Understanding Hierarchical Methods for Differentially Private Histograms

Wahbeh Qardaji, Weining Yang, Ninghui Li
Purdue University
305 N. University Street,
West Lafayette, IN 47907, USA
{wqardaji, yang469, ninghui}@cs.purdue.edu

ABSTRACT

In recent years, many approaches to differentially privately publish histograms have been proposed. Several approaches rely on constructing tree structures in order to decrease the error when answer large range queries. In this paper, we examine the factors affecting the accuracy of hierarchical approaches by studying the mean squared error (MSE) when answering range queries. We start with one-dimensional histograms, and analyze how the MSE changes with different branching factors, after employing constrained inference, and with different methods to allocate the privacy budget among hierarchy levels. Our analysis and experimental results show that combining the choice of a good branching factor with constrained inference outperform the current state of the art. Finally, we extend our analysis to multi-dimensional histograms. We show that the benefits from employing hierarchical methods beyond a single dimension are significantly diminished, and when there are 3 or more dimensions, it is almost always better to use the Flat method instead of a hierarchy.

1. INTRODUCTION

A histogram is an important tool for summarizing data. In recent years, significant improvements have been made in the problem of publishing histograms while satisfying differentially privacy. The utility goal is to ensure that range queries over the private histogram can be answered as accurately as possible. The naive approach, which we call the flat method, is to issue a count query for each unit bin in the histogram, and answer these queries using the Laplacian mechanism introduced by Dwork et al. [8]. Because each such query has sensitivity 1, the magnitude of the added noise is small. While this works well with histograms that have a small number of unit bins, the error due to this method increases substantially as the number of the bins increases. In particular, when answering range queries using the private histogram, the mean squared error increases linearly in the size of the query range.

Proceedings of the VLDB Endowment, Vol. 6, No. 14

Hay et al. [14] introduced the hierarchical method for optimizing differentially private histograms. In this approach, in addition to asking for counts of unit-length intervals, one also asks for counts of larger intervals. Conceptually, one can arrange all queried intervals into a tree, where the unit-length intervals are the leaves. The benefit of this method is that a range query which includes many unit bins can be answered using a small number of sub-intervals which exactly cover the query range. The tradeoff over the flat method is that when queries are issued at multiple levels, each query must satisfy differential privacy for a smaller ϵ. We refer to this as the privacy budget allocated to (or consumed by) the query. Hay et al. [14] also introduced novel constrained inference techniques to improve over the basic hierarchical method, which exploits the observation that query results at different levels should satisfy certain consistency relationships to obtain improved estimates. This hierarchical method with constrained inference has been adopted by other researchers [5, 4].

A number of other methods have also been developed and applied to the histogram program. For example, Xiao et al. [21] introduced the Privlet method, which decomposes the original histogram using the Haar wavelet, then adds noise to the decomposed coefficients, and finally reconstructs the histogram using the noisy coefficients. The benefit of this method is that, when answering range queries, noise added to the coefficients can be partially cancelled because of the nature of Haar wavelet processing. Li et al. [15] introduced the matrix mechanism, which tries to optimize answers for a fixed set of counting queries. As solving the optimization problem is computationally infeasible, several approximation techniques have been developed, such as the eigen-select algorithm [16] and the low-rank approximation algorithm [24].

We observe that hierarchical methods can be parameterized in several different ways, and different parameter values may significantly affect the accuracy of the results. Such parameters include the branching factor for the hierarchy, as well as the division of privacy budget among its levels. Furthermore, several other parameters also affect the relative strength of each method, including the total number of histogram bins, the number of dimensions, and the distribution of the queries. To thoroughly understand the strengths and weaknesses of these methods, we need to understand how these parameters affect the accuracy.

In this paper, we combine combinatorial analysis of how the different parameters affect the accuracy of each method, with experimental evaluation that validates the theoretical

analysis. We believe this two-pronged approach will promote a deeper understanding of the effectiveness of the hierarchical methods. For utility, we consider the Mean Squared Error (MSE) of answering all range queries over the data domain, with every query considered equally likely.

We make the following contributions. First, we perform a thorough analysis of hierarchical methods in one dimension. We analyze how different branching factors affect the MSE, and show that choosing a larger branching factor (e.g., 16) instead of 2, one can reduce MSE by a factor of more than 4. We also analyze the benefits due to using constrained inference, the effect of which depends on the branching factor. We also compare several privacy budget allocation schemes, and experimentally show that, when one uses optimal branching factor with constrained inference, equal privacy budget allocation suffices to give excellent result.

We then compare our method of combining a larger branching factor with constraint inference to other state of the art methods, including the Wavelet method and two instantiations of the Matrix Mechanism. We provide comparison results both from analytical computations and from experiments with real-world datasets. We show that our method outperforms other state of art methods.

Finally, we extend our analysis to multiple dimensions. We show that for higher-dimensional datasets, the benefit of using hierarchies is significantly limited. In particular, when the number of dimensions is 3 or higher, for all but the largest datasets, one would be better of with using the naive flat method.

The rest of the paper is organized as follows. We given the problem definition in Section 2, and analyze one-dimensional histogram in Section 3. We then compare our proposed method with the current state of the art in Section 4, and extend our analysis to multiple dimensions in Section 5. Finally, we discuss related work in Section 6, and conclude in Section 7.

2. PROBLEM DEFINITION

We begin by formally stating the privacy definition we employ, as well as our problem definition.

2.1 Differential Privacy

Differential privacy, which was developed in a series of papers [7, 10, 2, 9, 8], has been widely accepted in the research community as the notion of choice for private data analysis.

DEFINITION 1 (ϵ-DIFFERENTIAL PRIVACY [8, 9]). *A randomized mechanism $\mathcal{A}$ gives ϵ-differential privacy if for any pair of neighboring datasets D and D', and any $S \in Range(\mathcal{A})$,*

$$\mathsf{Pr}\left[\mathcal{A}(D) = S\right] \leq e^{\epsilon} \cdot \mathsf{Pr}\left[\mathcal{A}(D') = S\right].$$

In this paper we consider two datasets D and D' to be neighbors if one dataset equals the other dataset with one additional tuple added. We use $D \simeq D'$ to denote this.

Differential privacy is composable in the sense that combining multiple mechanisms that satisfy differential privacy for $\epsilon_1, \cdots, \epsilon_m$ results in a mechanism that satisfies ϵ-differential privacy for $\epsilon = \sum_i \epsilon_i$. Because of this, we refer to ϵ as the privacy budget of a privacy-preserving data analysis task. When a task involves multiple steps, the budget is divided for different steps to use.

One approach to evaluate a function g on a dataset D while satisfying ϵ-differential privacy is to add to the output random noise whose magnitude is proportional to GS_g, where GS_g is the *global sensitivity* or the L_1 sensitivity of g. We use $\mathcal{A}_g$ to denote this mechanism:

$$\begin{aligned} \mathcal{A}_g(D) &= g(D) + \mathsf{Lap}\left(\tfrac{\mathsf{GS}_g}{\epsilon}\right) \\ \text{where} \quad \mathsf{GS}_g &= \max_{D \simeq D'} |g(D) - g(D')|, \\ \text{and} \quad \mathsf{Pr}\left[\mathsf{Lap}\left(\beta\right) = x\right] &= \tfrac{1}{2\beta} e^{-|x|/\beta} \end{aligned}$$

In the above, $\mathsf{Lap}\left(\beta\right)$ represents a random variable sampled from the Laplace distribution with scale parameter β. This is generally referred to as the *Laplacian mechanism* for satisfying differential privacy.

2.2 Problem Definition

We consider a dataset being a set of points in a d-dimensional domain. For simplicity, we assume that each dimension has been quantized into n consecutive and non-overlapping bins; thus the overall domain has $N = n^d$ bins. We consider a hierarchy over a d-dimensional dataset that partitions the data domain into sets of b^d hypercubes. Hence we call b branching factor. Each node in the hierarchy counts the number of tuples that fall in the hypercube the node represents. The leaf nodes in this hierarchy correspond to the actual histogram bins, while the root node corresponds to the entire data domain.

In order to measure utility, we examine the accuracy of answering range queries over the dataset using the differentially private histogram. We assume that a range query, r, represents a hyperrectangle in the d-dimensional domain specified by the dataset, and asks for the number of tuples that fall within the bins that are completely included in the area covered by the hyperrectangle. We consider the behaviors of different histogram publishing methods over a dataset D. In the descriptions below, we leave the dataset D implicit.

For a query r, we use $A(r)$ to denote the correct answer to r. For a method $\mathcal{M}$ and a query r, we use $Q_{\mathcal{M}}(r)$ to denote the answer to the query r when using the histogram constructed by method $\mathcal{M}$ to answer the query r, and $E_{\mathcal{M}}(r)$ to denote the absolute error for query r under $\mathcal{M}$. That is

$$E_{\mathcal{M}}(r) = |Q_{\mathcal{M}}(r) - A(r)|.$$

Because the method $\mathcal{M}$ is often randomized, $E_{\mathcal{M}}(r)$ is a random variable. We use the variance of $E_{\mathcal{M}}(r)$ as a measurement of the error. We note that this is also the mean squared error (MSE) when viewing $Q_{\mathcal{M}}(r)$ as an estimator for $A(r)$. We assume that the user is interested in any set of range query, and that all the queries are equally likely to be chosen. Hence, we consider the average MSE (which we, with a slight abuse of terminology, still call the MSE), over all possible range queries. As this is also the average error variance, we use $\mathsf{V}_{\mathsf{Avg}}[\mathcal{M}]$ to denote this. Let Q be the set of all queries over some histogram, then

$$\mathsf{V}_{\mathsf{Avg}}[\mathcal{M}] = \frac{\sum_{r \in Q} (E_{\mathcal{M}}(r))^2}{|Q|}$$

We study the efficacy of various hierarchical approaches that publish a histogram while satisfying ϵ-DP. More specifically, we study various factors which affect the MSE when answering all range queries:

1. The histogram size N, which is the number of bins in the histogram.

2. The branching factor, b. In most existing work, a binary hierarchy is used. This results in a deep tree, and thus requires substantial division of privacy budget. We want to analyze how the branching factor affects the MSE and how to choose the optimal branching factor for a particular histogram.

3. The use of constrained inference [14], which is the process of utilizing the inherent redundancy in hierarchical queries in order to boost accuracy and ensure inter-level consistency. We aim to analyze its effect in order to estimate its benefit for various hierarchies.

4. The way privacy budget is allocated across multiple levels in the hierarchy. Two methods have been proposed in the literature. One is to divide the privacy budget equally among all levels. In [5], geometric budget allocation has been proposed. We also consider an approach that search for an optimal allocation.

5. Histogram dimension. While most datasets in practice are multi-dimensional, differentially private multi-dimensional histograms has received little attention in the literature. We aim to understand how the accuracy of hierarchical methods is affected by dimensionality.

3. THE ONE DIMENSIONAL CASE

In this section, we focus on histograms over one-dimensional datasets.

3.1 The Flat Method: F

In the flat method, which we denote by $\mathbf{F}$, one issues a count query for each unit-length interval. This is answered by adding noise following the Laplace distribution $\mathsf{Lap}\left(\frac{1}{\epsilon}\right)$. The variance at each bin is therefore $\mathsf{V_u} = \frac{2}{\epsilon^2}$. Let $|r|$ denote the number of unit-length intervals in a range r. Then the MSE is $\mathsf{Var}(E_{\mathcal{M}}(r)) = |r| \cdot \mathsf{V_u}$.

We note that the average length of all queries over N unit intervals is $\frac{\sum_{j=1}^{N} j(N-j+1)}{N(N+1)/2} = \frac{(N+2)}{3}$. Hence the average-case MSE is $\frac{(N+2)}{3}\mathsf{V_u}$. This is linear in the number of bins.

3.2 Hierarchical Methods: $\mathbf{H_2}$ and $\mathbf{H_b}$

In the hierarchical method $\mathbf{H_b}$, one arranges intervals into a tree with branching factor b, where the unit-length intervals are the leaves, and each node in the tree corresponds to an interval that is the union of the intervals of its children. We use $\mathbf{H_2}$ to denote the case of a binary tree.

Let $h = \lceil \log_b N \rceil$. Then the tree has $h+1$ levels. We say that the leaf nodes are at level 1, and the root node are thus at level $h+1$. A query is issued at each node in the tree except for the root; thus queries are issued at levels from 1 to h. Each level is allocated privacy budget $\frac{\epsilon}{h}$. When a range query comes, one finds the least number of nodes that correspond to the query range and sum up the noisy counts in the leaves.

The branching factor b affects the accuracy in two ways. Increasing b reduces the height of the tree, thereby increasing the privacy budget available to each query, and improving the accuracy. At the same time, increasing b requires more nodes to be used to answer a query on average. To choose the optimal branching factor b, one needs to balance these two effects. The following proposition gives the formula for the MSE.

PROPOSITION 1. *The method* $\mathbf{H_b}$ *has the following average-case MSE:*

$$\mathsf{V_{Avg}}[\mathbf{H_b}] = \frac{N}{N+1}\left((b-1)h^3 - \frac{2(b+1)h^2}{3} + O\left(\frac{bh}{N}\right)\right)\mathsf{V_u} \quad (1)$$

PROOF. Let $left(v)/right(v)$ be the index of the left-most/rightmost leaf node in the subtree rooted at node v, and $p(v)$ be the parent node of v. Then the number of queries that use the node v is

$$left(v) \cdot (N - right(v) + 1) - left(p(v)) \cdot (N - right(p(v)) + 1)$$

To answer all possible range queries, the total number of times nodes at level ℓ are used is:

$$M(\ell) = \sum_{j=1}^{N/b^\ell} \sum_{k=1}^{b} \left((j-1)b^\ell + (k-1)b^{\ell-1} + 1\right)\left(N - (j-1)b^\ell - kb^{\ell-1} + 1\right) - \left((j-1)b^\ell + 1\right)\left(N - jb^\ell + 1\right)$$

where j denotes the index of the parent nodes, and k iterates over node j's children. Expanding the above nested summations, we have

$$M(\ell) = (b-1)\left(\frac{N^2}{2} - \frac{N(b+1)b^{\ell-1}}{3} + N\right) \quad (2)$$

To answer all possible range queries, the number of nodes in a complete tree that are used is:

$$\sum_{\ell=1}^{h} M(\ell) = \left(\frac{(b-1)\,h\,N^2}{2} - \frac{(b+1)N^2}{3} + (b-1)Nh + \frac{(b+1)N}{3}\right)$$

Each node has variance $h^2\mathsf{V_U}$, and there are $N(N+1)/2$ queries in total. Therefore, we have $\mathsf{V_{Avg}}[\mathbf{H_b}] = \frac{\sum_{\ell=1}^{h} M(\ell)}{N(N+1)/2} h^2 \mathsf{V_U}$, proving the proposition. □

Note that as the MSE is poly-logarithmical in N instead of linear in N, when N is large, the hierarchical method results in better accuracy than the flat method.

By dropping the $\frac{N}{N+1}$ factor and the $O(\frac{1}{N})$ term in Equation (1), we obtain the following approximation of $\mathsf{V_{Avg}}[\mathbf{H_b}]$, which we use $\mathsf{V^*_{Avg}}[\mathbf{H_b}]$ to denote:

$$\mathsf{V^*_{Avg}}[\mathbf{H_b}] = \left((b-1)h^3 - \frac{2(b+1)h^2}{3}\right) \cdot \mathsf{V_u} \quad (3)$$

We now give a high-level argument for the above approximation. On each level, the left end of the query may include $0, 1, \cdots, b-1$ nodes, which averages to $(b-1)/2$; similarly the right end of the query includes on average $(b-1)/2$. Thus on average a query needs to be answered by $(b-1)h$ nodes, where each node is answered with privacy budget $\frac{\epsilon}{h}$, and hence error variance of $h^2\mathsf{V_u}$. This explains the first and most significant term in Equation (3). The second term is due to the fact that some queries may not need all levels to answer it. For the binary case, we have

$$\mathsf{V^*_{Avg}}[\mathbf{H_2}] = \left(h^3 - 2h^2\right) \cdot \mathsf{V_u} \quad (4)$$

Optimal Branching Factor. The optimal branching factor b should minimize Equation (3). Because $h = \lceil \log_b N \rceil$

is not a continuous function, we cannot find a closed form solution for the value b. However, given any N, it is straightforward to computationally find the b value that minimizes Equation (3) by trying different b values.

Another use of Proposition 1 is to get a sense of what is the optimal branching factor for very large N values. When N is very large, we can use $\log_b N$ to approximate $\lceil \log_b N \rceil$. Trying to minimizing $(b-1)(\log_b N)^3 - \frac{2}{3}(b+1)(\log_b N)^2$, however, results in b dependent on N. When N is very large, the first term $(b-1)(\log_b N)^3$ dominates the second term, and we can choose b such that the first term's derivative, $\frac{-3b+b\log b+3}{b\log^3 b}\log^3 N$, is 0. This results in $b \approx 16.8$. When setting $b = 16$, this results in MSE reduction from $\mathbf{H_2}$ by a factor of approximately $\frac{(\log_2 16)^3}{(16-1)} \approx 4.27$.

3.3 Constrained Inference - $\mathbf{H_b^c}$

Hay et al. [14] introduced constrained inference techniques to improve $\mathbf{H_2}$. This technique also applies to arbitrary branching factors. We thus present the general version of the technique. We dissect constrained inference into two steps: Weighted Averaging, and Mean Consistency.

Weighted averaging. The weighted averaging step is based on the following fact:

FACT 2. *Given two random variables X_1 and X_2, consider $X = \alpha X_1 + (1-\alpha)X_2$, the variance of X is minimized when $\alpha = \frac{\mathsf{Var}(X_2)}{\mathsf{Var}(X_1)+\mathsf{Var}(X_2)}$ and the minimal variance of X is $\frac{\mathsf{Var}(X_1)\mathsf{Var}(X_2)}{\mathsf{Var}(X_1)+\mathsf{Var}(X_2)}$.*

Let $n[v]$ be the noisy count for the node v which is at level i in the tree. We use the weighted average of its original noisy count and the sum of its children's count to update the node's noisy count. We use $z_i[v]$ to denote the updated count.

$$z_i[v] = \begin{cases} n[v], \text{if } i = 1, \text{ i.e., } v \text{ is a leaf node} \\ \frac{b^i - b^{i-1}}{b^i - 1} n[v] + \frac{b^{i-1}-1}{b^i-1} \sum_{u \in child(v)} z_{i-1}[u], \text{ow.} \end{cases}$$

Mean consistency. The mean consistency step aims at ensuring that for each node, its children values sum up to be the same as the parent. This is done by calculating the difference in the sum of the values of the children, and the value of the parent. This difference is divided equally among the children, as follows, where u is the parent of v.

$$\bar{n}_i[v] = \begin{cases} z_i[v], \text{if } i = h \\ z_i[v] + \frac{1}{b}\left(\bar{n}_{i+1}[u] - \sum_{v \in child(u)} z_i[v]\right), \text{ow.} \end{cases}$$

After constrained inference, each node's value is a weighted sum of the original noisy counts of all nodes in the hierarchy. With all nodes along the path from the leaf to its highest ancestor that has a noisy count has positive weights, and all other nodes under that ancestor have negative weights. Figure 1 gives an example showing the effect of constrained inference.

Effect of constrained inference. In the following, we try to quantify the error reduction effect of constrained inference, and to develop a way to choose the optimal branching factor when constrained inference is used. In addition to $\mathsf{V_U}$, which is the variance of each unit bin in the flat method, we use the following notations:

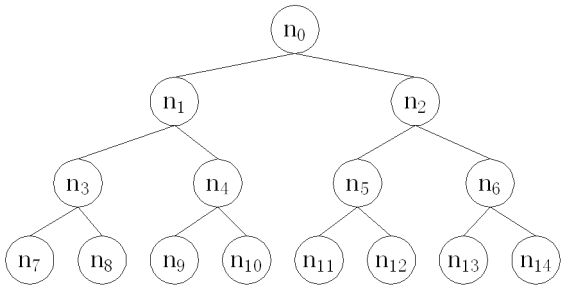


Figure 1: Effect of constrained inference. This figure demonstrates a hierarchical tree for N = 8, b = 2. We use n_i to denote the original noisy count for node i, and n_i' to denote the count after constrained inference. Then we have
$n_7' = \frac{13}{21}n_7 + \frac{5}{21}n_3 + \frac{1}{7}n_1 - \frac{2}{21}n_4 - \frac{1}{21}(n_9+n_{10}) - \frac{8}{21}n_8$,
$n_8' = \frac{13}{21}n_8 + \frac{5}{21}n_3 + \frac{1}{7}n_1 - \frac{2}{21}n_4 - \frac{1}{21}(n_9+n_{10}) - \frac{8}{21}n_7$,
$n_9' = \frac{13}{21}n_9 + \frac{5}{21}n_4 + \frac{1}{7}n_1 - \frac{2}{21}n_3 - \frac{1}{21}(n_7+n_8) - \frac{8}{21}n_{10}$,
$q = \frac{3}{7}n_1 + \frac{8}{21}n_3 + + \frac{1}{21}n_4 + \frac{4}{21}(n_7+n_8) + \frac{11}{21}n_9 - \frac{10}{21}n_{10}$,
where $q = n_7' + n_8' + n_9'$ is the answer to the range query includes the first 3 leaf nodes; we have $\mathsf{Var}[q] = \frac{399}{441}\mathsf{V_O}$, which is significantly than the $2\mathsf{V_O}$ needed before constrained inference (using $n_3 + n_9$), where $\mathsf{V_O}$ is the variance of the n_i's.

- $\mathsf{V_O}$ is the variance of each node in the hierarchy without constrained inference. Thus $\mathsf{V_O} = h^2\mathsf{V_U}$, where h is the height of the hierarchy.
- $V_\ell^\downarrow$ is the variance of the best estimate of a node at level ℓ "from below"; that is, the best estimate using only information from the node and its descendants. This is also the variance of the node after the weighted averaging step.
- $V_\ell^\uparrow$ is the variance of the best estimate of a node at level ℓ "from above"; that is, the best estimate using information from the hierarchy excluding its descendants.

The following Proposition shows the effect of the weighted averaging step.

PROPOSITION 3. *The weighted averaging step reduces the error variance of level ℓ to be $V_\ell^\downarrow = \frac{b^{\ell-1}}{\sum_{i=0}^{\ell-1} b^i}\mathsf{V_O}$. We also have $V_\ell^\downarrow > \frac{b-1}{b}\mathsf{V_O}$ for all levels, and when $\ell > 1$, we have $V_\ell^\downarrow \le \frac{b}{b+1}\mathsf{V_O}$.*

PROOF. We prove by induction. The leaf level (i.e., $\ell = 1$) obtains no benefit from the weighted averaging step, and their error variance equals $\mathsf{V_O} = \frac{b^0}{\sum_{i=0}^{0} b^i}\mathsf{V_O}$. Assume that $V_\ell^\downarrow = \frac{b^{\ell-1}}{\sum_{i=0}^{\ell-1} b^i}\mathsf{V_O}$, consider $V_{\ell+1}^\downarrow$. Summing up its children results in an estimate with variance $\frac{b \cdot b^{\ell-1}}{\sum_{i=0}^{\ell-1} b^i}\mathsf{V_O}$, combining this with the original noisy count of variance $\mathsf{V_O}$, the resulting variance is

$$V_{\ell+1}^\downarrow = \frac{\mathsf{V_O} \cdot \frac{b^\ell \mathsf{V_O}}{\sum_{i=0}^{\ell-1} b^i}}{\mathsf{V_O} + \frac{b^\ell \mathsf{V_O}}{\sum_{i=0}^{\ell-1} b^i}} = \frac{b^\ell}{\sum_{i=0}^{\ell} b^i}\mathsf{V_O}.$$

Note that forall $\ell \ge 1$, we have $\frac{b^{\ell-1}}{\sum_{i=0}^{\ell-1} b^i} = \frac{b^{\ell-1}(b-1)}{b^\ell - 1} > \frac{b^{\ell-1}(b-1)}{b^\ell} = \frac{(b-1)}{b}$. Also when $\ell > 1$, we have $\frac{b^{\ell-1}}{\sum_{i=0}^{\ell-1} b^i} \le \frac{b^{\ell-1}}{b^{\ell-1}+b^{\ell-2}} = \frac{b}{b+1}$. □

When applied to $\mathbf{H_2}$, this means that the variances of the nodes are reduced from 1 to (starting from leaf level going up) $1, \frac{2}{3}, \frac{4}{7}, \frac{8}{15}, \cdots$. For larger b values, this reduction effect is small, as $\frac{b-1}{b}$ is close to 1.

After the Mean Consistency step, the error variances for all non-root nodes are further reduced, including those of the leaf nodes, which do not benefit from weighted averaging. The following proposition gives a bound of this effect.

PROPOSITION 4. *After constrained inference, the error variance of each node is at least* $\frac{b-1}{b+1}\mathsf{V_O}$.

PROOF. For each node v at level ℓ, we can partition the hierarchy into two disjoint parts. One part consists of all the descendants of v, and the other consists of the rest of the hierarchy, including v. The value of v after constrained inference can be viewed as the average of the best estimates of its count obtained from the two parts.

The best estimate obtained from the descendants is the sum of the values of all of v's children after weighted averaging step, and has variance $bV^{\downarrow}_{\ell-1}$. From Proposition 3, we have $bV^{\downarrow}_{\ell-1} > b\frac{b-1}{b}\mathsf{V_O} = (b-1)\mathsf{V_O}$.

The best estimate from the rest has variance $V^{\uparrow}_{\ell}$. We now show that $V^{\uparrow}_{\ell} > \frac{b-1}{b}\mathsf{V_O}$ for all ℓ's. This estimate is the weighted average of the noisy count at the node, which has variance $\mathsf{V_O}$, and the noisy count obtained by subtracting the sum of its siblings from the the best estimate of its parent without using any information from its descendants, which has variance $V^{\uparrow}_{\ell+1} + (b-1)V^{\downarrow}_{\ell}$. We thus have $V^{\uparrow}_{\ell} = \frac{\mathsf{V_O}(V^{\uparrow}_{\ell+1}+(b-1)V^{\downarrow}_{\ell})}{\mathsf{V_O}+V^{\uparrow}_{\ell+1}+(b-1)V^{\downarrow}_{\ell}}$. We use induction to show that $V^{\uparrow}_{\ell} > \frac{b-1}{b}\mathsf{V_O}$. When $\ell = h$, we have $V^{\uparrow}_{h} = \mathsf{V_O} > \frac{b-1}{b}\mathsf{V_O}$. Assume that $V^{\uparrow}_{\ell+1} > \frac{b-1}{b}\mathsf{V_O}$, we have

$$V^{\uparrow}_{\ell} = \frac{\mathsf{V_O}(V^{\uparrow}_{\ell+1} + (b-1)V^{\downarrow}_{\ell})}{\mathsf{V_O} + V^{\uparrow}_{\ell+1} + (b-1)V^{\downarrow}_{\ell}} > \frac{\frac{b-1}{b} + \frac{(b-1)^2}{b}}{1 + \frac{b-1}{b} + \frac{(b-1)^2}{b}}\mathsf{V_O} = \frac{b-1}{b}\mathsf{V_O}.$$

The variance of a node at level j after constrained inference is thus $\frac{bV^{\downarrow}_{j-1}V^{\uparrow}_{j}}{bV^{\downarrow}_{j-1}+V^{\uparrow}_{j}} > \frac{(b-1)^2/b}{b-1+\frac{b-1}{b}}\mathsf{V_O} = \frac{b-1}{b+1}\mathsf{V_O}$. □

We have observed in experiments that in a deep hierarchy, nodes in levels around the middle have variances approaching the $\frac{b-1}{b+1}\mathsf{V_O}$ lower bound established in Proposition 4. Nodes at the lowest and highest levels have variances closer to $\frac{b-1}{b}\mathsf{V_O}$.

From Proposition 4, one can observe that when b is large, each individual node's variance is reduced only slightly, as $\frac{b-1}{b+1}$ is close to 1. However, as constrained inference makes the noise at different nodes correlated, there is further reduction effect when a query includes multiple nodes. For example, when a query includes substantially more than half of a node's children, then the query's variance can be significantly reduced by using a weighted average of two answers: one obtained by summing up nodes included in the query, and the other obtained by subtracting from the parent node's value the sum of the sibling nodes that are not included in the query. Mean consistency achieves this effect. In fact, for large b values, this is the main source of error reduction from constrained inference. Below we analyze the effect when we consider only the interactions among two levels.

Consider b nodes at level ℓ under the same parent, it is equally likely $0, 1, \cdots, b-1$ of them are included in a query. We consider the best answer from below, using nodes included in the query, and the best answer from above. When k nodes are included, the estimate from below has variance $kV^{\downarrow}_{\ell}$, and the estimate from above has variance $V^{\uparrow}_{\ell+1} + (b-k)V^{\downarrow}_{\ell}$, and their minimal variance combination has variance

$$\mathsf{Var}(Z) = \frac{kV^{\downarrow}_{\ell}\left(V^{\uparrow}_{\ell+1} + (b-k)V^{\downarrow}_{\ell}\right)}{V^{\uparrow}_{\ell+1} + bV^{\downarrow}_{\ell}}$$

Thus on average level ℓ's contribution to the variance of a query

$$\frac{1}{b-1}\sum_{k=1}^{b-1}\mathsf{Var}(Z) = \frac{b(b-1)(b+1)V^{\downarrow 2}_{\ell}+3b(b-1)V^{\uparrow}_{\ell+1}V^{\downarrow}_{\ell}}{6(V^{\uparrow}_{\ell+1}+bV^{\downarrow}_{\ell})}$$

Without constrained inference, the average variance is $\frac{1}{b-1}\sum_{k=1}^{b-1} k\mathsf{V_O} = \frac{b}{2}\mathsf{V_O}$. Diving the above by $\frac{b}{2}\mathsf{V_O}$, and using $\frac{b-1}{b}\mathsf{V_O}$ as an approximate value for $V^{\downarrow}_{\ell}$ and $V^{\uparrow}_{\ell}$, and we get that the average variance is reduced by

$$\frac{(b^2+4b)(b-1)}{6(b+1)b} = \frac{(b-1)(b+4)}{3b(b+1)}$$

The above analysis can be applied to any two adjacent levels in a hierarchy to estimate how much noise reduction one obtains in the lower level. It gives an estimate of the benefit of constrained inference to be approximately 3 when b is large. This value overestimates the error reduction benefit for two reasons. First, it uses $\frac{b-1}{b}\mathsf{V_O}$ for $V^{\downarrow}_{\ell}$ and $V^{\uparrow}_{\ell}$, which is a lower-bound, especially for $V^{\downarrow}_{1}$ and $V^{\uparrow}_{h}$, both equals $\mathsf{V_O}$. Second, the top level does not benefit from the reduction effect. At the same time, this value also underestimates the benefit from constrained inference because the analysis does not consider cross-level error reduction effect. For example, when a query includes b^2-1 grandchildren of a level 3 node, one could obtain a lower-variance answer by subtracting from the level-3 node one leaf node. This effect is quite pronounced when $b = 2$. However, when b is large, this effect is small since only a small portion of the queries benefit from this cross-level error reduction.

Input: N, b, ϵ_*vector*
Output: $V^{\uparrow}$_*vector*, $V^{\downarrow}$_*vector*,
$h = \lceil \log_b N \rceil$;
for $i = 1$ *to* h **do**
  $\mathsf{V_O}_i = \frac{2}{\epsilon_i^2}$;
end
$V^{\downarrow}_1 = \mathsf{V_O}_1$;
for $i = 2$ *to* h **do**
  $V^{\downarrow}_i = \frac{b\mathsf{V_O}_i V^{\downarrow}_{i-1}}{\mathsf{V_O}_i + bV^{\downarrow}_{i-1}}$;
end
$V^{\uparrow}_h = \mathsf{V_O}_h$;
for $i = h-1$ *downto* 1 **do**
  $V^{\uparrow}_i = \frac{\mathsf{V_O}_i(V^{\uparrow}_{i+1}+(b-1)V^{\downarrow}_i)}{\mathsf{V_O}_i+V^{\uparrow}_{i+1}+(b-1)V^{\downarrow}_i}$;
end
return $V^{\uparrow}, V^{\downarrow},$;

Algorithm 1: Computing $V^{\uparrow}$, $V^{\downarrow}$

	$\mathbf{H_2}$		$\mathbf{H_b}$					$\mathbf{H_2^c}$		$\mathbf{H_b^c}$	
N	$\mathsf{V}^*_{\mathsf{Avg}}[\mathbf{H_2}]$	$\mathsf{V}^{\mathsf{ex}}_{\mathsf{Avg}}[\mathbf{H_2}]$	b	h	$\mathsf{V}^*_{\mathsf{Avg}}[\mathbf{H_b}]$	$\mathsf{V}^{\mathsf{ex}}_{\mathsf{Avg}}[\mathbf{H_b}]$	$\frac{\mathsf{V}^{\mathsf{ex}}_{\mathsf{Avg}}[\mathbf{H_2}]}{\mathsf{V}^{\mathsf{ex}}_{\mathsf{Avg}}[\mathbf{H_b}]}$	$\mathsf{V}^{\mathsf{ex}}_{\mathsf{Avg}}[\mathbf{H_2^c}]$	$\frac{\mathsf{V}^{\mathsf{ex}}_{\mathsf{Avg}}[\mathbf{H_2}]}{\mathsf{V}^{\mathsf{ex}}_{\mathsf{Avg}}[\mathbf{H_2^c}]}$	$\mathsf{V}^{\mathsf{ex}}_{\mathsf{Avg}}[\mathbf{H_b^c}]$	$\frac{\mathsf{V}^{\mathsf{ex}}_{\mathsf{Avg}}[\mathbf{H_b}]}{\mathsf{V}^{\mathsf{ex}}_{\mathsf{Avg}}[\mathbf{H_b^c}]}$
2^4	64.00	79.53	16	1	7.33	12.00	6.76	34.46	2.31	12.00	1
2^5	150.00	163.83	32	1	18.00	22.67	7.27	61.34	2.67	22.67	1
2^6	288.00	299.15	64	1	39.33	44.00	6.81	99.92	2.99	44.00	1
2^7	490.00	498.38	128	1	82.00	86.67	5.75	152.18	3.27	86.67	1
2^8	768.00	773.98	16	2	149.33	150.98	5.13	220.06	3.52	79.23	1.91
2^9	1134.00	1138.11	23	2	224.00	221.57	5.14	305.54	3.73	114.02	1.94
2^{10}	1600.00	1602.73 32	32	2	320.00	320.83	5.00	410.58^s	3.90	164.30^s	1.95
2^{11}	2178.00	2179.77	46	2	469.33	463.83	4.70	535.63^s	4.07	233.66^s	1.99
2^{12}	2880.00	2881.12	16	3	606.00	606.30	4.75				
2^{13}	3718.00	3718.70	21	3	816.00	795.53	4.67				
2^{14}	4704.00	4704.43	26	3	1026.00	1011.33	4.65				
2^{15}	5850.00	5850.26	32	3	1278.00	1278.08	4.58				
2^{16}	7168.00	7168.16	16	4	1557.33	1557.37	4.60				
2^{17}	8670.00	8670.09	20	4	1984.00	1932.04	4.49				
2^{18}	10368.00	10368.05	23	4	2304.00	2278.98	4.55				
2^{19}	12274.00	12274.03	27	4	2730.67	2719.40	4.51				
2^{20}	14400.00	14400.02	16	5	3183.33	3183.34	4.52				

Table 1: Comparing $\mathbf{H_2}$, $\mathbf{H_b}$, $\mathbf{H_2^c}$, and $\mathbf{H_b^c}$. $\mathsf{V}^*_{\mathsf{Avg}}$ denotes the estimated average error variance by Equation (3). $\mathsf{V}^{\mathsf{ex}}_{\mathsf{Avg}}$ denotes the actual average error variance by experience. $\epsilon = 1.0$, so $\mathsf{V}_{\mathsf{u}} = 2.0$

Input: N, b, ϵ_*vector*, $V^{\uparrow}$_*vector*, $V^{\downarrow}$_*vector*
Output: $\mathsf{V}^*_{\mathsf{Avg}}[\mathbf{H_b^c}]$

$h = \lceil \log_b N \rceil$;
for $i = 1$ *to* $h-1$ **do**
  $Z_i = \frac{b(b-1)(b+1)V_i^{\downarrow 2} + 3b(b-1)V_{i+1}^{\uparrow}V_i^{\downarrow}}{6(V_{i+1}^{\uparrow} + bV_i^{\downarrow})}$;
end
$Z_h = V_h^{\downarrow} / \frac{2}{\epsilon_h^2}$;
for $i = 1$ *to* h **do**
  $M_i = (b-1)\left(\frac{N^2}{2} - \frac{N(b+1)b^{i-1}}{3} + N\right)$;
end
$sum = 0$;
for $i = 1$ *to* h **do**
  $sum += M_i Z_i \mathsf{V}_{\mathsf{O}i}$;
end
return *sum*;

Algorithm 2: Computing $\mathsf{V}^*_{\mathsf{Avg}}[\mathbf{H_b^c}]$

In Algorithm 1, we present pseudocode for computing $V_\ell^{\uparrow}$ and $V_\ell^{\downarrow}$. In Algorithm 2, we present an algorithm for estimating the error variance when N and b are given. It corrects for the two sources of over-estimation biases, but does not account for the cross-level error reduction. Thus it may slightly underestimate the MSE for hierarchies of height 3 or more. Even with the inexactness, we believe that using 3 as the benefit factor for constrained inference provides a reasonable approximation when b and h are large.

3.4 Checking Analysis with Experiments

We now use experiments to verify that our analysis is correct and to provide the actual numbers of error reduction. Table 1 shows the result for $\mathbf{H_2}$, $\mathbf{H_b}$, $\mathbf{H_2^c}$, and $\mathbf{H_b^c}$, for different N values. The table includes two kinds of average error variance numbers. $\mathsf{V}^*_{\mathsf{Avg}}[\cdot]$ means estimated values computed using Equation (3). $\mathsf{V}^{\mathsf{ex}}_{\mathsf{Avg}}[\cdot]$ means numbers obtained from numerical computations. $\mathsf{V}^{\mathsf{ex}}_{\mathsf{Avg}}[\mathbf{H_2}]$ and $\mathsf{V}^{\mathsf{ex}}_{\mathsf{Avg}}[\mathbf{H_b}]$ are computed by summing up the nodes needed to answer all queries; the calculation is exact and takes into account that the tree may not be fully balanced.

For constrained inference, $\mathsf{V}^{\mathsf{ex}}_{\mathsf{Avg}}[\mathbf{H_2^c}]$ and $\mathsf{V}^{\mathsf{ex}}_{\mathsf{Avg}}[\mathbf{H_b^c}]$ are computed as follows. We build the hierarchy such that each node's noise is represented as a linear combination of the raw noises added to all nodes in the tree, and then apply constrained inference. That is, for a tree with M nodes, in each node we maintain a size-M vector of coefficients, and update these coefficients via weighted averaging and mean consistency. This requires $O(N^2)$ space, and thus we were able to compute this only up to $N = 2^{11}$. For N between 2^4 and 2^9, we iterate over all queries. For each query, we sum up the leaf nodes included in the query to obtain the coefficients of final result noise, and then we compute the variance by summing up the square of the coefficients. We then average the variance over all queries. This way we obtain the *exact* error variance. For $N = 2^{10}$ and $N = 2^{11}$, enumerating through all $N(N+1)/2$ queries takes too long; we thus randomly sampled 100,000 queries and compute their exact variance, and then take their average. We add a superscript of ^{s} to denote that the numbers are obtained via sampled queries. These results are very close to the exact values, because we are computing exact variance for each query and have sampled 100,000 queries from the same distribution that queries are drawn.

Column b shows the best branching factor for $\mathbf{H_b}$. We compute these b values using two methods. One is to find the b values between 2 and 1024 that minimizes $\mathsf{V}^*_{\mathsf{Avg}}[\mathbf{H_b}]$. The other is to do exact counting for each b between 2 and 1024 and choose the optimal ones. For all N values in the table, these two methods result in exactly the same optimal b values.

From Table 1, we make the following observations. First, $\mathsf{V}^*_{\mathsf{Avg}}[\cdot]$ and $\mathsf{V}^{\mathsf{ex}}_{\mathsf{Avg}}[\cdot]$ are very close for larger N's. Second, for N values up to 128, setting $b = N$, i.e., using the flat method is the best method. (We found via experiments that

N	b	h	$\mathsf{V}^*_{\mathsf{Avg}}[\mathbf{H}^{\mathbf{c}}_{\mathbf{b}}]$	$\mathsf{V}^{\mathsf{ex}}_{\mathsf{Avg}}[\mathbf{H}^{\mathbf{c}}_{\mathbf{b}}]$	$\mathsf{V}^{\mathsf{ex}}_{\mathsf{Avg}}[\mathbf{H}_{\mathbf{b}}]$	$\frac{\mathsf{V}^{\mathsf{ex}}_{\mathsf{Avg}}[\mathbf{H}_{\mathbf{b}}]}{\mathsf{V}^{\mathsf{ex}}_{\mathsf{Avg}}[\mathbf{H}^{\mathbf{c}}_{\mathbf{b}}]}$
2^4	16	1	12.00	12.00	12.00	1
2^5	32	1	22.67	22.67	22.67	1
2^6	8	2	36.79	37.07	67.23	1.81
2^7	12	2	54.33	54.17	103.14	1.90
2^8	16	2	78.86	79.23	150.98	1.91
2^9	23	2	114.12	114.02	221.57	1.94
2^{10}	32	2	163.88	156.64^s	373.09	2.05
2^{11}	13	3	215.13	201.88^s	472.24	2.34

Table 2: Effect of constrained inference. b is the best branching factor by our estimation. $\epsilon = 1.0$, so $\mathsf{V}_{\mathsf{u}} = 2.0$

for up to $N = 187$, the flat method outperforms $\mathbf{H_b}$ for any b values.) Third, and the optimal branching factor is oscillating between 16 and a increasingly tighter upper-bound as N increases. Fourth, choosing the optimal branching factor reduces the error variance over $\mathbf{H_2}$ by a factor that stabilizes around 4.5 as N increases. Fifth, adding constrained inference reduces the error variance by a factor of between 2 and 4 when $b = 2$; and the reduction effect is less when b is large. Finally, $\mathbf{H}^{\mathbf{c}}_{\mathbf{b}}$ with optimal branching factor performs the best.

Table 2 is used to check our analysis for constrained inference. $\mathsf{V}^*_{\mathsf{Avg}}[\mathbf{H}^{\mathbf{c}}_{\mathbf{b}}]$ is computed using Algorithm 2. The branching factor b in the table is computed by finding the b value that minimizes $\mathsf{V}^*_{\mathsf{Avg}}[\mathbf{H}^{\mathbf{c}}_{\mathbf{b}}]$. We observe that our estimated $\mathsf{V}^*_{\mathsf{Avg}}[\mathbf{H}^{\mathbf{c}}_{\mathbf{b}}]$ is very accurate for hierarchies of levels 1 and2, but overestimate the error of $\mathbf{H}^{\mathbf{c}}_{\mathbf{b}}$ for 3-levels. As explained earlier, this is because our method does not account for cross-level error reductions.

We also compute the best branching factor by minimizing $\mathsf{V}^{\mathsf{ex}}_{\mathsf{Avg}}[\mathbf{H}^{\mathbf{c}}_{\mathbf{b}}]$. It turns out that this results in exactly the same b values, except for the case of $N = 2^{10}$. Below we show the values for the two branching factor values, with $b*$ chosen using estimated error.

N	b^*	$\mathsf{V}^*_{\mathsf{Avg}}[\mathbf{H}^{\mathbf{c}}_{\mathbf{b}*}]$	$\mathsf{V}^{\mathsf{ex}}_{\mathsf{Avg}}[\mathbf{H}^{\mathbf{c}}_{\mathbf{b}*}]$	b	$\mathsf{V}^*_{\mathsf{Avg}}[\mathbf{H}^{\mathbf{c}}_{\mathbf{b}}]$	$\mathsf{V}^{\mathsf{ex}}_{\mathsf{Avg}}[\mathbf{H}^{\mathbf{c}}_{\mathbf{b}}]$
2^{10}	32	163.88	163.81^s	11	169.01	156.64^s

We point out that the fact our method can choose the true optimal branching factor for $N = 2^{11}$ suggests that while Algorithm 2 underestimates the error variance, the bias has minimal impact when one compares different branching factors that result in the same number of levels.

3.5 Effect of Privacy Budget Allocation

Another optimization that affects the error of hierarchical methods is the division of privacy budget. So far we have assumed that the privacy budget is divided equally among h levels. In [5], a geometric privacy budget allocation scheme has been proposed, where each level has privacy budget that is $\sqrt[3]{b}$ of its parent level, because each level has b times the number of nodes as the parent level. Intuitively, an optimal privacy budget allocation should consider how much each level contributes to the total error variances for all queries, and then allocate privacy budget accordingly. In general, the average error variance can be written as

$$\mathbf{E}_{\mathcal{M}} = \frac{a_1}{\epsilon_1^2} + \ldots + \frac{a_h}{\epsilon_h^2} \tag{5}$$

where ϵ_i is the privacy budget allocated to level i, and the a_i is level i's weight. Using Lagrange multipliers, Equation (5) is minimized by

$$\epsilon_1 = \frac{\epsilon}{\sum_{i=1}^{h} \left(\frac{a_i}{a_1}\right)^{\frac{1}{3}}}; \qquad \epsilon_i = \left(\frac{a_i}{a_1}\right)^{\frac{1}{3}} \cdot \epsilon_1$$

For $\mathbf{H_b}$, a_i equals $M(i)$ in Equation (2), which gives the total number of nodes at level i used to answer be the number of nodes needed in level i. For $\mathbf{H}^{\mathbf{c}}_{\mathbf{b}}$, however, it is difficult to come up with an analytical formula for $a(i)$, in part because the $a(i)$'s are also determined by the ϵ_i's.

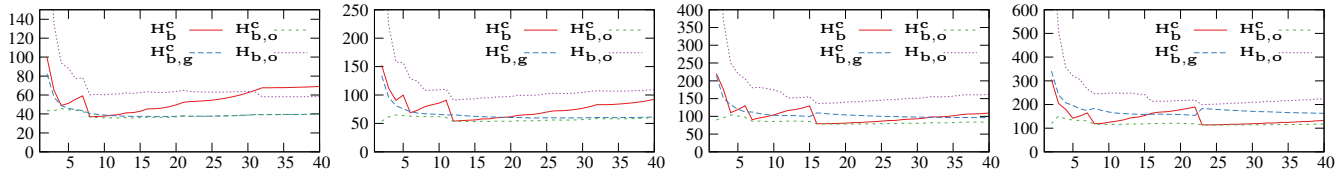
To understand the effect of privacy budget allocation, we propose an iterative process to compute a locally optimal privacy budget allocation for $\mathbf{H}^{\mathbf{c}}_{\mathbf{b}}$, when N and b are given. Starting from equal privacy budget allocation, we compute the $a(i)$'s, by summing this up for all queries using the method of maintaining noise coefficients described in Section 3.4. We then allocate privacy budget to minimize Equation (5), and iterate. We stop until the privacy budget allocation changes very little. In our experiments, this search always terminates after a few rounds.

In Figure 2, we plot the average error variance against the branching factor for four algorithms $\mathbf{H_{b,o}}, \mathbf{H^c_b}, \mathbf{H^c_{b,o}}, \mathbf{H^c_{b,g}}$. $\mathbf{H_{b,o}}$ does not use constrained inference, but allocate privacy budget optimally. $\mathbf{H}^{\mathbf{c}}_{\mathbf{b}}$ uses equal privacy budget allocation, $\mathbf{H}^{\mathbf{c}}_{\mathbf{b,g}}$ uses geometric privacy budget allocation, and $\mathbf{H}^{\mathbf{c}}_{\mathbf{b,o}}$ uses locally optimal budget allocation obtained through iteration. Again, our computations are exact.

From Figure 2, we observe that using constrained inference is more important than choosing optimal privacy budget allocation. We also observe that constrained inference with optimal privacy budget allocation performs the best across all branching factors; however, $\mathbf{H}^{\mathbf{c}}_{\mathbf{b}}$ with equal privacy budget allocation, can match the performance of $\mathbf{H}^{\mathbf{c}}_{\mathbf{b,o}}$ when b is close to the optimal, e.g., when b is between 8 and 16. At the same time, the curve for $\mathbf{H}^{\mathbf{c}}_{\mathbf{b,o}}$ is quite flat, showing that if one applies constrained inference and optimal privacy budget allocation, then a wider range of branching factors would provide similarly good results.

3.6 Recommendation: $\mathbf{H}^{\mathbf{c}}_{\mathbf{b}}$

In summary, our analysis and experimental results suggest that using $\mathbf{H}^{\mathbf{c}}_{\mathbf{b}}$ with branching factor between 8 and 16 provides excellent results, and it is unnecessary to further optimize privacy budget allocation. We now consider the computational complexity of $\mathbf{H}^{\mathbf{c}}_{\mathbf{b}}$. The space complexity is linear in the number of nodes in the hierarchy, which is $\sum_{i=1}^{h} \frac{N}{b^i} < \frac{b}{b-1} N \leq 2N$. The time complexity is linear in $\max(M, N)$, which M is the number of data points, and N is the number of unit bins. The algorithm has the following step. Step 1, scan all data points to compute the true count of each leaf node, which takes time linear in M. Step 2, generate noises for each leaf node, and then visit all nodes in the hierarchy from bottom up to compute the true count, raw noisy count, and count after weighted averaging. The cost for the leaf level is linear in N, as the cost for each leaf node is constant. The cost for the level up is b times the number of nodes at that level, and is thus also linear in N. The next level up costs $\frac{N}{b}$, and the next level up costs $\frac{N}{b^2}$, and so on. Thus the total cost is linear in N. Step 3, updates each internal node via the mean consistency step and then publish all leaf counts. This also has complexity linear in N.



(a) $N = 2^6$ (b) $N = 2^7$ (c) $N = 2^8$ (d) $N = 2^9$

Figure 2: Effect of private budget division. $\mathbf{H_{b,g}}$ and $\mathbf{H^c_{b,g}}$ use geometric allocation of privacy budget. $\mathbf{H_{b,o}}$ and $\mathbf{H^c_{b,o}}$ use optimized allocation of privacy budget. $\epsilon = 1.0$, so $\mathsf{V_u} = 2.0$. X-axis indicates branching factor b. Y-axis indicates $\mathsf{V^{ex}_{Avg}}$

4. COMPARING $\mathbf{H^c_b}$ WITH EXISTING METHODS

In this section, we compare our suggested $\mathbf{H^c_b}$ method with other state of the art methods in the literature. We chosen two b values, 12 and 16.

4.1 Existing Methods

Haar Wavelet Transformation - $\mathbf{W_H}$ Xiao et al. [21] introduced the Privlet method for histogram publishing.The method works by performing a discrete Harr wavelet transformation of the data, and then adding noise to the Haar coefficients. Intuitively, one can think of a Harr wavelet transform as a binary tree with N leaves corresponding to histogram bins. Each non-leaf node stores the Harr coefficient with value $(a_\ell - a_r)/2$, where a_ℓ is the average of all leaves in the left subtree, and a_r is the average of all leaves in the right subtree. When a range query r is answered, one sums up all leaf nodes, which becomes a weighted sum of the noisy Harr coefficients. For any coefficient, each leaf from its left subtree causes that coefficient to be added once, and each leaf from its right subtree causes it to be subtracted once. This results in a desirable noise "cancellation" effect when the number of leaves from either subtree are equal. This effect is most prominent for large queries.

Structure First Method - SF Xu et al. [23] propose an alternative non-hierarchical mechanism for publishing histograms that can improve upon the Flat method. The basic premise behind their approach is that the accuracy of a differentially-private histogram relies heavily on its structure. In order to generate the optimal histogram structure, [23] proposes using dynamic programming techniques. A target number of bins, k, is chosen, and the optimization is performed to find the best k-bin histogram structure that approximates the original histogram. To make this compliant with differential privacy, the resulting histogram bins are then perturbed. After the structure of the histogram is determined, noise is injected to the bins using the remaining privacy budget (that is, the flat method is applied on the resulting bins). This method is referred to as Structure First (**SF**). The running time of this method is $O(kN^2)$ and $k = O(N)$. This large running time greatly limits the applicability of this method for histograms with more than a few thousand bins.

Matrix Mechanisms - MM Proposed by Li et al. [15], the matrix mechanism is a method of answering a workload of predicate counting queries. Given some workload $\mathbf{W}$, the method finds an alternative set of queries $\mathbf{A}$, called a query strategy. Standard Laplace noise is then added to the strategy queries, the answers to which are used to give accurate answers to the original workload. The total error of the estimates of workload $\mathbf{W}$ using $\mathbf{A}$ is $(\frac{2}{\epsilon^2})\Delta^2_{\mathbf{A}} trace((\mathbf{A}^t\mathbf{A})^{-1}\mathbf{W}^t\mathbf{W})$. In the formula, $\Delta_{\mathbf{A}} = \max_j \sum_{i=1}^{n} |a_{ij}|$ is called the sensitivity of matrix $\mathbf{A}$. A natural way to adapt the matrix mechanism to our problem is to assume that the workload consists of all range queries. The problem is thus finding the best query strategy in order to answer the given workload queries. In [15], this problem is expressed as a semi-definite optimization problem. Unfortunately, given a query workload on a histogram of size N, solving the optimization problem requires time complexity $O(N^6)$. Both the low Rank Mechanism in [24] and the Eigen Select Mechanism in [16] give approximations to the best solution with time complexity $O(N^3)$. For all practical purposes, running the original optimization problem is unfeasible. The running time of the approximations also bars the applicably of the methods to datasets with more than a few hundred bins.

4.2 Exact Comparison

Table 3 shows the exact average error variances of multiple methods. These numbers are exact, except for the numbers with ^{s}, which use sampled queries. We do not include numbers for **SF**, because the method is data-dependent, and all numbers in Table 3 are dataset-independent.

For $\mathbf{H_b}$ and $\mathbf{H^c_b}$, we use the techniques we describe in Section 3. For the wavelet method, we get the Haar coefficient nodes that are used to answer the query and we compute the variances, and the numbers are exact. For the Matrix Mechanisms, we calculate the error variance of the strategy matrix using the formulas in [15, 16]. Since the Matrix Mechanism does not scale for large histograms, we only present results up to $N = 2^9$.

For the Matrix Mechanism, we observe that the eigen-select algorithm performs poorly, probably because the approximation made in [16] is too inaccurate. The low-rank algorithm performs reasonably well. Up to $N \leq 2^8$, it is better than all other methods except for Flat and $\mathbf{H_{16}}$. For the Wavelet Method, we observe that it perform very similarly to $\mathbf{H^c_2}$, but perform worse than $\mathbf{H^c_{16}}$. The quadtree method proposed in [5], when applied to 1-dimensional, is essentially $\mathbf{H^c_2}$, but with geometric privacy budget allocation. This method performs slightly better than $\mathbf{H^c_2}$ for small N's, but performs increasingly worse as N increases. Perhaps this is because when the number of level increases, the privacy budget higher-level nodes receives become too small to be useful. We see that $\mathbf{H^c_{12}}$ and $\mathbf{H^c_{16}}$ are the best-performing algorithm in the table, only outperformed by $\mathbf{F}$ for small N's.

	2^4	2^5	2^6	2^7	2^8	2^9	2^{10}	2^{11}
$\mathsf{V}_{\mathsf{Avg}}[\mathbf{F}]$	12.00	22.67	44.00	86.67	172.00	342.67	684.00	1366.67
$\mathsf{V}^{\mathsf{ex}}_{\mathsf{Avg}}[\mathbf{H_{12}}]$	42.47	65.02	82.85	103.14	301.80	349.41	398.22	801.97
$\mathsf{V}^{\mathsf{ex}}_{\mathsf{Avg}}[\mathbf{H^c_{12}}]$	20.82	29.60	39.31	54.17	116.61	134.51	160.58^s	288.96^s
$\mathsf{V}^{\mathsf{ex}}_{\mathsf{Avg}}[\mathbf{H_{16}}]$	12.00	82.94	105.62	125.16	150.98	439.72	498.87	546.48
$\mathsf{V}^{\mathsf{ex}}_{\mathsf{Avg}}[\mathbf{H^c_{16}}]$	12.00	35.64	45.43	57.91	79.23	163.48	185.94^s	213.87^s
$\mathsf{V}^{\mathsf{ex}}_{\mathsf{Avg}}[\mathbf{H^c_2}]$	34.46	61.34	99.92	152.18	220.06	305.54	409.48^s	535.63^s
Quad= $\mathbf{H^c_{2,g}}$	28.52	49.65	82.60	133.96	214.39	340.92	538.75^s	853.49^s
Wavelet	37.80	64.84	102.92	154.49	221.62	306.31	410.29	536.32
MM, eigen-select	104.74	301.12	857.96	2465.75	7129.72	20389.12		
MM, low-rank	20.21	30.85	60.34	119.42	118.92	453.10		

Table 3: A comparison of the exact average expected error for all range queries under different mechanisms. Empty cells are cases where it is computationally infeasible to compute error variance. $\epsilon = 1.0$, so $\mathsf{V}_\mathsf{u} = 2.0$

4.3 Experimental Comparison

We next provide an experimental comparison using $\mathbf{H^c_b}$ and the current state of the art methods. We do not provide an experimental comparison with the Matrix Mechanisms since they do not scale to the size of the datasets we consider.

Datasets We perform our evaluation using four different datasets. We use two datasets taken from the Integrated Public Use Microdata Series for the USA (IPUMS-USA) [20]. This contains federal census data collected in the US between 2000-2009, and contains around 19.5 million rows. We generate two histograms to run our experiments; one on the age attribute which has 96 distinct values, and one on the income attribute discretized to bins of size 100, for a total of 10,000 bins. The third is the Arxiv HEP-TH (high energy physics theory) [12] citation graph, where each bin, x, in the histogram, is the number of papers that have been cited x times. The result is a sparse histogram with 2414 bins. The fourth is derived from the checkin dataset from the Gowalla location-based social networking website, where users share their locations by checking-in. We projected the coordinates onto the x-axis and discretized to a histogram of 2^{16} bins.

Evaluation Metric. We set $\epsilon = 1.0$. For each dataset, we choose between 10 and 20 evenly spaced query sizes, and for each size, issue 100 queries with randomly generated locations. For each query, we measure the absolute error between the accurate answer and the noisy one.

In order to provide a clearer picture of an algorithm's behavior, we provide two types of graphs. For the line graphs, we compute the average error of the 100 queries that are generated for each size, and report the average. We repeat this 5 times for each method, plotting both the average of 5 runs, and the standard deviation.

For the second type of graph, we use candlesticks to plot the profile of errors for all query sizes. Each candlestick provides 5 pieces of information: the 25 percentile (the bottom of candlestick), the median (the bottom of the box), the 75 percentile (the top of the box), the 95 percentile (the top of the candlestick), and the arithmetic mean (the red bar).

For $\mathbf{H^c_b}$, we use the theoretical analysis in Section 3 to choose the best branching factor and the optimal division of privacy budget.

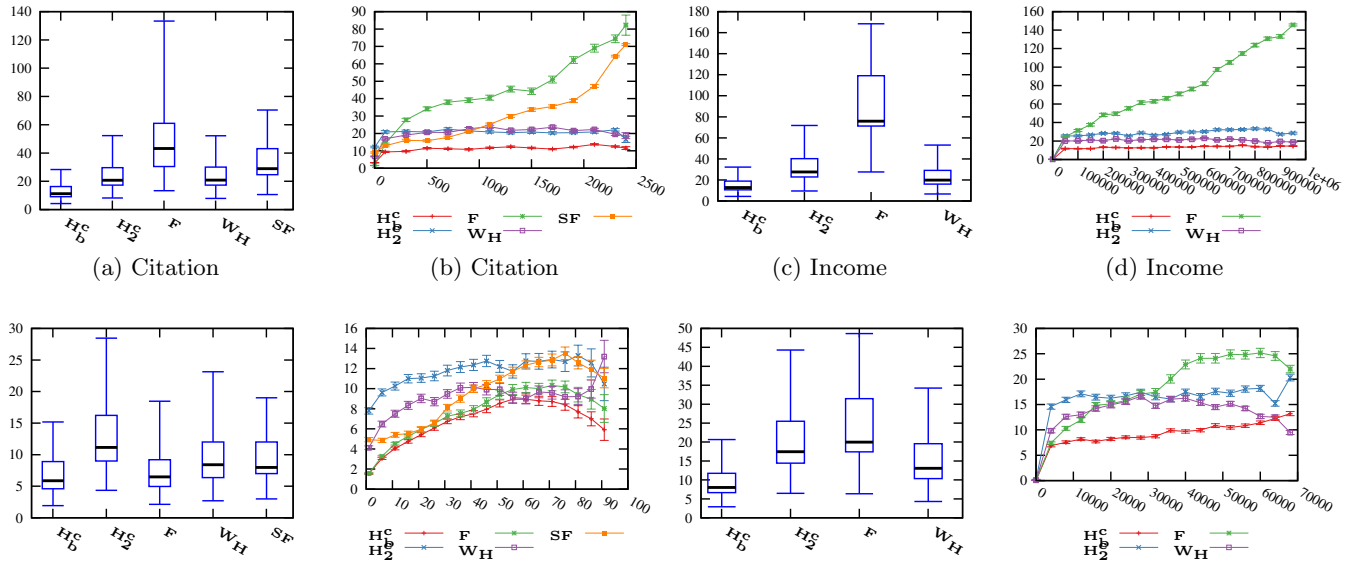
Results We show the results from the comparison in Figure 3. For the Structure First method, we show the results for using the best setting of parameters. Since this method does not scale for histograms with more than a few thousand bins, we only show its results for the age and citation datasets. The results match our previous analysis. The flat method, $\mathbf{F}$, results the largest error. This error is more prominent in larger histograms and larger range sizes (Figure 3d). For smaller histograms, such as Age in Figure 3f, the error of the flat method is small and comparable to the others. $\mathbf{W_H}$ generally outperforms $\mathbf{H^c_2}$, performs especially well when query range sizes are large. due to the noise cancelation effect. **SF** performed better than Flat for the citation dataset (Figures 3b and 3a); however, it was still worse than all other hierarchical methods. In almost all cases, $\mathbf{H^c_b}$ outperformed all the other methods. This matches our analytical prediction.

5. THE EFFECT OF DIMENSIONALITY

We have seen that hierarchical methods provide great accuracy improvement over the flat method for one-dimensional dataset when the number of bins is large. However, many datasets in practice have more than one dimensions. It is thus important to understand whether hierarchical methods can also prove effective for constructing multi-dimensional histograms.

Hierarchies have been applied to 2-dimensional dataset [5], and to more general multi-dimensional dataset [18, 22, 21, 17]. However, it has been recently shown in [19] that for 2-dimensional datasets, the simple flat method, called the Uniform Grid method in [19], with carefully selected grid size can perform as well as sophisticated hierarchical methods with constrained inference and optimized privacy budget allocation. It is also pointed out in [19] that, the benefit of binary hierarchies for the 2-dimensional case is quite limited, and the benefit can only decrease with increasing dimensionality. The observation is that in a hierarchical method, one needs to spend privacy budgets on higher levels, and thus has less privacy budget to use for the leaf nodes. Any query can be viewed as consisting of two regions: the interior region, which can be answered by higher-level nodes, and the border region of certain constant "thickness", which has to be answered using leaf nodes. When using a hierarchy, one is trading off inaccuracy for the border region with improved accuracy for the interior region. This tradeoff is beneficial only when the ratio of the volume of border region to the volume of the whole query region is small. With a constant-thickness border, the higher the dimensionality, the larger the portion of the border region takes. When the dimensionality is high, almost all volume of a query are in the border region.

Here we investigate the effectiveness of hierarchical methods on higher-dimensional datasets by analyzing the aver-



(a) Citation (b) Citation (c) Income (d) Income

(e) Age (f) Age (g) Gowalla Checkin (h) Gowalla Checkin

Figure 3: Experimental comparison for all dataset

age error variance. We also estimate the number of unit bins needed for one to benefit from using the hierarchical method at all for high-dimensional datasets.

We consider a d-dimensional histogram with $N = n \times n \times \ldots n = n^d$ unit bins. The objective is to publish the histogram in a way that both satisfies differential privacy, and is able to answer range queries with high accuracy. For simplicity, we consider the case of a hierarchy with branching factor b^d. That is, each dimension has branching factor b. We use $\mathbf{H_b}$ to denote this method. We use $h = \lceil \log_b n \rceil = \lceil \log_{b^d} N \rceil$ to denote the height of the resulting tree.

5.1 The 2-Dimensional Case

PROPOSITION 5. *For 2-dimensional datasets with $n \times n$ bins and a hierarchy with branching factor $b \times b$, on average a random rectangular range query has noise variance*

$$\mathsf{V}_{\mathsf{avg}}(\mathbf{H_b}) = \Theta\left(b(n-1)\lceil \log_b n \rceil^2\right) \mathsf{V}_{\mathsf{u}} \tag{6}$$

Where we define $\mathsf{V}_{\mathsf{u}} = \frac{2}{\epsilon^2}$.

PROOF. We consider on average how many nodes at each level are needed to answer a randomly selected query. Let j denote the depth of nodes in a tree such that the root is at depth 0, the next level at depth 1, and the leaves are at depth h. At depth j, we have a $b^j \times b^j$ grid. A rectangular query has 4 border edges. Consider, wlog, the left border edge. For the j'th level, the edge includes between 0 and $b-1$ columns, with each being equally likely; thus on average it includes $(b-1)/2$ columns. Each full column includes b^j nodes at the j'th level; and on average a query's edge is of length $(b^j+2)/3 \approx b^j/3$. (Recall that $\frac{\sum_{j=1}^{N} j(N-j+1)}{N(N+1)/2} = \frac{(N+2)}{3}$.) Therefore, a query includes on average about $4 \times (b-1)/2 \times b^j/3 = \frac{2}{3}(b-1)b^j$ nodes at level j. The total number of nodes from all levels is thus

$$\sum_{j=1}^{h} \frac{2}{3}(b-1)b^j = \frac{2}{3}b(b^h - 1) = \frac{2}{3}b(n-1)$$

Each node contributes variances of the scale $h^2 = \lceil \log_b n \rceil^2$; the total variance is thus $b(n-1)\lceil \log_b n \rceil^2$. □

From the above proposition, we can see that the benefit of adopting hierarchical methods relative to the flat method is significantly less than that of the one-dimensional case. For the one dimensional case, one is able to reduce average noise variance to $2(b-1)\lceil \log_b N \rceil^3$. For the two-dimensional case, the reduction is to $b(\sqrt{N}-1)\lceil \log_{b^2} N \rceil^2$, which includes a $\sqrt{N}$ term. Note that roughly we are exchanging a $\log N$ factor with a $\sqrt{N}$ factor. Intuitively, this is because in 1-dimensional case, a "border" of constant width has a volume that is $\Theta(\frac{1}{N})$ of the total domain, whereas in 2-dimensional case, a "border" of constant width occupies a volume that is $\Theta(\frac{1}{\sqrt{N}})$ of the total domain.

5.2 The d-dimensional case

We can apply similar analysis to the d-dimensional case.

PROPOSITION 6. *For d-dimensional datasets with $N = n^d$ unit bins and a hierarchy with branching factor b^d, where $d \geq 2$, we have*

$$\mathsf{V}_{\mathsf{avg}}(\mathbf{H_b}) = \Theta\left(\frac{d \times b^{d-1}}{b^{d-2} + \cdots + 1}\left(\frac{1}{3}\right)^{d-1}\left(N^{(d-1)/d} - 1\right)\right)\mathsf{V}_{\mathsf{u}} \tag{7}$$

PROOF. At each level j, we have a $b^j \times b^j \times \cdots \times b^j$ grid of b^{jd} nodes. A d-dimensional hyper-rectangular query's border consists of $2d$ sides. One side consists of on average $(b-1)/2$ hyperplanes of $d-1$ dimension. Each whole hyperplane contains $b^{j(d-1)}$ nodes, and a query's side contains on average $\approx (1/3)^{d-1}$ portion of it. The total number of nodes for all levels is thus

$$\begin{aligned}
& \textstyle\sum_{j=1}^{h} 2d * (b-1)/2 * b^{j(d-1)}(1/3)^{d-1} \\
= \; & d(b-1)(1/3)^{d-1} \textstyle\sum_{j=0}^{h-1} (b^{d-1})^j \\
= \; & d(b-1)(1/3)^{d-1} \frac{(b^{d-1})^h - 1}{b^{d-1}-1} \\
= \; & d\frac{b^{d-1}}{b^{d-2}+b^{d-3}+\cdots+1}(1/3)^{d-1}\left(N^{(d-1)/d} - 1\right)
\end{aligned}$$

We note that the second step above applies only when $d \geq 2$. □

Asymptotically, the average squared error is $\Theta(N^{(d-1)/d})$, a number that is increasingly close to N as d increases.

5.3 When Flat is Best.

In Section 3, we have shown that in the one-dimensional case for a smaller value of N. We enumerated through all N values and found that up until $N = 45$, the Flat method outperforms all Hierarchical methods. From Proposition 6, we can estimate how large N needs to be for different dimensions in order for us to see benefit of using hierarchical methods. Note that from the proof of Proposition 6 that the formula in Equation (7) actually keeps all relevant constants. We can thus compare the formula to that of the flat method, to analyze what values of N would we start seeing an improvement. For the flat method, on average a query would need to be answered by $\frac{N}{3^d}$ nodes. For the hierarchical method, we need to estimate the error reduction due to the constrained inference step. Note that we are only considering whether a 2-level hierarchy improves upon flat. The leaf level benefits from mean consistency to reduce average variance to slightly more than $\frac{1}{3}$ of the original, from analysis similar to that in Section 3.3. The next level up benefits only from weighted averaging, which reduces error to $\frac{b^d}{b^d+1}$, which is close to 1. However, since the leaf level will dominate the overall variance, we estimate constrained inference can reduce variance by a factor of 3. The following table gives the condition under which a hierarchy with constrained inference will result in a improvement.

$d = 2$	$\frac{2}{3}b(\sqrt{N}-1)\lceil\log_{b^2} N\rceil^2 \times \frac{1}{3} < N/9$
approx cond:	$2b\lceil\log_b n\rceil^2 \leq n$
solution:	$b = 8, n \geq 64, N \geq 8^4 = 4096$
$d = 3$	$\frac{3b^2}{9(b+1)}(N^{2/3}-1)\lceil\log_{b^3} N\rceil^2 \times \frac{1}{3} < N/27$
approx cond:	$3\frac{b^2}{b+1}\lceil\log_b n\rceil^2 \leq n$
solution:	$b = 11, n \geq 11^2 = 121, N \geq 11^6 = 1.7E6$
$d = 4$	$\frac{4}{27}\frac{b^3}{2(b^2+b+1)}(N^{3/4}-1)\lceil\log_{b^4} N\rceil^2 \times \frac{1}{3} < N/81$
approx cond:	$4\frac{b^2}{b+1}\lceil\log_b n\rceil^2 \leq n$
solution:	$b = 6, n \geq b^3 = 216, N \geq 6^{12} = 2.18E9$

To solve the approximate condition above, we try different values for $h = \lceil\log_b n\rceil$, e.g., $2, 3, 4$, and then compute the smallest b that satisfies the condition, this yields a value n. We then choose the n that is the smallest among all choices of h.

This suggests that for 2-dimensional datasets, one could benefit from using a hierarchy roughly when $N \geq 4096$, which likely apply for most 2-dimensional datasets of interest. For 3-dimensional datasets, however, only very large ones (i.e., those with N more than 1 million unit bins), one could benefit from hierarchy. For 4 dimensional datasets, we would need billions of unit bins to benefit from using a hierarchy.

In Figure 4, we experimentally verify the above analysis. For each setting, we uniformly randomly sample 10,000 queries and compute their the average error variance under different methods. From Figure 4(a), we can see that for $N = 2^{10}$, no hierarchical method beats $\mathbf{F}$. In Figure 4(b), where $N = 2^{12}$, we can see that a branching factor of 8×8 performs the best and it is slightly better than $\mathbf{F}$. For $N = 2^{14}$, we have $b \approx \sqrt{n = 2^7}$ performs the best because it results in a 2-level hierarchy. Figure 4(d) shows the result for 3-dimensional data with $N = 2^{12}$. We see that no hierarchical method beats $\mathbf{F}$. We are unable to run $\mathbf{H_b^c}$ on large enough N values to show a benefit of hierarchical methods.

6. RELATED WORK

Differential privacy was presented in a series of papers [7, 10, 2, 9, 8]. Recent literature has focused on using differential privacy for data publishing scenarios [3, 11].

Our paper is inspired by Hay et al.'s work [14], which proposed the hierarchical method for answering histogram queries. They also introduced the notion of constrained inference to improve query accuracy. Their work, however, use only binary hierarchies. They analyzed the benefit provided by constrained inferencing, and showed that there exist queries that can benefit significantly from constrained inference. They, however, did not conduct an average case analysis. In [21], Xiao et al. proposed the wavelet methods for answering histogram queries.

In [23], Xu et al. explore methods of constructing an optimal histogram using ϵ-DP. The histogram counts are then returned using the flat method. In this paper, we experimentally compare them to our proposed $\mathbf{H_b^c}$ method.

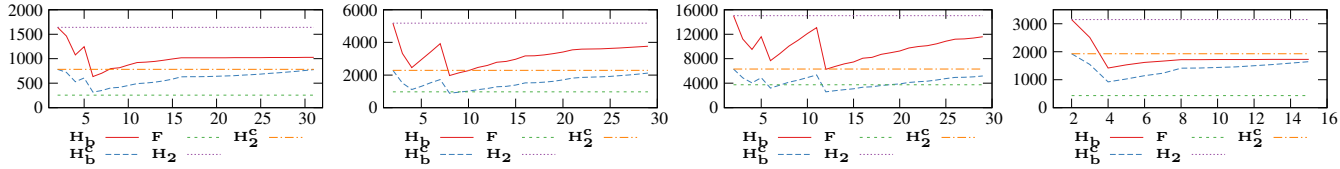
Barak et al. [1] considered the problem of publishing marginals over high-dimensional datasets with many binary attributes. This is orthogonal to the problem we consider, which publishes full contingency table for low-dimensional datasets where each attribute has many values. Ding et l. [6] performed consistency and noise optimization for marginal publishing. When applied to the setting of publishing the full histogram, this is equivalent to the flat method.

Other techniques for analyzing general query workloads under ϵ-DP have been discussed in [13], but no optimizations are offered. In [15], Li et al. discuss methods of optimizing linear count queries. They consider that the queries one wishes to answer are known beforehand, and then provide a theoretical method of optimizing differentially private solutions to such queries. Practical approximations of these methods have been developed in [24, 16]. When applied to our setting of optimizing for all possible range queries, these methods do not scale, and they underperform $\mathbf{H_b^c}$. In [15], it is also shown that for any query, $\mathbf{H_2^c}$ and the Wavelet method result in expected errors that are within a factor of 2 of one another. The techniques in [15] can also be used to compute the error variance of $\mathbf{H_b^c}$ for smaller N values.

In [3], Blum et al. propose a method that can be used to answer histogram queries. Their method is closest to the flat method, and offers minimal improvement to that. Hay et al. [14] analyzed the method in [3], and show its error is $O(N^{2/3})$, which is worse than the hierarchical methods considered in this paper.

7. CONCLUSION

In this paper, we analyzed the effectiveness of using hierarchical methods for differentially private histogram publication. We consider the utility of the published histogram in terms of its ability to range queries over the data domain accurately. First, we perform a thorough analysis of hierarchical methods in one dimension. We show how to select the optimal branching factor with or without constrained inference. We show that combining an optimal branching



(a) $d = 2, N = 2^{10}/n = 2^5$ (b) $d = 2, N = 2^{12}/n = 2^6$ (c) $d = 2, N = 2^{14}/n = 2^7$ (d) $d = 3, N = 2^{12}/n = 2^4$

Figure 4: A comparison of the average error for larger dimensions. $\epsilon = 1.0$, so $\mathsf{V}_u = 2.0$. X-axis indicates branching factor b. Y-axis indicates V^{ex}_{Avg}

factor with constrained inference suffices to provide excellent results, and it is unnecessary to optimize privacy budget allocation. We then compare our proposed method with the current state of the art methods, and show that our optimizations advance the current state of the art. Finally, we extend our analysis to multiple dimensions. We show how to use analysis to find the minimal domain size in order for the hierarchical method to outperform the flat method.

Acknowledgement

This paper is based upon work supported by the United States National Science Foundation under Grant No. 1116991, and by the United States AFOSR under grant titled "A Framework for Managing the Assured Information Sharing Lifecycle".

We would like to thank the reviewers for their helpful comments, which helped improve the paper.

8. REFERENCES

[1] B. Barak, K. Chaudhuri, C. Dwork, S. Kale, F. McSherry, and K. Talwar. Privacy, accuracy, and consistency too: a holistic solution to contingency table release. In *PODS'07*, pages 273–282, 2007.

[2] A. Blum, C. Dwork, F. McSherry, and K. Nissim. Practical privacy: the SuLQ framework. In *PODS*, pages 128–138, 2005.

[3] A. Blum, K. Ligett, and A. Roth. A learning theory approach to non-interactive database privacy. In *STOC*, pages 609–618, 2008.

[4] T.-H. H. Chan, E. Shi, and D. Song. Private and continual release of statistics. *ACM Trans. Inf. Syst. Secur.*, 14(3):26:1–26:24, Nov. 2011.

[5] G. Cormode, M. Procopiuc, E. Shen, D. Srivastava, and T. Yu. Differentially private spatial decompositions. In *ICDE*, pages 20–31, 2012.

[6] B. Ding, M. Winslett, J. Han, and Z. Li. Differentially private data cubes: optimizing noise sources and consistency. In *SIGMOD*, pages 217–228, 2011.

[7] I. Dinur and K. Nissim. Revealing information while preserving privacy. In *PODS*, pages 202–210, 2003.

[8] C. Dwork. Differential privacy. In *ICALP*, pages 1–12, 2006.

[9] C. Dwork, F. Mcsherry, K. Nissim, and A. Smith. Calibrating noise to sensitivity in private data analysis. In *TCC*, pages 265–284. Springer, 2006.

[10] C. Dwork and K. Nissim. Privacy-preserving datamining on vertically partitioned databases. In *In CRYPTO*, pages 528–544. Springer, 2004.

[11] A. Friedman and A. Schuster. Data mining with differential privacy. In *KDD '10*, pages 493–502, New York, NY, USA, 2010. ACM.

[12] J. Gehrke, P. Ginsparg, and J. M. Kleinberg. Overview of the 2003 kdd cup, 2003.

[13] M. Hardt and K. Talwar. On the geometry of differential privacy. STOC '10, pages 705–714, New York, NY, USA, 2010. ACM.

[14] M. Hay, V. Rastogi, G. Miklau, and D. Suciu. Boosting the accuracy of differentially private histograms through consistency. *PVLDB*, 3:1021–1032, September 2010.

[15] C. Li, M. Hay, V. Rastogi, G. Miklau, and A. McGregor. Optimizing linear counting queries under differential privacy. PODS '10, pages 123–134. ACM, 2010.

[16] C. Li and G. Miklau. An adaptive mechanism for accurate query answering under differential privacy. *Proc. VLDB Endow.*, 5(6):514–525, Feb. 2012.

[17] N. Mohammed, R. Chen, B. Fung, and P. S. Yu. Differentially private data release for data mining. In *Proceedings of the 17th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 493–501. ACM, 2011.

[18] W. Qardaji and N. Li. Recursive partitioning and summarization: A practical framework for differential private data publishing. In *ASIACCS*, 2012.

[19] W. Qardaji, W. Yang, and N. Li. Differentially private grids for geospatial data, 2012. arXiv:1209.1322 [cs.CR] To appear in ICDE 2013.

[20] S. Ruggles, J. T. Alexander, K. Genadek, R. Goeken, M. B. Schroeder, and M. Sobek. Integrated public use microdata series: Version 5.0 [machine-readable database], 2010.

[21] X. Xiao, G. Wang, and J. Gehrke. Differential privacy via wavelet transforms. *IEEE Transactions on Knowledge and Data Engineering*, 23:1200–1214, 2011.

[22] Y. Xiao, L. Xiong, and C. Yuan. Differentially private data release through multidimensional partitioning. In *Secure Data Management*, pages 150–168, 2010.

[23] J. Xu, Z. Zhang, X. Xiao, Y. Yang, and G. Yi. Differentially private histogram publication. In *ICDE '12: Proceedings of the 2012 IEEE 28th International Conference on Data Engineering*. IEEE Computer Society, 2012.

[24] G. Yuan, Z. Zhang, M. Winslett, X. Xiao, Y. Yang, and Z. Hao. Low-rank mechanism: optimizing batch queries under differential privacy. *Proc. VLDB Endow.*, 5(11):1352–1363, July 2012.