups (see Figure 1(a)) are *sensitive* information. Therefore, the paper focuses on protecting the privacy of such information. For simplicity, this paper assumes that the region hierarchy has exactly L levels and uses $\mathcal{T}$ as a shorthand for $\mathcal{T}_L$. The paper also focuses on the vastly common case when $z_i \in \{0, 1\}, (i \in [n])$, but the results generalize to arbitrary z_i values.

3 Differential Privacy

This paper adopts the framework of differential privacy [7,8], which is the de-facto standard for privacy protection.

Definition 1 (Differential Privacy [7]). *A randomized algorithm $\mathcal{M} : \mathscr{D} \to \mathscr{R}$ with domain $\mathscr{D}$ and range $\mathscr{R}$ is* ϵ-differentially private *if*

$$\Pr[\mathcal{M}(D_1) \in O] \leqslant \exp(\epsilon) \Pr[\mathcal{M}(D_2) \in O], \tag{1}$$

for any output response $O \in \mathscr{R}$ and any two datasets $D_1, D_2 \in \mathscr{D}$ differing in at most one individual (called neighbors *and written $D_1 \sim D_2$).*

Parameter $\epsilon > 0$ is the *privacy budget* of the algorithm, with values close to 0 denoting strong privacy. Intuitively, the definition states that the probability of any event does not change much when a single individual data is added or removed to the dataset, limiting the amount of information that the output reveals about any individual.

This paper relies on the *global sensitivity method* [7]. The global sensitivity Δ_q of a function $q : \mathscr{D} \to \mathbb{R}^k$ (also called *query*) is defined as the maximum amount by which q changes when a single individual is added to, or removed from, a dataset:

$$\Delta_q = \max_{D_1 \sim D_2} \|q(D_1) - q(D_2)\|_1. \tag{2}$$

Queries in this paper concern the group size vectors $\boldsymbol{n}^r$ and neighboring datasets differ by the presence or absence of at most one record (see Figure 1(a)).

The global sensitivity is used to calibrate the amount of noise to add to the query output to achieve differential privacy. There are several sensitivity-based mechanisms [7,20] and this paper uses the *Geometric* mechanism [13] for *integral* queries. It relies on a double-geometric distribution and has slightly less variance than the ubiquitous Laplace mechanism [7].

Definition 2 (Geometric Mechanism [13]). *Given a dataset D, a query $q \in \mathbb{R}^k$, and $\epsilon > 0$, the geometric mechanism adds independent noise to each dimension of the query output $q(D)$ using the distribution: $P(X = v) = \frac{1-e^{-\epsilon}}{1+e^{-\epsilon}} e^{(-\epsilon|v|/\Delta_q)}$.*

This distribution is also referred to as *double-geometric* with scale Δ_q/ϵ. In the following, $Geom(\lambda)^k$ denotes the i.i.d. double-geometric distribution over k dimensions with parameter λ. The Geometric Mechanism satisfies ϵ-differential privacy [13]. Differential privacy also satisfies several important properties [8].

Lemma 1 (Sequential Composition). *The composition of two ϵ-differentially private mechanisms ($\mathcal{M}_1$, $\mathcal{M}_2$) satisfies 2ϵ-differential privacy.*

Lemma 2 (Parallel Composition). *Let D_1 and D_2 be disjoint subsets of D and $\mathcal{M}$ be an ϵ-differential private algorithm. Computing $\mathcal{M}(D \cap D_1)$ and $\mathcal{M}(D \cap D_2)$ satisfies ϵ-differential privacy.*

Lemma 3 (Post-Processing Immunity). *Let $\mathcal{M}$ be an ϵ-differential private algorithm and g be an arbitrary mapping from the set of possible output sequences O to an arbitrary set. Then, $g \circ \mathcal{M}$ is ϵ-differential private.*

4 The Privacy-Preserving Group Size Release Problem

This section formalizes the Privacy-preserving Group Size Release (PGSR) problem. Consider a dataset D, a region hierarchy $\mathcal{T}$ for D, where each node a^r in $\mathcal{T}$ is associated with a vector $\boldsymbol{n}^r \in \mathbb{Z}_+^N$ describing the group sizes for region $r \in \mathcal{R}$, and let $G = \sum_{s\in[N]} n_s^\top$ be the total number of individual groups in D, which is public information (see Figure 2(a) for an example). The PGSR problem consists in releasing a hierarchy of group sizes $\tilde{\mathcal{T}} = \langle \tilde{\boldsymbol{n}}^r \mid r \in \mathcal{R} \rangle$[1] that satisfies the following conditions:

1. *Privacy*: $\tilde{\mathcal{T}}$ is ϵ-differentially private.
2. *Consistency*: For each region $r \in \mathcal{R}$ and group size $s \in \mathcal{S}$, the group sizes in the subregions r' of r add up to those in region r: $\tilde{n}_s^r = \sum_{r' \in ch(r)} \tilde{n}_s^{r'}$.
3. *Validity*: The values $\tilde{n}_s^r$ are non-negative integers.
4. *Faithfulness*: The group sizes at each level ℓ of the hierarchy add up to the value G: $\sum_{r\in\mathcal{R}_\ell} \sum_{s\in[N]} \tilde{n}_s^r = G$.

These constraints ensure that the hierarchical group size estimates satisfy all publicly known properties of the original data.

[1] We abuse notation and use the angular parenthesis to denote a hierarchy.

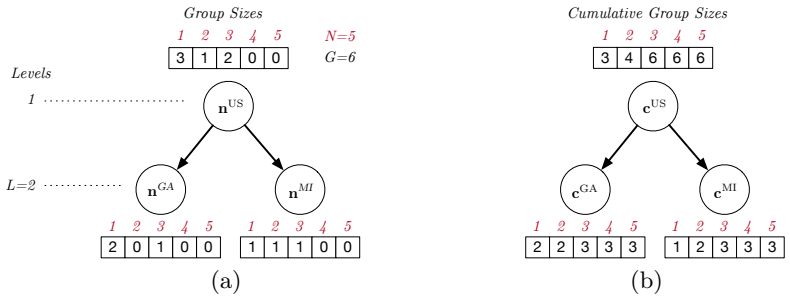


Fig. 2. Region hierarchies associated with the dataset of Figure 1(a): *group size* hierarchy $\mathcal{T}_2$ (a), and *cumulative group size* hierarchy $\mathcal{T}_2^c$ (b).

5 The Direct Optimization-Based PGSR Mechanism

This section presents a two-step mechanism for the PGSR problem. The first step produces a noisy version of the group sizes, while the second step restores the feasibility of the PGSR constraints while staying as close as possible to the noisy counts. The first step produces a noisy hierarchy $\tilde{\mathcal{T}} = \{\tilde{\boldsymbol{n}}^r \mid r \in \mathcal{R}\}$ using the geometric mechanism with parameter $\lambda = \frac{2L}{\epsilon}$ on the vectors $\boldsymbol{n}^r$:

$$\tilde{\boldsymbol{n}}^r = \boldsymbol{n}^r + Geom\Big(\frac{2L}{\epsilon}\Big)^N. \tag{3}$$

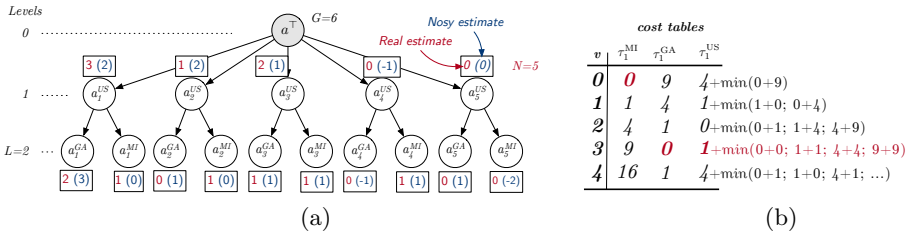
This step satisfies ϵ-differential privacy due to the following lemma.

Lemma 4. *The sensitivity $\Delta_{\boldsymbol{n}}$ of the group estimate query is 2.*

$$\begin{aligned}
\underset{\{\hat{\boldsymbol{n}}^r\}_{r\in\mathcal{R}}}{\text{minimize}} \quad & \sum_{r\in\mathcal{R}} \|\hat{\boldsymbol{n}}^r - \tilde{\boldsymbol{n}}^r\|_2^2 && \text{(H1)}\\
\text{s.t:} \quad & \sum_{s\in[N]} \hat{n}_s^r = G \quad \forall r \in \mathcal{R} && \text{(H2)}\\
& \sum_{c\in ch(r)} \hat{n}_s^c = \hat{n}_s^r \quad \forall r \in \mathcal{R}, s \in [N] && \text{(H3)}\\
& \hat{n}_s^r \in D_s^r \quad \forall r \in \mathcal{R}, s \in [N] && \text{(H4)}
\end{aligned}$$

Fig. 3. The $\mathcal{M}_H$ post-processing step.

The output of the first step satisfies Condition 1 of the PGSR problem but it will violate (with high probability) the other conditions. To restore feasibility, this paper uses a post-processing strategy similar to the one proposed in [10] for mobility applications. After generating $\tilde{\mathcal{T}}$ using Equation (3), the mechanism post-processes the values $\tilde{\boldsymbol{n}}^r$ of $\tilde{\mathcal{T}}$ through the *Quadratic Integer Program (QIP)* depicted in Figure 3. Its goal is to find a new region hierarchy $\hat{\mathcal{T}}$, optimizing over the variables $\hat{\boldsymbol{n}}^r = (\hat{n}_1^r \dots \hat{n}_N^r)$ for each $r \in \mathcal{R}$, so that their values stay close to the noisy counts of the first step, while satisfying faithfulness (Constraint (H2)), consistency (Constraint (H3)), and validity (Constraint (H4)). In the optimization model, D_s^r represents the domain (of integer, non-negative values) of $\hat{n}_s^r$. The resulting mechanism is called the *Hierarchical PGSR* and denoted by $\mathcal{M}_H$. It satisfies ϵ-differential privacy because of post-processing immunity of differential privacy (Lemma 3), since the post-processing step of $\mathcal{M}_H$ uses exclusively differentially private information ($\tilde{\mathcal{T}}$).



(a) (b)

Fig. 4. (a): Region hierarchy $\mathcal{T}^{dp}$ associated with the dataset of Figure 1(a). (b): Example of cost table computation for subtree associated to the group 1 estimates.

Solving this QIP is intractable for the datasets of interest to the census bureau. Therefore, the experimental results consider a version of $\mathcal{M}_H$ that *relaxes the integrability constraint* (H4) and rounds the solutions. The resulting optimization problem becomes convex but presents two limitations: *(i)* its final solution $\hat{T}$ may violate the PGSR *consistency* (2) and *faithfulness* (4) conditions, and *(ii)* the mechanism is still too slow for very large problems.

6 The Dynamic Programming PGSR Mechanism

This section proposes a dynamic-programming approach for the post-processing step to remedy the limitation of $\mathcal{M}_H$ and its convex relaxation. The resulting mechanism is called the *Dynamic Programming PGSR* mechanism and denoted by $\mathcal{M}_H^{\text{dp}}$. The dynamic program relies on a new hierarchy $\mathcal{T}^{\text{dp}}$ that modifies the original region hierarchy $\mathcal{T}$ as follows. It creates as many subtrees as the number N of groups in $\mathcal{S}$. The nodes of subtree s represent the groups of size s for the regions in $\mathcal{R}$. In other words, the root node a_s^r of subtree s–where r is level 1 in the region hierarchy (see Figure 1(b))–is associated with the number n_s^r of groups of size s in region r. Its children $\{a_s^c\}_{c \in ch(r)}$ are associated with the numbers n_s^c, and so on. Finally, the new hierarchy has a root note $a^\top$ that represents the total number G of groups: It is associated with a *dummy* region $\top$ whose children are the N subtrees introduced above. The resulting region hierarchy is denoted $\mathcal{T}^{dp}$.

Example 1. The region hierarchy $\mathcal{T}^{dp}$ associated with the running example is shown in Figure 4(a). The root node $a^\top$ is associated with the total number of groups in D, i.e., $G = 6$. Its children $a_1^{US}, \ldots, a_5^{US}$ represent the group sizes for the root of the region hierarchy for each group size $s \in [N = 5]$. Subtree 1, rooted at a_1^{US}, has two children: a_1^{GA} and a_1^{MI}, representing the number of groups of size 1: n_1^{GA} and n_1^{MI}. The figure illustrates the association of every node a_s^r with its *real* group size n_s^r (in red) and its noisy group size generated by the geometrical mechanism (in blue and parenthesis).

Note that *(i)* the value of a node equals to the sum of the values of its children, *(ii)* the sum of the group sizes at a given level add up to G, and *(iii)* the PGSR consistency conditions of the nodes in a subtree are independent of those of other subtrees. These observations allow us to develop a dynamic program that

guarantees the PGSR conditions and exploits the independence of each subtree associated with groups of size s to solve the post-processing problem efficiently.

For notational simplicity, the presentation omits the subscripts denoting the group size s and focuses on the computation of a single subtree representing a group of size s. The dynamic program associates a *cost table* $\boldsymbol{\tau}^r$ with each node a^r of $\mathcal{T}^{dp}$. The cost table represents a function $\boldsymbol{\tau}^r : D^r \rightarrow \mathbb{R}_+$ that maps values (i.e., group sizes) to costs, where D^r is the *domain* (a set of natural numbers) of region r. Intuitively, $\boldsymbol{\tau}^r(v)$ is the optimal cost for the post-processed group sizes in the subtree rooted at a^r when its post-processed group size is equal to v, i.e., $\hat{n}^r = v$. *The key insight of the dynamic program is the observation that the optimal cost for* $\boldsymbol{\tau}^r(v)$ *can be computed from the cost tables* $\boldsymbol{\tau}^c$ *of each of its children* $c \in ch(r)$ *using the formula given in Figure 5.* In the formula, (d1) describes the cost for v of deviating from the noisy group size $\tilde{n}^r$. The function $\boldsymbol{\phi}^r(v)$, defined in (d2), (d3), and (d4), uses the cost table of the children of r to find the combination of post-processed group sizes $\{x_c \in D^c\}_{c \in ch(r)}$ of r's children that is consistent (d3) and minimizes the sum of their costs (d2).

The dynamic program exploits these concepts in two phases. The first phase is bottom-up and computes the cost tables for each node, starting from the leaves only, which are defined by (d1), and moving up, level by level, to the root. The cost table at the root is then used to retrieve the optimal cost of the problem. The second phase is top-down: Starting from the root, each node a^r receives its post-processed group size $\hat{n}^r$ and solves $\boldsymbol{\phi}^r(\hat{n}^r)$ to retrieve the optimal post-processed group sizes $\hat{n}^c = x_c$ for each child $c \in ch(r)$.

$$\boldsymbol{\tau}^r(v) = \left(v - \tilde{n}^r\right)^2 + \quad \text{(d1)}$$

$$\boldsymbol{\phi}^r(v) = \min_{\{x_c\}_{c \in ch(r)}} \sum_{c \in ch(r)} \boldsymbol{\tau}^c(x_c) \quad \text{(d2)}$$

$$\text{s.t.} \sum_{c \in ch(r)} x_c = v \quad \text{(d3)}$$

$$x_c \in D^c \quad \forall c \in ch(r) \quad \text{(d4)}$$

Fig. 5. The $\mathcal{M}_H^{dp}$ Bottom-up step.

An illustration of the process for the running example is illustrated in Figure 4(b). It depicts the cost tables $\boldsymbol{\tau}_1^{GA}, \boldsymbol{\tau}_1^{MI}$, and $\boldsymbol{\tau}_1^{US}$ related to the subtree rooted at a_1^{US} (groups of size 1) computed during the bottom-up phase. The values selected during the top-down phase are highlighted red.

In the implementation, the values $\boldsymbol{\phi}^r(v)$ are computed using a constraint program where (d1) is implemented using a table constraint. The number of optimization problems in the dynamic program is given by the following theorem.

Theorem 1. *Constructing* $\hat{\mathcal{T}}^{dp}$ *requires solving* $O(|\mathcal{R}|N\bar{D})$ *optimization problems given in Figure 5, where* $\bar{D} = \max_{s,r} |D_s^r|$ *for* $r \in \mathcal{R}, s \in [N]$.

7 A Polynomial-Time PGSR Mechanism

The dynamic program relies on solving an optimization problem for each region. This section shows that this optimization problem can be solved in polynomial time by exploiting the structure of the cost tables.

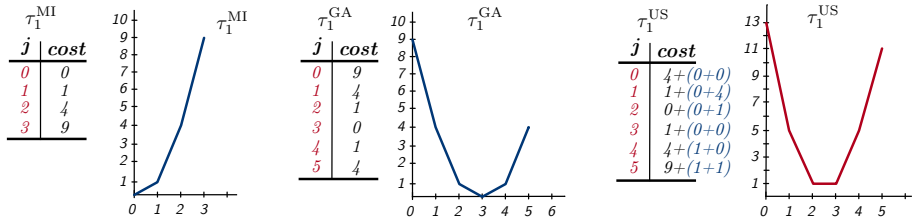


Fig. 6. Cost tables, extending Figure 4(b), computed by the mechanism.

A cost table is a finite set of pairs (s, c) where s is a group size and c is a cost. When the pairs are ordered by increasing values of s and line segments are used to connect them as in Figure 6, the resulting function is Piecewise Linear (PWL). For simplicity, we say that a cost table is PLW if its underlying function is PWL. Observe also that, at a leaf, the cost table is Convex PWL (CPWL), since the L2-Norm is convex (see Equation (d1)).

The key insight behind the polynomial-time mechanism is the recognition that the function $\boldsymbol{\phi}^r$ is CPWL whenever the cost tables of its children are CPWL. As a result, by induction, the cost table of every node a^r is CPWL.

Lemma 5. *The cost table $\boldsymbol{\tau}_s^r$ of each node a_s^r of $\mathcal{T}^{\text{dp}}$ is CPWL.*

Lemma 5 makes it possible to design a polynomial-time algorithm to compute $\boldsymbol{\tau}^r$ (subscripts omitted for succinctness) that replaces the constraint program used in the dynamic program. We give the intuition underlying the algorithm.

Given a node a^r, the first step of the algorithm is to select, for each node $c \in ch(r)$, the value v_c^0 with minimum cost, i.e., $v_c^0 = \text{argmin}_v\, \boldsymbol{\tau}^c(v)$. As a result, the value $V^0 = \sum_c v_c^0$ has minimal cost $\boldsymbol{\phi}^r(V^0) = \sum_c \boldsymbol{\tau}^c(v_c^0)$. Having constructed the minimum value in cost table $\boldsymbol{\phi}^r$, it remains to compute the costs of all values $V^0 + k$ for all integer $k \in [1, \max D^r - V^0]$ and all values $V^0 - k$ for all integer $k \in [1, V^0 - \min D^r]$. The presentation focuses on the values $V^0 + k$ since the two cases are similar. Let $\mathbf{v}^0 = \{v_c^0\}_{c \in ch(r)}$. The algorithm builds a sequence of vectors $\mathbf{v}^1, \mathbf{v}^2, \ldots, \mathbf{v}^k, \ldots$ that provides the optimal combinations of values for $\boldsymbol{\phi}^r(V^0+1), \boldsymbol{\phi}^r(V^0+2), \ldots, \boldsymbol{\phi}^r(V^0+k), \ldots$. Vector $\mathbf{v}^k$ is obtained from $\mathbf{v}^{k-1}$ by changing the value of a single child whose cost table has the smallest slope, i.e.,

$$v_c^k = \begin{cases} v_c^{k-1} + 1 & \text{if } \ c = \text{argmin}_c\, \boldsymbol{\tau}^c(v_c^{k-1}+1) - \boldsymbol{\tau}^c(v_c^{k-1}) \\ v_c^{k-1} & \text{otherwise.} \end{cases} \tag{4}$$

Once $\boldsymbol{\phi}^r$ has been computed, cost table $\boldsymbol{\tau}^r$ can be computed easily since both $(v - \tilde{n}^r)^2$ and $\boldsymbol{\phi}^r$ are CPWL and the sum of two CPWL functions is CPWL.

Example 2. These concepts are illustrated in Figure 6, where the values of $\boldsymbol{\phi}^r$ are highlighted in blue, and in parenthesis, in the right table. The first step identifies that $v_{MI}^0 = 0$, $v_{GA}^0 = 3$, thus $V^0 = 3$ and $\boldsymbol{\phi}^r(3) = \boldsymbol{\tau}^{MI}(0) + \boldsymbol{\tau}^{GA}(3) = 0 + 0 = 0$. In the example, $\mathbf{v}^0 = (v_{MI}^0, v_{GA}^0) = (0, 3)$ and $\mathbf{v}^1 = (v_{MI}^1, v_{GA}^1) = (1, 3)$, since $MI = \text{argmin}\{\boldsymbol{\tau}^{MI}(0+1) - \boldsymbol{\tau}^{MI}(0), \boldsymbol{\tau}^{GA}(3+1) - \boldsymbol{\tau}^{GA}(3)\} = \text{argmin}\{1-0, 1-0\}$. Its associated cost, $\boldsymbol{\phi}^{US}(\mathbf{v}^1) = 1+0 = 1$. $\mathbf{v}^2 = (1, 4)$ since $GA = \text{argmin}\{(\boldsymbol{\tau}^{MI}(1+$

$1)-\boldsymbol{\tau}^{MI}(1)), (\boldsymbol{\tau}^{GA}(3+1)-\boldsymbol{\tau}^{GA}(3))\} = \text{argmin}\{(4-1),(1-0)\}$, and its associated cost $\boldsymbol{\phi}^{US}(\mathbf{v}^2) = 1+1 = 2$.

Theorem 2. *The cost table $\boldsymbol{\tau}_s^r$ of each node a_s^r of $\mathcal{T}^{dp}$ is CPWL.*

Theorem 3. *The cost table $\boldsymbol{\tau}_s^r$ for each region r and size s can be computed in time $O(\bar{D}\log\bar{D})$.*

8 Cumulative PGSR Mechanisms

In the mechanisms presented so far, each query has sensitivity $\Delta_{\boldsymbol{n}} = 2$. This section exploits the structure of the group query to reduce the query sensitivity and thus the noise introduced by the geometric mechanism.

Define the operator $\oplus : \mathbb{Z}_+^N \to \mathbb{Z}_+^N$ that, given a vector $\boldsymbol{n} = (n_1, \ldots, n_N)$ of group sizes, returns its cumulative version $\boldsymbol{c} = (c_1, \ldots, c_N)$ where $c_s = \sum_{k=1}^{s} n_k$ is the cumulative sum of the first s elements of $\boldsymbol{n}$. This operator can be used to produce a hierarchy $\mathcal{T}^c = \{\boldsymbol{c}^r | r \in \mathcal{R}\}$ of cumulative group sizes. An example of such a hierarchy $\mathcal{T}^c$ is provided in Figure 2(b).

Lemma 6. *The sensitivity $\Delta_{\boldsymbol{c}}$ of the cumulative group estimate query is 1.*

The result follows from the fact that removing an element from a group in $\boldsymbol{c}$ only affects the group preceding it. This idea is from [15], where cumulative sizes are referred to as *unattributed histograms*.

To generate a privacy-preserving version $\tilde{\mathcal{T}}^c$ of $\mathcal{T}^c$, it suffices to apply the geometrical mechanism with parameter $\lambda = (L/\epsilon)^N$ on the vectors $\boldsymbol{c}^r$ associated with every node $\boldsymbol{\tau}^r$ of the region hierarchy. Once the noisy sizes are computed, the noisy group sizes can be easily retrieved via an inverse mapping $\ominus : \mathbb{Z}_+^N \to \mathbb{Z}_+^N$ from the cumulative sums.

Note however that the resulting private versions $\tilde{\boldsymbol{c}}^r$ of $\boldsymbol{c}^r$ may no longer be non-decreasing (or even non-negative) due to the added noise. Therefore, as in Section 5, a post-processing step is applied to restore consistency and to guarantee the PGSR conditions 2 to 4. The post-processing is illustrated in Figure 7. It takes as input the noisy hierarchy of cumulative sizes $\tilde{\mathcal{T}}^c$ computed with the geometrical mechanism and optimizes over variables $\hat{\boldsymbol{c}}^r = (\hat{c}_1^r, \ldots, \hat{c}_N^r)$ for $r \in \mathcal{R}$, minimizing the L2-norm wrt their noisy counterparts (Equation (C1)). Constraints (C2) guarantees that the sum of the sizes equals the public value G (PGSR condition 4), where $\top$ denotes the root region of the region hierarchy. Constraints (C3) guarantee consistency of the cumulative counts. Finally, Constraints (C4) and (C5), respectively, guarantee the PGSR consistency (2) and validity (3) conditions.

$$\underset{\{\hat{\boldsymbol{c}}^r\}_{r\in\mathcal{R}}}{\text{minimize}} \quad \sum_{r\in\mathcal{R}} \|\hat{\boldsymbol{c}}^r - \tilde{\boldsymbol{c}}^r\|_2^2 \qquad \text{(C1)}$$

$$\text{s.t: } \hat{c}_N^{\top} = G \qquad \text{(C2)}$$

$$\hat{c}_i^r \leqslant \hat{c}_{i+1}^r \quad \forall r \in \mathcal{R}, i \in [N-1] \qquad \text{(C3)}$$

$$\sum_{r'\in ch(r)} \hat{c}_i^{r'} = \hat{c}_i^r \quad \forall r \in \mathcal{R}, i \in [N] \qquad \text{(C4)}$$

$$\hat{c}_i^r \in \{0, 1, \ldots\} \quad \forall r \in \mathcal{R}, i \in [N] \qquad \text{(C5)}$$

Fig. 7. The $\mathcal{M}_c$ post-processing step.

Once the post-processed hierarchy $\hat{\mathcal{T}}^c$ is obtained, operator $\ominus$ is applied to obtain a post-processed version of the group size hierarchy $\hat{\mathcal{T}}$. It is easy to see that the above mechanism satisfies ϵ-differential privacy, due to post-processing immunity. This mechanism is called the cumulative PGSR and denoted by $\mathcal{M}_c$.

Like for $\mathcal{M}_H$, the structure of the PGSR problem can be exploited to create an efficient mechanism that satisfies the conditions of the PGSR problem using the cumulative group sizes. The resulting mechanism is called $\mathcal{M}_c^{dp}$ and operates in three steps:

1. $\mathcal{M}_c^{dp}$ creates a noisy hierarchy $\tilde{\mathcal{T}}^c$.
2. $\mathcal{M}_c^{dp}$ executes the post-processing step described in Algorithm 1.
3. $\mathcal{M}_c^{dp}$ runs the post-processing step of the polynomial-time PGSR mechanism (see Section 7).

Algorithm 1: $\mathcal{M}_c^{dp}$.*post-process*

input : $\tilde{\mathcal{T}}^c = \{\tilde{\boldsymbol{c}}^r \mid r \in \mathcal{R}\}$

1 **foreach** $r \in \mathcal{R}$ **do**

2 $\quad \hat{\boldsymbol{c}}^r \leftarrow \operatorname{argmin}_{\hat{\boldsymbol{c}}} \|\hat{\boldsymbol{c}} - \tilde{\boldsymbol{c}}^r\|^2$ s.t. $\hat{c}_i \leqslant \hat{c}_{i+1} \;\; \forall i \in [N-1]$, $0 \leqslant \hat{c}_i \leqslant G \;\; \forall i \in [N]$

3 $\quad \bar{\boldsymbol{n}} \leftarrow \ominus(round(\hat{\boldsymbol{c}}^r))$

4 $\quad \hat{\mathcal{T}} \leftarrow \bar{\mathcal{T}} \cup \{\bar{\boldsymbol{n}}\}$

5 $\hat{\mathcal{T}} \leftarrow \mathcal{M}_{dp}.post\text{-}process(\hat{\mathcal{T}})$

The novelty is in step 2 which takes $\tilde{\mathcal{T}}^c$ as input and, for each node $\tilde{\boldsymbol{c}}^r$, solves the *convex program* described in line 2 to create a new noisy hierarchy $\hat{\boldsymbol{c}}^r$ that is non-decreasing and non-negative. The resulting cumulative vector $\hat{\boldsymbol{c}}^r$ is then rounded and transformed to its corresponding group size vector through operator $\ominus(\cdot)$ (line 3). The resulting vector $\bar{\boldsymbol{n}}^r$ is added to the region hierarchy $\hat{\mathcal{T}}$ (line 4). Observe that this post-processing step pays a polynomial-time penalty w.r.t. the runtime of the $\mathcal{M}_H^{dp}$ post-processing. The convex program of line (2) is executed in $O(\text{poly}(N))$ and the resulting post-processing step runtime is in $O\big(|\mathcal{R}|\text{poly}(N) + |\mathcal{R}|N\bar{D}\log\bar{D}\big)$.

It is important to note that $\mathcal{M}_c^{dp}$ does not solve the same post-processing program as the cumulative PGSR mechanism specified by Equations (C1) to (C2), since it restores consistency of the cumulative counts locally. However, the experimental results show that it consistently reduces the final error: See Section 9 for detailed results.

9 Experimental Evaluation

This section evaluates the privacy-preserving mechanisms for the PGSR problem. The evaluation focuses on comparing runtime and accuracy. Consistent with the privacy literature, accuracy is measured in term of the L_1 difference between the privacy-preserving group sizes and the original ones, i.e., given the original group sizes $\mathcal{T} = \{\boldsymbol{n}^r \mid r \in \mathcal{R}\}$, and their private counterparts $\hat{\mathcal{T}}$, the L_1-error is defined as $\sum_r \|\boldsymbol{n}^r - \hat{\boldsymbol{n}}^r\|_1$. Since the mechanisms are nondeterministic due to the noise added by the geometric mechanism, 30 instances are generated for each benchmark and the results report average values and standard deviations. Each mechanism is run on a single-core 2.1 GHz terminal with 24GB of RAM and is implemented in Python 3 with Gurobi 8.0 for solving the convex quadratic optimization problems.

Mechanisms The evaluation compares the PGSR mechanisms $\mathcal{M}_H$, its cumulative version $\mathcal{M}_c$, and their polynomial-time dynamic-programming (DP) counterparts $\mathcal{M}_H^{dp}$, and $\mathcal{M}_c^{dp}$. The former are referred to as OP-based methods and the latter as DP-based methods. In addition to $\mathcal{M}_H$ and $\mathcal{M}_c$, that solve the associated post-processing QIPs, the experiments evaluate the associated *relaxations*, $\mathcal{M}_H^r$ and $\mathcal{M}_c^r$, respectively, that relax the integrality constraints (H4) and (C5) and rounds the solutions. For completeness, the experiments also evaluate the performance of the optimization-based mechanism $\mathcal{M}_{dp}^{\text{OP}}$ that *does not* exploit the structure of the cost function to compute the cost tables.

Datasets The mechanisms are evaluated on three datasets.

- **Census Dataset**: The first dataset has 117,630,445 groups, 7592 leaves, 305,276,358 individuals, 3 levels, and N=1,000. Individuals live in facilities, i.e., households or dormitories, assisted living facilities, and correctional institutions. Due to privacy concerns and lack of available methods to protect group sizes during the 2010 Decennial Census release, group sizes were aggregated for any facility of size 8 or more (see Summary File 1 [24]). Therefore, following [17] and starting from the truncated group sizes Census dataset, the experiments augment the dataset with group sizes up to $N = 1,000$ that mimic the published statistics, but add a heavy tail to model group quarters (dormitories, correctional facilities, etc.). This was obtained by computing the ratio $r = n_7/n_6$ of household groups of sizes 7 and 6, subtracting from the aggregated groups n_{8+} M people according to the ratio r, and redistributed these M people in groups $k > 8$ so that the ratio between any two consecutive groups holds (in expectation). Finally, 50 outliers were added, chosen uniformly in the interval between 10 and $1,000$. The region hierarchy is composed by the National level, the State levels (50 states + Puerto Rico and District of Columbia), and the Counties levels (3143 in total).
- **NY Taxi Dataset**: The second dataset has 13,282 groups, 3,973 leaves, 24,489,743 individuals, 3 levels, and N=13,282. The 2014 NY city Taxi dataset [3] describes trips (pickups and dropoffs) from geographical locations in NY city. The dataset views each taxi as a group and the size of the group is the number of pickups of the taxi. The region hierarchy has 3 levels: the entire NY city at level 1, the boroughs: *Bronx*, *Brooklyn*, *EWR*, *Manhattan*, *Queens*, and *Staten Island* at level 2, and a total of 263 zones at level 3.
- **Synthetic Dataset**: Finally, to test the runtime scalability, the experiments considered synthetic data from the NY Taxi dataset by limiting the number of group sizes N arbitrarily, i.e., removing group sizes greater than a certain threshold.

9.1 Scalability

The first results concern the scalability of the mechanisms, which are evaluated on the synthetic datasets for various numbers of group sizes. Figure 8 illustrates the runtimes of the algorithms at varying of the number of group sizes N from 5 to 50 for the synthetic dataset. The experiments

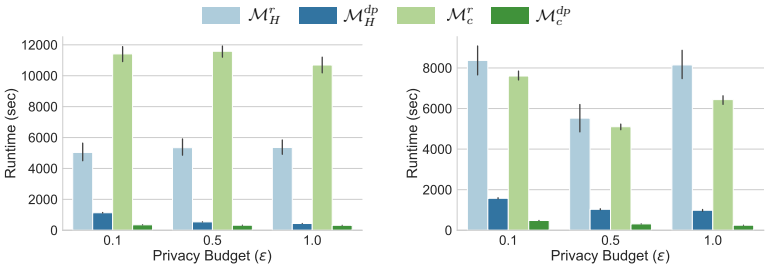


Fig. 9. The Runtime for the mechanisms: Census data (left) and Taxi data (right).

have a timeout of 30 minutes and the runtime is reported in log-10 scale. The figure shows that the exact OP-based approaches and $\mathcal{M}_{dp}^{\text{OP}}$ are not competitive, even for small groups sizes. Therefore, these results rule out the following mechanisms: $\mathcal{M}_{dp}^{\text{OP}}$, $\mathcal{M}_H$, and $\mathcal{M}_c$ and the remaining results focus on comparing the relaxed versions of the OP-based mechanisms versus their proposed DP-counterparts.

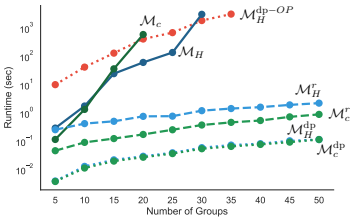


Fig. 8. Runtime (in seconds) at varying of the number of group size N.

9.2 Runtimes

Figure 9 reports the runtime, in seconds, for the hierarchical mechanism $\mathcal{M}_H^r$ and its DP-counterpart $\mathcal{M}_H^{dp}$, and the hierarchical cumulative mechanism $\mathcal{M}_c^r$ and its dp-counterpart $\mathcal{M}_c^{dp}$. The left side of the figure illustrates the results for the Census data and the right side those for the NY Taxi data. The main observations can be summarized as follows:

1. Although the OP-based algorithms consider only a relaxation of the problem, the *exact* DP-versions are consistently faster. In particular, $\mathcal{M}_H^{dp}$ is up to one order of magnitude faster than its counterpart $\mathcal{M}^r$, and $\mathcal{M}_c^{dp}$ is up to two orders of magnitude faster than its counterpart $\mathcal{M}_c^r$.
2. $\mathcal{M}_c^r$ is consistently slower then $\mathcal{M}_H^r$. This is because, despite the fact that the two post-processing steps have the same number of variables, the $\mathcal{M}_c$ post-processing step has many additional constraints of type (C3).
3. The runtime of the DP-based mechanisms decreases as the privacy budget increases, due to the sizes of the cost tables that depend on the noise variance.[2]
4. The cumulative version $\mathcal{M}_c^{dp}$ outperforms its $\mathcal{M}_H^{dp}$ counterpart. Once again, the reason is due to the domain sizes. In fact, due to reduced sensitivity, $\mathcal{M}_c^{dp}$ applies a smaller amount of noise than that required by $\mathcal{M}_H^{dp}$ to guarantee the same level of privacy and resulting in smaller domain sizes.

[2] The implementation uses $D_s^r = \{\tilde{n}_s^r - \delta \ldots \tilde{n}_s^r + \delta\} \cap \mathbb{Z}_+$, where $\delta = 3 \times \lceil 2\lambda^2 \rceil$, i.e., 3 times the variance associated with the double-geometrical distribution with parameter λ.

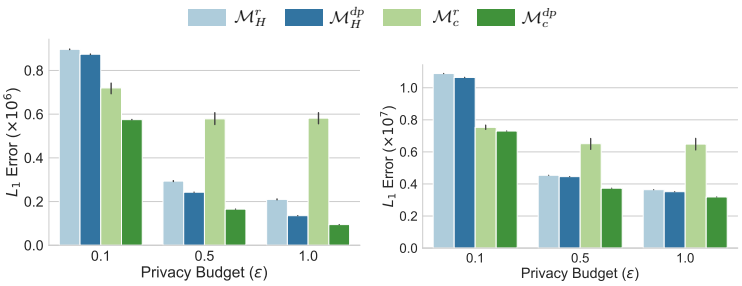


Fig. 10. The L_1 errors for the algorithms: Census data (left) and Taxi data (right).

9.3 Accuracy

Figure 10 reports the error induced by the mechanisms, i.e., the L_1-distance between the privacy-preserving and original datasets. The main observations can be summarized as follows:

1. The DP-based mechanisms produce more accurate results than their counterparts and $\mathcal{M}_{dp}^c$ dominates all other mechanisms.
2. As expected, the error of all mechanisms decreases as the privacy budget increases, since the noise decreases as privacy budget increases. The errors are larger in the NY Taxi dataset, which has a larger number of group sizes than the Census dataset.
3. Finally, the results show that the cumulative mechanisms tend to concentrate the errors on small group sizes. Unfortunately, these are also the most populated groups, and this is true for each subregion of the hierarchy. On the other hand, the DP-based version, that retains the integrality constraints, better redistributes the noise introduced by the geometrical mechanism and produce substantially more accurate results.

To shed further light on accuracy, Table 1 reports a breakdown of the average errors of each mechanism at each level of the hierarchies. Mechanism $\mathcal{M}_{dp}^c$ is clearly the most accurate. Note that the table reports the average number of *constraint violations* in the output datasets. A constraint violation is counted whenever a subtree of the hierarchy violates the PGRP *consistency* condition (2). Being exact, the DP-based methods report no violations. In contrast, both $\mathcal{M}_H^r$ and $\mathcal{M}_c^r$ report a substantial amount of constraint violations.

10 Related Work

The release of privacy-preserving datasets using differential privacy has been subject of extensive research [16,19,12]. However, these methods focus on creating *unattributed histograms* that count the number of individuals associated with each possible property in the dataset universe. Extensions to hierarchical problems were also explored. For instance, [15,21] study methods to answer count queries using a hierarchical structure to impose consistency of counts. Extensions for optimizing various count queries have also been proposed [5,18,21].

		Taxi Data				**Census Data**			
		L_1 Errors ($\times 10^4$)			#CV	L_1 Errors ($\times 10^3$)			#CV
ϵ	**Alg**	Lev 1	Lev 2	Lev 3		Lev 1	Lev 2	Lev 3	
0.1	$\mathcal{M}_H^r$	25.4	158.7	904.4	18206	40.3	54.3	802.1	1966
	$\mathcal{M}_H^{dp}$	26.6	121.9	915.7	**0**	10.3	38.4	825.4	**0**
	$\mathcal{M}_c^r$	47.9	153.2	**551.6**	19460	23.1	64.5	632.2	1715
	$\mathcal{M}_c^{dp}$	**19.9**	**65.6**	644.3	**0**	**0.9**	**23.2**	**550.6**	**0**
0.5	$\mathcal{M}_H^r$	8.6	81.2	364.2	18591	39.4	37.9	216.3	1990
	$\mathcal{M}_H^{dp}$	5.5	31.0	408.9	**0**	2.4	9.4	230.8	**0**
	$\mathcal{M}_c^r$	46.7	153.5	450.7	19531	23.1	61.0	494.2	1718
	$\mathcal{M}_c^{dp}$	**4.0**	**16.4**	**352.9**	**0**	**0.2**	**5.8**	**159.1**	**0**
1.0	$\mathcal{M}_H^r$	7.7	77.2	**279.0**	18085	40.7	39.2	130.0	1989
	$\mathcal{M}_H^{dp}$	3.1	19.8	328.5	**0**	1.2	5.1	128.8	**0**
	$\mathcal{M}_c^r$	47.1	154.2	447.1	19706	24.1	63.0	494.5	1728
	$\mathcal{M}_c^{dp}$	**2.0**	**8.7**	307.8	**0**	**0.1**	**3.2**	**91.0**	**0**

Table 1. L_1-errors and constraint violations (CV) for each level of the hierarchies.

These methods differ in two ways from the mechanisms proposed here: (1) They focus on histograms queries, rather than group queries; the latter generally have higher L_1-sensitivity and thus require more noise and (2) they ensure neither the consistency for integral counts nor the non-negativity of the release counts. They thus violate the requirements of group sizes (see Section 4).

Finally, [10] proposed a hierarchical-based solution based on minimizing the L2-distance between the noisy counts and their private counterparts. While this solution guarantees non-negativity of the counts, their mechanism, if formulated as a MIP/QIP, cannot cope with the scale of the census problems discussed here which compute privacy-preserving country-wise group sizes. If their solution is used as is, in its relaxed form, then it cannot guarantee the integrality of the counts. These mechanisms reduce to $\mathcal{M}_H^r$, which has been shown, in the previous section, to be strongly dominated by the DP-based mechanisms.

11 Conclusions

The release of datasets containing sensitive information concerning a large number of individuals is central to a number of statistical analysis and machine learning tasks. Of particular interest are hierarchical datasets, in which counts of individuals satisfying a given property need to be released at different granularities (e.g., the location of a household at a national, state, and county levels). The paper defined the *Privacy-perserving Group Release* (PGRP) problem and proposed an exact and efficient constrained-based approach to privately generate consistent counts across all levels of the hierarchy. This novel approach was evaluated on large, real datasets and results in speedups of up to two orders of magnitude, as well as significant improvements in terms of accuracy with respect to state-of-the-art techniques. Interesting avenues of future directions include exploiting different forms of parallelism to speed up the computations of the dynamic programming-based mechanisms even further, using, for instance, Graphical Processing Units as proposed in [11].

References

1. AAAS: New Privacy Protections Highlight the Value of Science Behind the 2020 census. https://www.aaas.org/news/new-privacy-protections-highlight-value-science-behind-2020-census, accessed: 2019-23-04
2. NBC News: Potential privacy lapse found in Americans' 2010 census data. https://www.nbcnews.com/news/us-news/potential-privacy-lapse-found-americans-2010-census-data-n972471, accessed: 2019-23-04
3. New York City Taxi Data. http://www.nyc.gov/html/tlc/html/about/trip_record_data.shtml, accessed: 2019-20-04
4. NY Times: To Reduce Privacy Risks, the Census Plans to Report Less Accurate Data. https://www.nytimes.com/2018/12/05/upshot/to-reduce-privacy-risks-the-census-plans-to-report-less-accurate-data.html
5. Cormode, G., Procopiuc, C., Srivastava, D., Shen, E., Yu, T.: Differentially private spatial decompositions. In: 2012 IEEE 28th International Conference on Data Engineering. pp. 20–31. IEEE (2012)
6. Dinur, I., Nissim, K.: Revealing information while preserving privacy. In: Proceedings of the twenty-second ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems. pp. 202–210. ACM (2003)
7. Dwork, C., McSherry, F., Nissim, K., Smith, A.: Calibrating noise to sensitivity in private data analysis. In: Theory of cryptography conference. pp. 265–284. Springer (2006)
8. Dwork, C., Roth, A.: The algorithmic foundations of differential privacy. Theoretical Computer Science **9**(3-4), 211–407 (2013)
9. Erlingsson, Ú., Pihur, V., Korolova, A.: Rappor: Randomized aggregatable privacy-preserving ordinal response. In: Proceedings of the 2014 ACM SIGSAC conference on computer and communications security. pp. 1054–1067. ACM (2014)
10. Fioretto, F., Lee, C., Van Hentenryck, P.: Constrained-based differential privacy for private mobility. In: Proceedings of the International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS). pp. 1405–1413 (2018)
11. Fioretto, F., Pontelli, E., Yeoh, W., Dechter, R.: Accelerating exact and approximate inference for (distributed) discrete optimization with gpus. Constraints **23**(1), 1–43 (2018)
12. Fioretto, F., Van Hentenryck, P.: Constrained-based differential privacy: Releasing optimal power flow benchmarks privately. In: Proceedings of the International Conference on the Integration of Constraint Programming, Artificial Intelligence, and Operations Research (CPAIOR). pp. 215–231 (2018)
13. Ghosh, A., Roughgarden, T., Sundararajan, M.: Universally utility-maximizing privacy mechanisms. SIAM Journal on Computing **41**(6), 1673–1693 (2012)
14. Golle, P.: Revisiting the uniqueness of simple demographics in the us population. In: Proceedings of the 5th ACM workshop on Privacy in electronic society. pp. 77–80. ACM (2006)
15. Hay, M., Rastogi, V., Miklau, G., Suciu, D.: Boosting the accuracy of differentially private histograms through consistency. Proceedings of the VLDB Endowment **3**(1-2), 1021–1032 (2010)
16. Huang, D., Han, S., Li, X., Yu, P.S.: Orthogonal mechanism for answering batch queries with differential privacy. In: Proceedings of the 27th International Conference on Scientific and Statistical Database Management. p. 24. ACM (2015)
17. Kuo, Y.H., Chiu, C.C., Kifer, D., Hay, M., Machanavajjhala, A.: Differentially private hierarchical group size estimation. arXiv preprint arXiv:1804.00370 (2018)

18. Li, C., Hay, M., Miklau, G., Wang, Y.: A data-and workload-aware algorithm for range queries under differential privacy. Proceedings of the VLDB Endowment **7**(5), 341–352 (2014)
19. Li, T., Li, N.: On the tradeoff between privacy and utility in data publishing. In: Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining. pp. 517–526. ACM (2009)
20. McSherry, F., Talwar, K.: Mechanism design via differential privacy. In: Foundations of Computer Science, 2007. FOCS'07. 48th Annual IEEE Symposium on. pp. 94–103. IEEE (2007)
21. Qardaji, W., Yang, W., Li, N.: Understanding hierarchical methods for differentially private histograms. Proceedings of the VLDB Endowment **6**(14), 1954–1965 (2013)
22. Sweeney, L.: k-anonymity: A model for protecting privacy. International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems **10**(05), 557–570 (2002)
23. Team, A.D.P.: Learning with privacy at scale. Apple Machine Learning Journal **1**(8) (2017)
24. U.S. Census Bureau: 2010 census summary file 1: census of population and housing, technical documentation. https: //www.census.gov/prod/cen2010/doc/sf1.pdf (2012)
25. Winkler, W.: Single ranking micro-aggregation and re-identification. Statistical Research Division report RR **8**, 2002 (2002)