Designing Efficient Algorithms for Approximating Probabilistic Data

Stylianos-Georgios Mammas

Department of Electronic and Computer Engineering

Technical University of Crete

A thesis submitted for the degree of Master of Science

February 11, 2013

Supervisory Committee:

1. Assistant Prof. Antonios Deligiannakis (Supervisor)
2. Prof. Minos Garofalakis
3. Assistant Prof. Vasilis Samoladas

Contents

List of Figures

Dedicated to my father.

Acknowledgements

I want to express my appreciation to my supervisor Dr. Antonios Deligiannakis for his guidance. I am grateful to the members of the supervisory committee. A big thank you to my friends, family and colleagues for their patience, love and support.

Introduction

Histograms have proven to be a very effective summarization mechanism, and are widely used to capture data distributions. Currently, they are an important part in commercial query engines.

Let a data distribution of tuple frequencies. Typically, an approximation of such data distribution with a histogram, is obtained by partitioning the data domain into a small number of consecutive ranges (the buckets), and by storing only concise statistics to summarize the tuple frequencies within a bucket. The bucket boundaries are chosen to minimize a given error metric that measures within-bucket dissimilarities, i.e., the differences of the concise statistic selected for the bucket (e.g., the average value of tuple frequencies within the bucket) from the real tuple frequencies, and aggregates errors across the buckets (using summation or maximum). The fast computation of such synopses is a crucial parameter in Data Base Management Systems (DBMSs), in terms of fast approximate answers from the query engines.

Moreover, since histograms are themselves approximation schemes, an extra approximation factor in the histogram construction process for speed up, can be tolerable. We present such an $(1+\varepsilon)$-approximation scheme for histogram construction of one-dimensional deterministic data in Section 1.1.

Another topic of this thesis, is the integration of such mechanisms in Probabilistic Data Base Management Systems (PDBMSs), i.e., integrating methods for probabilistic histogram construction over probabilistic data, since there is a growing realization that modern DBMSs must be able to manage data that contain *uncertainties* that are represented in the form of probabilistic relations. In Section 1.2, we present two-step Dynamic Programming (DP) algorithms for building optimal probabilistic histograms, for a variety of error metrics [8].

The main contribution of this thesis (Chapter 2), is the incorporation of $(1+\varepsilon)$-approximation schemes in the histogram construction process, in the case of two- dimensional probabilistic data. In order to reduce the high computational cost of optimal probabilistic histogram construction, we set an approximation scheme to the general DP framework. The speed up is obtained, because our approximate algorithm takes

into consideration only a small set of sub-problems in the DP framework. Our algorithm provides guarantees to the overall error of the histogram, and gives the user a useful trade-off between the accuracy of the histogram and the computational cost.

Our experimental study, to explore the effect of such approximation, shows that our approximation scheme produces almost-optimal probabilistic histograms, with much lower computational cost than the optimal algorithm.

Chapter 1

Theoretical Background and Related Work

Histograms have proven to be a very effective summarization mechanism, and are widely used to capture data distributions. Currently, they are an important part in commercial query engines. Let a data distribution of tuple frequencies. Typically, an approximation of such data distribution with a histogram, is obtained by partitioning the data domain into a small number of consecutive ranges (the buckets), and by storing only concise statistics to summarize the tuple frequencies within a bucket. The bucket boundaries are chosen to minimize a given error metric that measures within-bucket dissimilarities, i.e., the differences of the concise statistic selected for the bucket (e.g., the average value of tuple frequencies within the bucket) from the real tuple frequencies, and aggregates errors across the buckets (using summation or maximum).

The fast computation of such synopses is a crucial parameter in Data Base Management Systems (DBMSs), in terms of fast approximate answers from the query engines. Moreover, since a histogram is by itself an approximation scheme, an extra approximation factor in histogram construction process for speed up, can be tolerable. We present such an approximation scheme for histogram construction of one-dimensional deterministic data in Section 1.1. Another topic of this thesis, is the integration of such mechanisms in Probabilistic Data Base Management Systems (PDBMSs), i.e., integrating methods for probabilistic histogram construction over probabilistic data.

In Section 1.2, we present DP programming schemes for optimal, in terms of overall error, probabilistic histogram construction.

1.1 Approximate Histograms over Deterministic Data

S. Guha, N. Koudas, K. Shim [15], [16], present approximate algorithms for histogram construction over one-dimensional deterministic data. Since, a histogram is by itself an approximation of data distribution, they insert an additional approximation factor in histogram construction process, in order to accelerate the computation, with tolerable cost in accuracy of the produced histogram. Specifically, they integrate an approximation scheme in the V-Optimal algorithm, i.e., an algorithm which produces optimal histograms for a variety of error metrics. Also, they proof that their approximation is an $(1+\varepsilon)$-approximation of the corresponding V-Optimal histogram, i.e., their histograms have at most $1+\varepsilon$ times the error of the corresponding V-Optimal histograms.

1.1.1 Preliminaries

Let $X = \{x_1, x_2, ..., x_n\}$ be a finite data sequence. Given a space constraint B, the goal of histogram construction is to create a compact representation of sequence X (Let H_B denote such representation) of space at most B. In histogram construction process, the data sequence is partitioned into B disjoint intervals of consecutive data points, i.e., the data space is split into B buckets, and for each bucket we store only a single value (the bucket representative). The bucket boundaries and bucket representatives are chosen in order to minimize an error metric. Let the error of such approximation of data sequence denoted by $E_X(H_B)$. There are two problem statements delivered from the above.

Optimal Histogram Construction Problem. Given a sequence X of length n, a number of buckets B, find H_B to minimize $E_X(H_B)$ under the given error function E.

$(1+\varepsilon)$-Approximate Histogram Construction Problem. Given a sequence X of length n, a number of buckets B, and a precision parameter $\varepsilon > 0$, find H_B with $E_X(H_B)$

at most $(1+\varepsilon)$ min_H $E_X(H_B)$ where the minimization is taken over all histograms H with B buckets.

Measuring the total error of a histogram with the sum-squared error metric:
One of the most common methods to measure the total error of a histogram is the sum-squared error error metric. This metric, sums the squares of errors of each data point i ($1 \leq i \leq n$). Since the buckets are disjoint and cover the entire domain, i.e., each data point belongs to exactly one bucket, we can express the total error of the histogram as the sum of bucket errors of the histogram. Let $b_1, b_2, ..., b_B$ be the buckets of the histogram. Thus, the total sum-squared error of the histogram is $\sum_{1 \leq i \leq B} SQERROR(b_i)$. Let $b_r = (s_r, e_r)$, be the interval covered from bucket r, i.e., all data points from s_r to e_r belong to bucket r. Based on this notation we can express the error of bucket r as $SQERROR(b_r) = \sum_{s_r \leq i \leq e_r} (x_i - h_r)^2$, where with h_r we denote the representative value of bucket r. This quantity is minimized by taking as bucket representative the mean of data points belong to the bucket, i.e., $h_r = \frac{1}{e_r - s_r + 1} \sum_{s_r \leq i \leq e_r} x_i$. Thus, after some processing, we can express the bucket error as:

$$SQERROR(s_r, e_r) = \sum_{i=s_r}^{e_r} x_i^2 - \frac{1}{e_r - s_r + 1} (\sum_{i=s_r}^{e_r} x_i)^2 \tag{1.1}$$

V-Optimal Algorithm
In this paragraph we present the V-Optimal algorithm, a dynamic programming algorithm which constructs optimal histograms for a variety of error metrics. Here, we focus on sum-squared error metric [17], but is straightforward for someone, to see how other error metrics can be supported, e.g., Max-Error Measure.

Recall the data sequence $X = \{x_1, x_2, ..., x_n\}$. The V-Optimal algorithm splits this domain into B consecutive disjoint intervals, examining all possible bucket boundaries via a dynamic programming scheme, to find the partition that has the minimum (optimal) sum-squared error. The key observation here is the principle of optimality, that is, if the last bucket contains the data points of the interval $[i+1, n]$ in the optimal histogram, then the rest of the buckets must form an optimal histogram with $B-1$ buckets for the interval $[1, i]$, where $1 \leq i \leq n-1$.

```
Procedure V-OPT()
begin
1.    SUM[1, 1] := v_1
2.    SQSUM[1, 1] := v_1^2
3.    for i := 2 to n do {
4.        SUM[1, i] := SUM[1, i − 1] + v_i
5.        SQSUM[1, i] := SQSUM[1, i − 1] + v_i^2
6.    }
7.    for j := 1 to n do {
8.        TERR[j, k] := ∞
9.        for k := 2 to B do
10.           for i := 1 to j-1 do
11.               TERR[j, k] := min (TERR[j, k], TERR[i, k − 1] + SQERROR(i + 1, j))
12.   }
end
```

Figure 1.1: The V-Optimal Algorithm.

Let TERR[i, k] be the minimum (optimal) sum-squared error in approximation of the interval [1, i], with a k-bucket histogram. Notice that the optimal histogram construction problem (in sum-squared error metric) is to find a histogram with the minimum sum-squared error in the approximation of the interval [1, n], with a B-bucket histogram, i.e., Terr[n, B]. We present the V-Optimal algorithm, in case of sum-squared error metric, in Figure 1.1.

Let us define two vectors of length n, SUM and SQSUM, such that: $SUM[1,i] = \sum_{l=1}^{i} x_l$ and $SUM[1,i] = \sum_{l=1}^{i} x_l^2$, where $1 \leq i \leq n$. By maintaining these vectors, we can compute the bucket error of equation (1.1) in O(1) time, i.e., in constant time, since the partial sums of equation (1.1) can be calculated with the following formulas:
$\sum_{i=s_r}^{e_r} x_i^2 = SQSUM[1,e_r] - SQSUM[1,s_r]$ $\quad$ $\sum_{i=s_r}^{e_r} x_i = SUM[1,e_r] - SUM[1,s_r]$.

The V-Optimal algorithm has $O(n^2B)$ time complexity, since we must calculate $O(nB)$ error entries ($TERR[i,k], 1 \leq i \leq n,\ 1 \leq k \leq B$) in the dynamic programming table, and each entry needs $O(n)$ time to be calculated.

1.1.2 The AHIST-S: An Approximate Algorithm with Small Space

S. Guha, N. Koudas, K. Shim [15] present various techniques, for the approximate histogram construction problem, to produce near-optimal histograms. Since the histograms are themselves used to approximate query answering, an extra approximation

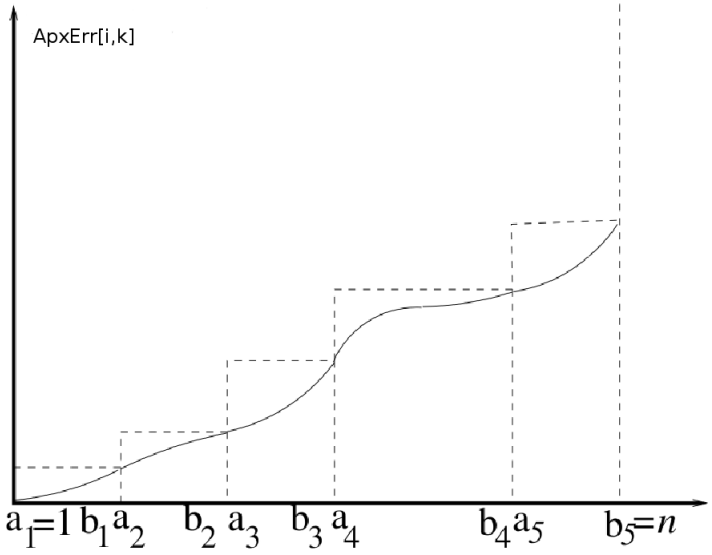


Figure 1.2: Approximating an error curve with a staircase function

factor used to speed up the computation is usually acceptable. In this subsection we present the AHIST-S algorithm [15], [16], an algorithm which integrates an approximation scheme to the classic V-Optimal algorithm.

The key observation they did, was the monotonicity property of the optimal error functions $SQERROR(i,j)$, $TERR(i,k)$ (where $i,j \in \{1,2,..,n\}$ and $k \in \{1,2,..,B\}$), discussed before. Specifically, they observed that $SQERROR(i,j)$ and $TERR(i,k)$ are non-increasing and non-decreasing (both are non-negative), respectively, functions over the values of i (as i increases). Clearly, $TERR(i,k) \leq TERR(i+1,k)$, since the optimal error in approximation of first i data points cannot be larger than the error in approximation of first i+1 data points. Accordingly, the optimal error of bucket (i,j), i.e., $SQERROR(i,j)$, cannot be smaller than the optimal error of bucket (i+1, j), i.e., $SQERROR(i+1,j)$. They used this monotonicity property in their inductive proof, to guarantee that the AHIST-S produces an $1+\varepsilon$ approximation of optimal solution, i.e., $ApxErr[i,k] \leq (1+\varepsilon)TERR[i,k]$ ($\forall i \in \{1,..,n\}$, $\forall k \in \{1,..,B\}$), where with ApxErr we denote the error function of the approximate algorithm AHIST-S.

The key component of their algorithm is that instead of storing/remembering the whole error functions ApxErr(i,k-1) ($i = 1..n$, $k \in \{2,..,B\}$) like the classic V-Optimal

algorithm, they approximate them by staircase functions, as shown in Figure 1.2, reducing the searching space needed from the algorithm. The error curve ApxErr(i,k) ($i = 1..n$), depicts the error in approximation of first i data items using k buckets, as the number of data items increases from 1 to n. Let us explore how such error curves can be approximated by staircase functions. The data domain, i.e., the interval (1, n), is broken down into τ (clearly τ depends on k, i.e., the number of buckets) intervals (a_i, b_i) to approximate the whole error function with a staircase function. These intervals are disjoint and cover the entire domain, i.e., $a_1 = 1$, $a_{i+1} = b_i + 1$ and $b_\tau = n$.

Moreover, these intervals are stored by a $1+\delta$ factor, meaning that the value of error function at the right boundary of an interval cannot be greater than $1+\delta$ times the value of error function at left boundary of the interval, i.e., $Apxerr[b_i,k] \leq (1+\delta)Apxerr[a_i,k]$, $\forall i \in \{1,2,..,\tau\}$. The parameter δ is fixed to be $\frac{\varepsilon}{2B}$ ($\varepsilon < 1$), in order to produce an $(1+\varepsilon)$-approximation of the optimal error curve, i.e., the algorithm computes an $(1+\varepsilon)$-approximate B-bucket histogram. The inductive proof of that can be found at [16].

```
Procedure AHIST-S()
begin
1.    Set up (B − 1) lists Q[k] to store the intervals
2.    SUM := SQSUM := 0
3.    for j := 1 to n do {
4.        SUM := SUM + v_j
5.        SQSUM := SQSUM + v_j^2
6.        for k := 2 to B do {
7.            APXERR[j, k] = ∞
8.            for i := each end point b of interval list for (k − 1)-th list Q[k − 1] do
9.                // Recall APXERR[j,1] = SQERROR(1, j)
10.               APXERR[j, k] = min (APXERR[j, k], APXERR[i, k − 1] + SQERROR(i + 1, j))
11.           // a_ℓ is the start index of the last interval in Q[k]
12.           // b_ℓ is the end index of the last interval in Q[k]
13.           if (k ≤ B − 1 and APXERR[j,k] > (1+δ)APXERR[a_ℓ,k]) {
14.               // Now, we have APXERR[j, k], SUM = SUM[j] and SQSUM = SQSUM[j]
15.               a_{ℓ+1} := b_{ℓ+1} := j
16.               Insert a new interval [a_{ℓ+1},b_{ℓ+1},APXERR[j,k],SUM,SQSUM] to Q[k]
17.           }
18.           else
19.               Set b_ℓ to j
20.       }
21.   }
end
```

Figure 1.3: The AHIST-S Algorithm.

We present the AHIST-S algorithm in Figure 1.3. The algorithm maintains $B-1$ interval lists of bounded size. For each element (interval) of the k-th list, the algorithm stores, the index number x (the left boundary of the interval), the index number y (the right boundary of the interval), Sum[y], Sqsum[y] and Apxerr[y, k] values. The algorithm maintains Apxerr[y, 1] = Terr[y, 1] = Sqerror(1, y), i.e., for representation

by one bucket, it computes the error exactly. Concluding, in [16], the interested reader may see the proof that the algorithm AHIST-S computes an $(1+\varepsilon)$-approximate B-bucket histogram in $O(nB^2\varepsilon^{-1}logn)$ time and $O(B^2\varepsilon^{-1}logn)$ space.

1.2 Optimal Histogram Construction over Probabilistic Data

There is a growing realization that modern DBMSs must be able to manage data that contain *uncertainties* that are represented in the form of probabilistic relations. Thus, there is a need to revisit the problem of creating synopses in presence of uncertain data, for fast approximate answering over probabilistic data.

G. Cormode, A. Deligiannakis, M. Garofalakis, A. McGregor [8] present algorithms for two-dimensional probabilistic histogram construction over probabilistic data. They propose two-step dynamic programming algorithms for optimal probabilistic histogram construction, in terms of minimum possible error for a variety of error metrics.

Although, their algorithms are optimal in terms of total error of the approximation, they have prohibited time complexity and it is impossible to cope with massive data sets. The main contribution of our work in this problem (Chapter 2), is the integration of $(1+\varepsilon)$-approximation schemes, like those of Section 1.1, to the optimal algorithms presented in [8], in order to speed up the computation of the probabilistic histograms. Moreover, we bound the error of such approximation and we offer guarantees that the errors of our approximate probabilistic histograms are no more than $(1+\varepsilon)$ times the errors of the corresponding optimal probabilistic histograms.

1.2.1 Probabilistic Data Model

Let $\mathscr{U}$ be an ordered domain indexing the uncertain data presented in form of a probabilistic relation. Assuming, for simplicity, that the $\mathscr{U}$ is a set of integers, i.e., $\mathscr{U} = \{1,2,..,N\}$ $(|\mathscr{U}| = N)$. Each item $i \in \mathscr{U}$ is probabilistic data distribution (PDF), taking values from a ordered value domain $\mathscr{V}$, i.e., $\mathscr{V} = \{1,2,..,V\}$ $(|\mathscr{V}| = V)$.

The $\mathscr{U}$ is a probabilistic data sequence defining a set of ”possible worlds”. We can think ”the worlds” as a set of N-dimensional vectors f. Each single ”grounded” vector f is an instance of these worlds, i.e., it provides a value for each item in $\mathscr{U}$. Each value is taken from the value domain $\mathscr{V}$ discussed above.

For instance, $\mathscr{U}$ could correspond to a set of N temperature sensors placed at every mile of a highway, and in this case, f_i refers to the uncertain measure of a sensor placed at i^{th} mile of the highway on a particular day.

Let us define the meaning of a probabilistic data model. A probabilistic data model defines a probabilistic distribution over the ”possible worlds” mentioned before. There are many different probabilistic models, less or more descriptive, with a corresponding description size according to the description power of the model, i.e., the ability of the model to express more or less complex distributions.

The most descriptive model, and thus the most complex, is the model which is able to describe any possible N-dimensional probability distribution, e.g., by listing each possible world and its corresponding probability. Although, such a model is the most descriptive and thus the most accurate, this comes with a prohibited time requirements and complexity, since there is a huge number of possible worlds requiring the computation of an enormous number of parameters that is exponential in N.

For these reasons, a less descriptive model can be tolerable. By making certain independence assumptions, we can reduce the number of parameters needed from the model. If such correlations exist in the real data, this comes with a loss of accuracy, but if their impact is low, this can be tolerable in terms of having a less complex and time consuming probabilistic data model, without a large handicap in the quality of the model. Such a model is the Item-PDF Model (IPM) [8]:

Definition 1. *In the* Item-PDF Model (IPM)*, each item $i \in \mathscr{U}$ is assumed to behave independently of all others. A PDF X_i is provided to describe the distribution of item i. The probability of any given possible world f under this model can then be calculated directly as* $\Pr[f] = \prod_{i \in \mathscr{U}} \Pr[X_i = f_i]$.

The Item-PDF Model can capture the tuple- and attribute-level uncertainty. Since, the model provides a distribution of X conditioned on the value of i, it captures the

correlations between variable and i. Although, this model reduces the number of parameters needed to be calculated, i.e., $O(NV)$ parameters than the exponential (in N) number of the general model, there is still a need for more compact descriptions and thus the need of effective synopsis construction, like histograms, arises.

1.2.2 Probabilistic Histograms: Definition and Construction

Often, adjacent items in $\mathscr{U}$ are correlated, i.e., their distributions are quite similar, and thus, using buckets to group together items with similar distributions in order compute a compact representation of the original data is quite useful. If this "smoothness" property does indeed hold, the resulting histograms may effectively capture the original data distributions, and thus, the error of such approximation may be tolerable in terms of fast approximate answering.

In the probabilistic histogram construction process, the domain $\mathscr{U}$ is broken down into a number of consecutive disjoint ranges (the buckets) and for each bucket only a concise statistic, the bucket representative, is stored. The bucket boundaries and bucket representatives are chosen in terms of minimizing a given error metric. The authors of [8] propose a richer histogram representation, where the bucket representative is itself a (compact) distribution over $\mathscr{V}$, than earlier approaches [7], where each bucket representative is a single value chosen to minimize a given error metric. This richer representation leads to capture the underlying data semantics more accurately, that is, with a given space bound, the produced probabilistic histogram is optimal, i.e., the overall error function (with respect to the original IPM) is minimized.

Specifically, let $b = (s,e)$ be a bucket with left boundary denoted by s and right boundary denoted by e. Clearly, this bucket covers the range (s,e), i.e., it contains all item PDFs within the range ($|b| = e - s + 1$ item PDFs). In order to create a compact representation of the item PDFs within the bucket, i.e., $X_s, X_{s+1}, ..., X_e$, a bucket representative which is also itself a compact PDF over $\mathscr{V}$, $\hat{X}(b)$, is chosen in terms of minimizing the bucket error. We show this process in Figure 1.4.

At this point, we present a variety of PDF distance functions, i.e., functions that measure the dissimilarity of probability distributions, as given in [8]:

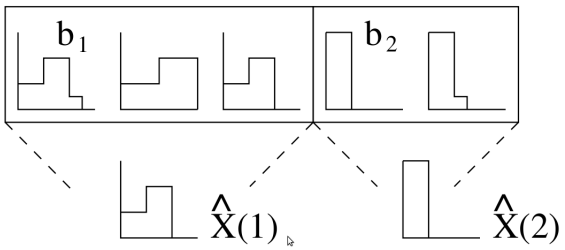


Figure 1.4: Summarizing a set of PDFs with compact PDF representatives.

- The *Variation Distance* (aka L_1) between two probability distributions over the same value domain $\mathscr{V}$ is the sum of absolute differences between the probabilities of each value. Formally, it is given by

$$d(X,Y) = \|X - Y\|_1 = \sum_{v \in \mathscr{V}} |\Pr[X = v] - \Pr[Y = v]|.$$

- The *Sum-squared error* (aka L_2^2) is similar to the Variation Distance, but takes the square of the difference of each pair of probabilities. It is defined by

$$d(X,Y) = \|X - Y\|_2^2 = \sum_{v \in \mathscr{V}} (\Pr[X = v] - \Pr[Y = v])^2.$$

- The *Kullback-Leibler divergence*, also known as the relative entropy, uses a more information theoretic approach to compare distributions. It is defined by

$$d(X,Y) = KL(X,Y) = \sum_{v \in \mathscr{V}} \Pr[X = v] \log_2 \frac{\Pr[X = v]}{\Pr[Y = v]}.$$

  Note that KL is not symmetric. It is natural to consider the second argument as the representative or approximation for the first argument.

- The *(Squared) Hellinger distance* is another commonly used measure of distribution similarity, given by

$$d(X,Y) = H^2(X,Y) = \sum_{v \in \mathscr{V}} \frac{(\Pr[X = v]^{\frac{1}{2}} - \Pr[Y = v]^{\frac{1}{2}})^2}{2}.$$

- The *Max-Error* measure (or L_∞) tracks the maximum difference between pairs of corresponding probabilities, and is given by

$$d(X,Y) = \|X,Y\|_\infty = \max_{v \in \mathscr{V}} |\Pr[X = v] - \Pr[Y = v]|$$

Here, in contrast to other metrics, the error of a histogram bucket is taken to be the maximum of this value over the different PDFs, rather than the sum.

With the above definitions we can express the overall error of the histogram as a summation or maximum of bucket errors. Since, the histogram splits the domain into B consecutive disjoint buckets which cover the entire domain $\mathscr{U}$, i.e., the k^{th} bucket $b_k = (s_k, e_k)$ covers the the item PDFs within the range (s_k, e_k) (where $s_1 = 1$, $e_B = N$, *and* $s_k = e_{k-1} + 1$, $\forall k \in \{2, .., B\}$), we can express the total error of the histogram with one of the two following formulas:
Summation of bucket errors:

$$S = \sum_{k=1}^{B} \sum_{i=s_k}^{e_k} d(\hat{X}(b_k), X_i) \tag{1.2}$$

Maximum of bucket errors:

$$M = \max_{k=1}^{B} \sum_{i=s_k}^{e_k} d(\hat{X}(b_k), X_i) \tag{1.3}$$

At this point, we formally present the definition of the probabilistic histogram construction problem and the two cases delivered from this definition as given in [8]:

Probabilistic Histogram Construction. Given a distance function $d()$, a space - complexity bound $\mathscr{S}$, and an input set of item PDFs $X_1, \ldots, X_N$ over $\mathscr{V}$, construct a probabilistic histogram of space complexity at most $\mathscr{S}$ which minimizes the histogram Sum- or Max-error under PDF distance $d()$.

- ***B*-bucket Case:** The histogram consists of exactly B buckets, each of which is represented by a detailed, V-term PDF over values $\mathscr{V}$. Such a representation makes sense when the size of the frequency-value domain, $\mathscr{V}$, is relatively small, and so each of these PDFs is quite small. In the B-bucket case, the overall space requirement of the probabilistic histogram is $\mathscr{S} = O(BV)$.

- ***T*-term Case:** When V is large, it makes more sense to try to find a T-term summary: if we represent each bucket-representative PDF by a set of piecewise

constant values (i.e., a conventional histogram) then the total description length is the total number T of such constant terms across all bucket representatives. The overall space requirement of the histogram in this case is $\mathscr{S} = O(T)$.

The T-term case of probabilistic histogram construction problem gereralizes the corresponding B-bucket case, since, assuming the same overall space ($T = BV$), in T-term case there is a larger search space of space-allotments, i.e., greater flexibility of space-allotments to each bucket. Thus, we will focus only on the T-Term case of probabilistic histogram construction problem.

1.2.3 Two-Step Dynamic Programming Schemes for Optimal Probabilistic Histogram Construction

The authors of [8] propose a general dynamic programming framework for optimal probabilistic histogram construction for a variety of error metrics such Variation Distance, Sum-Squared Error, Max-Error measure, and (Squared) Hellinger Distance.

Briefly, they broke down the general probabilistic histogram construction problem into two pieces, i.e., two sub-problems: the inner-part and the outer-part of the general problem, and solved them with dynamic programming schemes.

In the inner-part, the objective is, given a number of t terms for a bucket, to find the optimal bucket representative under a given error metric, i.e., to construct a t-term PDF over $\mathscr{V}$ that approximates the item PDFs within the bucket with the minimum possible error.

In the outer-part, the objective is, given the total number of terms for the histogram (T), to allocate them efficiently among the buckets and find the appropriate bucket boundaries in order to minimize the overall error of such approximation of the data input $\mathscr{U}$.

The Intra-Bucket Algorithm

Let $b = (s,e)$ be a bucket with left and right boundaries $s,e \in \mathscr{U}$ respectively. Clearly, the bucket b covers the interval (s,e) and thus contains $e-s+1$ item PDFs. The goal in the T-term case of probabilistic histogram construction, as already discussed, is to find a t-term piece-wise constant representative over $\mathscr{V}$, which approximates the $e-s+1$

item PDFs within the bucket b, with the minimum bucket error under a given error metric.

Moreover, let VALERR(b,v,w) denote the optimal error under a given error metric, in the approximation of the probability values in the range (v,w) (where $v,w \in \mathscr{V}$) within the bucket b, with a single value. In other words, the VALERR(b,v,w) function, computes a single constant representative which approximates the $(e-s+1)\times(w-v+1)$ probability values with the minimum possible error under a given error metric.

The computation of the VALERR(b,v,w) function, depends on the given error metric. Several solutions for a variety of error metrics are given in [8]. Here, we present only the proof for the sum-squared error metric (Lemma 1) as given in [8]. For completeness, the interested reader may refer to [8], for the proofs of other error metrics.

Let B-OPT$^b[v,T]$ denote the minimum possible error in the approximation of the range $(1,v)$ (where $v \in \mathscr{V}$) within the bucket b, using at most T terms. With respect to the principle of optimality, this error can be expressed with the following Dynamic Programming recurrence:

$$\text{B-OPT}^b[w,T] = \min_{1\leq v\leq w-1}\{\text{B-OPT}^b[v,T-1] + \text{VALERR}(b,v+1,w)\}. \tag{1.4}$$

The algorithm B-OPT$^b[w,t]$ (where $1 \leq w \leq V$) computes recursively the optimal error, in the representation (with t terms) of the first w values of item PDFs within the bucket b. Thus, B-OPT$^b[V,t]$ constructs a PDF with t terms, which approximates all item PDFs within the bucket b, with the minimum possible error. Notice that in the initial call B-OPT$^b[V,t]$, there is no benefit in assignment of $t > V$ terms, since a PDF has only V values.

This algorithm search exhaustively the range $(1,V)$ within a bucket in order to find the appropriate intra-bucket boundaries, by testing all possible choices to find the representation which has the minimum possible error. Moreover, for each intra-bucket, there is a space allotment of a single term. In cases where the size of the domain $\mathscr{V}$ is large, this approach may have prohibited time requirements. We inspired from this observation and we go further, reducing the searching space by settling an approximation scheme to the intra-bucket algorithm (Chapter 2).

The Inter-Bucket Algorithm

Let H-OPT$[m,T]$ denote the minimum possible error in the approximation up to domain value $m \in \mathscr{U}$, using at most T terms. With respect to the principle of optimality, this error can be expressed with the following Dynamic Programming recurrence (Max-Error case is handled similarly):

$$\text{H-OPT}[m,T] = \min_{1\leq k\leq m-1, 1\leq t\leq T-1}\{\text{H-OPT}[k,T-t] + \text{B-OPT}^{(k+1,m)}[V,t]\}. \quad (1.5)$$

Clearly, the H-OPT$[N,T]$ computes the optimal probabilistic histogram over the domain $\mathscr{U}$ using at most T terms. This algorithm search exhaustively the range $(1,N)$ in order to find the appropriate bucket boundaries and space allotments to each bucket, by testing all possible choices to find the representation which has the minimum possible error. In cases where the size of the domain $\mathscr{U}$ is large, this approach may have prohibited time requirements. As in the case of the intra-bucket algorithm, we reduce the searching space by settling an approximation scheme to the inter-bucket algorithm in order to produce near-optimal probabilistic histograms with reduced time requirements (Chapter 2).

Lemma 1. *The optimal cost for representing a range of values in a particular bucket under Sum-Squared Error in the T-term case can be found in constant time using $O(VN)$ precomputed values.*

Proof. Consider a range $r = (v,w)$ (where $v,w \in \mathscr{V}$) within a bucket $b = (s,e)$ that we wish to represent with a single value p. The contribution to the error is $\sum_{i=s}^{e}\sum_{j=v}^{w}(\Pr[X_i = j] - p)^2$. Differentiating with respect to p shows that this is minimized by setting

$$p = \bar{p} = \frac{\sum_{i=s}^{e}\sum_{j=v}^{w}\Pr[X_i = j]}{(e-s+1)(w-v+1)},$$

the average of the relevant probabilities. The cost VALERR(b,v,w) is then

$$\begin{aligned}&\sum_{i=s}^{e}\Big(\sum_{j=v}^{w}(\Pr[X_i = j])^2 - 2\bar{p}\Pr[X_i = j] + \bar{p}^2\Big)\\ =&\sum_{i=s}^{e}\Big(\sum_{j=v}^{w}(\Pr[X_i = j])^2\Big) - \bar{p}^2(e-s+1)(w-v+1) \quad .\end{aligned}$$

This cost can be computed quickly based on $O(VN)$ precomputed values. Define

$$A[e,w] = \sum_{i=1}^{e}\sum_{j=1}^{w} \Pr[X_i = j] \quad B[e,w] = \sum_{i=1}^{e}\sum_{j=1}^{w} (\Pr[X_i = j])^2.$$

$$\begin{aligned}
\text{Then} \quad & \bar{p}[(s,e),(v,w)] \cdot (e-s+1)(w-v+1) \\
& = (A[e,w] - A[s-1,w] - A[e,v-1] + A[s-1,v-1]) \\
\text{and} \quad & \sum_{i=s}^{e}\sum_{j=v}^{w} (\Pr[X_i = j])^2 \\
& = B[e,w] - B[s-1,w] - B[e,v-1] + B[s-1,v-1]).
\end{aligned}$$

From these, VALERR(b,v,w) can be computed in constant time. Last, note that this does indeed generate a valid PDF: clearly, each $\bar{p}$ is in $[0,1]$, since it is the mean of other probability values; and for a set of intervals $\mathscr{I} = \{(v,w)\}$ that partition $\mathscr{V}$, we have the cumulative probability,

$$\begin{aligned}
& \sum_{(v,w)\in\mathscr{I}} \sum_{j=v}^{w} \bar{p}[(s,e),(v,w)] \\
& \quad = \sum_{(v,w)\in\mathscr{I}} (w-v+1) \frac{\sum_{j=v}^{w}\sum_{i=s}^{e} \Pr[X_i = j]}{(e-s+1)(w-v+1)} \\
& \quad = \sum_{i=s}^{e}\sum_{j=v}^{w} \frac{\Pr[X_i = j]}{e-s+1} = \sum_{i=s}^{e} \frac{1}{e-s+1} = 1.
\end{aligned}$$

□

Concluding, we present the theorems for time complexity of optimal probabilistic histogram construction for a variety of error metrics as given in [8]:

Theorem 2. *The optimal T-term histogram of a probabilistic relation can be found in time $O(N^2T(\min(T,V)+V^2))$ under sum-squared error. The optimal B-bucket histogram of a probabilistic relation under the sum-squared error can be found in time $O(N(BN+V))$.*

Theorem 3. *The optimal T-term representation of a probabilistic relation under the variation distance can be found in time $O(N^2(T\min(T,V)+V^2\min(V,N)\log(VN)))$. The optimal B-bucket representation of a probabilistic relation under the variation distance can be found in time $O(N^2(B+\log(VN)))$.*

Theorem 4. *An ε-error (normalized) approximation to the optimal T-term probabilistic histogram of a probabilistic relation under variation distance can be found in time $O(N^2T^3V^2\varepsilon^{-1}\log(T\varepsilon^{-1}))$. An ε-error (normalized) approximation to the optimal B-bucket histogram can be found in time $O(N^2BV^4\varepsilon^{-1}\log(T\varepsilon^{-1}))$.*

Theorem 5. *The optimal T-term histogram of a probabilistic relation under squared Hellinger distance can be found in time $O(N^2T(\min(T,V)+V^2))$. The optimal B-bucket histogram can be found in time $O(N(BN+V))$.*

Theorem 6. *The optimal T-term probabilistic histogram of a probabilistic relation under max-error can be found in time $O(TVN^2)$. The optimal B-bucket histogram can be found in time $O(BVN^2)$.*

Chapter 2

Building Approximate Probabilistic Histograms

In this Chapter, we present a novel approximate algorithm for the probabilistic histogram construction problem. The optimal dynamic programming algorithm, of Section 1.2, has high computational cost. Our purpose is to produce efficiently near-optimal probabilistic histograms with much lower computational cost than the cost of the optimal algorithm, and provide guarantees to the precision (quality) of the produced histograms. The speed up is obtained, because our approximate algorithm explores only a small set of sub-problems in the DP framework, with respect to the underlying approximation scheme. Our approximation scheme provides a useful trade-off between the precision of the histogram and the time cost of construction.

In the T-term case of probabilistic histogram construction, we have a number of available terms (which user specifies) for the histogram. Our algorithm tries to allocate them efficiently among the buckets. Specifically, the algorithm tries to find the appropriate, with respect to the underlying approximation scheme, bucket boundaries and term assignments to each bucket, in order to minimize (approximately) a given error metric. We try to find the approximately best choice of bucket boundaries and term assignments to each bucket, i.e., the number of terms to construct the bucket representative PDF, in order to minimize (approximately) the overall error. The error of the histogram (overall error) can be the sum of bucket errors or the maximum of bucket

errors, depending on the given error metric.

Our algorithm consists of two nested approximate dynamic programming algorithms: The inter-bucket algorithm and the intra-bucket algorithm. The inter-bucket algorithm (**inter**) tries to find the appropriate bucket boundaries, with respect to the underlying approximation scheme, and term assignments to each bucket. The intra-bucket algorithm (**intra**) is called from the inter-bucket algorithm with any possible (with respect to the underlying approximation scheme) combination of bucket boundaries and term assignments. For each call, the intra-bucket algorithm constructs the bucket representative PDF, and returns the bucket error to the inter-bucket algorithm. Then, after all appropriate choices have been tested, we have the final histogram, i.e., a sequence of bucket PDFs which approximate the underlying item PDFs.

2.1 The Approximate Inter-Bucket Algorithm

We adjusted the AHIST-S algorithm (discussed in subsection 1.1.2), to the two- dimensional case of probabilistic data. Specifically, the AHIST-S algorithm, builds approximate one-dimensional histograms over deterministic data. We keep the basic ideas of that work and we create an algorithm which builds approximate probabilistic histograms over probabilistic data. Let T be the number of available terms for histogram construction, and $\mathscr{U} = \{1,2,...,N\}$ ($|\mathscr{U}| = N$) be the set of items on which we want to build the histogram. Assume that item PDFs have a value domain $\mathscr{V}$ ($|\mathscr{V}| = V$). Also, let $inter[m,t]$ ($m \in \mathscr{U}$) be the error in approximation of first m items with t terms. Our algorithm guarantees that $inter[m,t]$ is an $(1+\varepsilon)^2-$approximation of the optimal solution, i.e., $inter[m,t] \leq (1+\varepsilon)^2$ H-OPT$[m,t]$ ($\varepsilon \leq 1$), where with H-OPT$[m,t]$ we denote the optimal solution of Section 1.2. If we set the approximation scheme only to the inter-bucket algorithm (if we don't set the approximation both to inter- and intra-bucket algorithms), we have $(1+\varepsilon)-$approximation of the optimal solution, i.e., $inter[m,t] \leq (1+\varepsilon)$ H-OPT$[m,t]$ ($\varepsilon \leq 1$).

Recall the Dynamic Programming table of optimal algorithm. Our approximate algorithm builds a similar table, but it chooses to remember/store only a small set of solutions of sub-problems, which stores in appropriate lists, reducing the searching

space needed from the algorithm. These solutions are the necessary/important information, needed to produce an $(1+\varepsilon)^2-$approximation of the optimal solution.

Our algorithm maintains T-1 lists $Q[t]$ $(t \leq T-1)$ of intervals. The intervals, of each list, cover the entire domain $\mathscr{U} = \{1,2,..,N\}$ and are disjoint. Each record within a list $Q[t]$ has the form: $(a_i^{(t)}, b_i^{(t)}, err_i^{(t)})$, where $a_i^{(t)} \in \mathscr{U}$ is the start of the i^{th} interval in the t^{th} list, and $b_i^{(t)} \in \mathscr{U}$ is the end. The $err_i^{(t)}$ is the error record of the i^{th} interval in the t^{th} list, which is fixed to be equal with the error in approximation (with t terms) up to domain value denoted by the end of the interval, i.e., $err_i^{(t)} = inter[b_i^{(t)}, t]$. The intervals are stored by a $1+\delta$ factor which will be fixed in lemma 8, i.e., the error up to domain value denoted by the end of an interval is not more than $(1+\delta)$ times the error up to domain value denoted by the start of the interval. Thus, a list $Q[t]$ has the following form: $Q[t] = \{(a_1^{(t)}, b_1^{(t)}, err_1^{(t)}), (a_2^{(t)}, b_2^{(t)}, err_2^{(t)}), ..., (a_l^{(t)}, b_l^{(t)}, err_l^{(t)})\}$, where $a_1^{(t)} = 1, b_i^{(t)} + 1 = a_{i+1}^{(t)}, b_l^{(t)} = N$.

Each list $Q[t]$ approximates the error curve $inter[i,t](i = 1..N)$ with a staircase function with l terms (with l error values), where l is the number of intervals stored in list Q[t]. Notice that the number of intervals in a list, i.e., l, depends on the error curve we approximating, and is known only after execution. In most cases $l \ll N$, since the intervals are stored by a $1+\delta$ factor, suggesting that the searching space of our approximate algorithm, is highly reduced in contrast with the optimal algorithm.

The algorithm inspects the items in increasing order. Let $m \leq N$ be the current item, and $t \leq T$ be the current number of terms. The error in approximation of first m items with t terms is computed with respect to the following recursion (initial call is inter[N,T], Max-Error case is handled similarly):

$$inter[m,t] = \min_{\substack{b^{(t-t')} \in Q[t-t'] \\ 1 \leq t' \leq \mathbf{min(V,\, t-1)}}} \{inter[b^{(t-t')}, t-t'] + intra^{(b^{(t-t')}+1, m)}[V, t']\}$$

In previous recursion, $b^{(t-t')}$ stands for the ends of intervals in $Q[t-t']$ $(1 \leq t' \leq min(V,\ t-1))$ lists. It is remarkable that our approximate algorithm needs to examine only the buckets starting from the ends of the intervals stored in the lists, against the optimal algorithm which examines every possible bucket boundaries. At each step, and for each interval $(a_i^{(t)}, b_i^{(t)})$, the algorithm guarantees that: $inter[b_i^{(t)}, t] \leq (1+\delta)\ inter[a_i^{(t)}, t]$. Thus, when a new entry $(inter[m,t])$ has been computed, the algorithm

checks if $inter[m,t] \leq (1+\delta)\ inter[a_l^{(t)},t]$. If this inequality doesn't hold, it creates a new interval $(a_{l+1}^{(t)}, b_{l+1}^{(t)}, err_{l+1}^{(t)})$, where $a_{l+1}^{(t)} = b_{l+1}^{(t)} = m$ and $err_{l+1}^{(t)} = inter[m,t]$. Otherwise, the algorithm change the last interval in list $Q[t]$, i.e., $(a_l^{(t)}, b_l^{(t)}, err_l^{(t)})$, to be $(a_l^{(t)}, m, err_l^{(t)})$, where $err_l^{(t)} = inter[m,t]$. We present the pseudocode of inter-bucket algorithm in Figure 2.1.

2.2 The Approximate Intra-Bucket Algorithm

The algorithm $intra^{(b)}[w,t]$ (where $1 \leq w \leq V$) computes recursively an approximation of the optimal error, in the representation (with t terms) of the first w values of item PDFs within the bucket b. Thus, $intra^{(b)}[V,t]$ constructs a PDF with t terms, which approximates all item PDFs within the bucket b. Notice that in the initial call $(intra^{(b)}[V,t])$, there is no a benefit in assignment of $t > V$ terms, since a PDF has only V values. The intra-bucket algorithm is an $(1+\varepsilon)$-approximate algorithm with the same logic as the inter-bucket algorithm, i.e., $intra^{(b)}[w,t] \leq (1+\varepsilon)\,\text{B-OPT}^{(b)}[w,t]\,(\varepsilon \leq 1)$, where with $\text{B-OPT}^{(b)}[w,t]$ we denote the optimal solution of Section 1.2. It maintains $min(V-1, t-1)$ list of intervals with the same logic as before. Each list Q[k] corresponds to a staircase function which approximates the whole error function $intra^{(b)}[w,k]$ $(w = 1..V)$ with $l \ll V$ error values. In the DP table of the approximate intra-bucket algorithm, the computation of an entry $intra^{(b)}[w,t]$ follows the formula:

$$intra^{(b)}[w,t] = \min_{b^{(t-1)} \in Q[t-1]} \{intra^{(b)}[b^{(t-1)}, t-1] + \text{VALERR}(b, b^{(t-1)}+1, w)\}$$

The previous recurrence is similar with the corresponding recurrence of optimal intra-bucket algorithm. The only change is the reduced searching space, i.e., our approximate algorithm examines only the intra-buckets starting from the ends of the intervals, in contrast with the optimal algorithm which examines every possible bucket boundaries. We remind that the function $\text{VALERR}(b,u,v)$ computes the optimal error in the representation of the range (u,v) (where u,v $\in \mathcal{V}$) within the bucket b, with one term (with one value p). The computation of $\text{VALERR}()$ depends on the given error metric [8]. Algorithm's pseudocode is in Figure 2.2.

Lemma 7. *The algorithm $intra^{(b)}[w,T]$ produce an $(1+\varepsilon)$-approximation of the optimal solution, i.e., $intra^{(b)}[w,T] \leq (1+\varepsilon)\,\text{B-OPT}^{(b)}[w,T]$ ($\varepsilon \leq 1$).*

Proof. We will prove by induction that: $intra^{(b)}[w,T] \leq (1+\delta)^{T-1}\text{B-OPT}^{(b)}[w,T]$. The base case is $T=1$. In this case we have: $intra^{(b)}[w,1] = \text{B-OPT}^{(b)}[w,1] = \text{VALERR}(b,1,w)$. We assume that the statement holds for $t<T$, and we will look if it holds for $t=T$. Let $(v,w]$ be the last bucket which has been chosen by optimal algorithm ($\text{B-OPT}^{(b)}[w,T]$) to separate the interval $[1,w]$. Assume that (s,e) is the interval of $(T-1)^{th}$ list, which has been created from $intra^{(b)}[w,T]$ algorithm and contains the v: $s \leq v \leq e < w$. With respect to the above assumptions, we have:

$$
\begin{aligned}
&\text{B-OPT}^{(b)}[w,T] = \text{B-OPT}^{(b)}[v,T-1] + \text{VALERR}(b,v+1,w)\\
&\geq \text{B-OPT}^{(b)}[s,T-1] + \text{VALERR}(b,v+1,w)\\
&(\text{B-OPT}^{(b)}[.\,,\,T-1] \text{ is monotone})\\
&\geq \text{B-OPT}^{(b)}[s,T-1] + \text{VALERR}(b,e+1,w)\\
&(v \leq e < w \text{ and } \text{VALERR}(b,f,w) \text{ is non-increasing as } f \text{ increases})\\
&\geq \frac{1}{(1+\delta)^{T-2}}(intra^{(b)}[s,T-1]) + \text{VALERR}(b,e+1,w)\\
&(\text{From the induction assumption})\\
&\geq \frac{1}{(1+\delta)^{T-2}}\Big(\frac{1}{1+\delta}intra^{(b)}[e,T-1] + \text{VALERR}(b,e+1,w)\Big)
\end{aligned}
$$

The last inequality holds, because the intra-bucket algorithm guarantees that: $intra^{(b)}[e,T-1] \leq (1+\delta)\,intra^{(b)}[s,T-1]$. So, we have that:

$$
\begin{aligned}
\text{B-OPT}^{(b)}[w,T] &\geq \frac{1}{(1+\delta)^{T-2}}\Big(\frac{1}{1+\delta}intra^{(b)}[e,T-1] + \text{VALERR}(b,e+1,w)\Big)\\
&\geq \frac{1}{(1+\delta)^{T-1}}\,(intra^{(b)}[e,T-1] + \text{VALERR}(b,e+1,w))\\
&\geq \frac{1}{(1+\delta)^{T-1}}intra^{(b)}[w,T]
\end{aligned}
$$

The last inequality holds, since, according to the intra-bucket algorithm, we have: $intra^{(b)}[w,T] = \min_{e \in Q[T-1]}\{intra^{(b)}[e,T-1] + \text{VALERR}(b,e+1,w)\}$. By setting $\delta = \frac{\varepsilon}{2T}$, the approximation factor is at most $(1+\frac{\varepsilon}{2T})^{T-1}$, which is at most $1+\varepsilon$ for small ε ($\varepsilon \leq 1$). This proves the lemma. □

Lemma 8. *The inter-bucket algorithm produce an* $(1+\varepsilon)^2$*-approximation of the optimal solution, i.e.,* $inter[m,T] \leq (1+\varepsilon)^2$ H-OPT$[m,T]$ *(*$\varepsilon \leq 1$*).*

Proof. **Let** ε_1 **be the aprroximation factor of intra-bucket algorithm. In Lemma 7 we proved that:** $\mathbf{intra^{(b)}[w,t] \leq (1+\varepsilon_1)}$ B-OPT$^{(\mathbf{b})}[\mathbf{w},\mathbf{t}]$**.** We will prove by induction that: $inter[m,T] \leq (1+\varepsilon_1)(1+\delta)^{T-1}$H-OPT$[m,T]$. The base case is $T=1$. In this case we have: $inter[m,1] =$ H-OPT$[m,1] =$ VALERR$((1,m),1,V)$. Assume that the statement holds for $t<T$. We will look if it holds for $t=T$. Let $(k,m]$ be the last bucket, chosen by optimal algorithm to separate the interval $[1,m]$ and that assigned t terms. Also let (s,e) be the interval of $(T-t)^{th}$ list which has created by inter-bucket algorithm and contains the k, i.e., $s \leq k \leq e < m$. With respect to the above assumptions, we have:

$$\text{H-OPT}[m,T] = \text{H-OPT}[k,T-t] + \text{B-OPT}^{(k+1,m)}[V,t]$$

$$\geq \text{H-OPT}[s,T-t] + \text{B-OPT}^{(k+1,m)}[V,t] \text{ (H-OPT}[.\,,\,T-t] \text{ is monotone)}$$

$$\geq \text{H-OPT}[s,T-t] + \text{B-OPT}^{(e+1,m)}[V,t]$$

($k \leq e < m$ and B-OPT$^{(f,m)}[V,t]$ is non-increasing as f increases)

$$\geq \frac{1}{(1+\varepsilon_1)(1+\delta)^{T-t-1}}(inter[s,T-t]) + \text{B-OPT}^{(e+1,m)}[V,t]$$

(From the induction assumption)

$$\geq \frac{1}{(1+\varepsilon_1)(1+\delta)^{T-t-1}}(\frac{1}{1+\delta}inter[e,T-t]) + \text{B-OPT}^{(e+1,m)}[V,t]$$

(The inter-bucket algorithm guarantees that:

$inter[e,T-t]) \leq (1+\delta)inter[s,T-t])$

$$\geq \frac{1}{(1+\varepsilon_1)(1+\delta)^{T-t}}(inter[e,T-t]) + \text{B-OPT}^{(e+1,m)}[V,t]$$

$$\geq \frac{1}{(1+\varepsilon_1)(1+\delta)^{T-t}}(inter[e,T-t]) + \frac{1}{(1+\varepsilon_1)}(intra^{(e+1,m)}[V,t])$$

(From lemma 7)

$$\geq \frac{1}{(1+\varepsilon_1)(1+\delta)^{T-1}}\big(inter[e,T-t] + intra^{(e+1,m)}[V,t]\big)$$

($T \geq t$ and $t \geq 1$)

$$\geq \frac{1}{(1+\varepsilon_1)(1+\delta)^{T-1}}inter[m,T]$$

The last inequality holds, since, according to the inter-bucket algorithm :
$inter[m,T] = \min_{e \in Q[T-t]} \{inter[e,T-t] + intra^{(e+1,m)}[V,t]\}$ $(1 \leq t \leq min\{V,T-1\})$.
Setting $\delta = \frac{\varepsilon_2}{2T}$, the factor $(1+\frac{\varepsilon_2}{2T})^{T-1}$ is at most $1+\varepsilon_2$ for small ε_2 ($\varepsilon_2 \leq 1$). Thus, $inter[m,T] \leq (1+\varepsilon_1)(1+\varepsilon_2)$H-OPT$[m,T]$, and if we fix $\varepsilon_1 = \varepsilon_2 = \varepsilon \leq 1$, we have an $(1+\varepsilon)^2$-approximation of optimal solution. This proves the lemma.

□

Lemma 9. *Let $\tau_1 = min\{T\varepsilon^{-1}\ln(N),\ N\}$. Each list created by inter-bucket algorithm has $O(\tau_1)$ size, under sum-squared error metric.*

Proof. Consider a list $Q[T] = \{[a_1,b_1,inter[b_1,T]],...,[a_l,b_l,inter[b_l,T]],[a_{l+1},b_{l+1},inter[b_{l+1},T]]\}$, which contains $l+1$ intervals, where $a_1 = 1$ and $a_i = b_{i-1}+1 (2 \leq i \leq l+1)$. With respect to the inter-bucket algorithm, we have:

$$\begin{aligned} inter[b_1+1,T] = inter[a_2,T] &> (1+\delta)inter[a_1,T] \\ inter[b_2+1,T] = inter[a_3,T] &> (1+\delta)inter[a_2,T] \\ \vdots &> \vdots \\ inter[b_l+1,T] = inter[a_{l+1},T] &> (1+\delta)inter[a_l,T] \end{aligned}$$

$$inter[a_{l+1},T] > (1+\delta)^{l-1} inter[a_2,T]$$

Notice that $inter[a_2,T]$ cannot be zero, because, in that case, the first inequality cannot be satisfied.

Let the minimum possible non-zero bucket error be $0 <$ B-OPT$^b[V, min\{V,T\}-k] =$ $\mathbf{x}$ $(k \geq 0)$, measured with sum-squared error metric. The k defined, as the minimum integer which gives a non-zero error. It is clear that this is also the minimum possible error of the approximate algorithm $(inter[N,T])$, since $inter[N,T] \geq$ H-OPT$[N,T]$. **Suppose, for contradiction, that: $\mathbf{l > 1 + 2\delta^{-1} ln(x^{-1} N V)}$ ★**. The maximum error of inter-bucket algorithm is when it approximates all item PDFs, with one term. In this case, we have:

$$inter[N,1] = \text{VALERR}((1,N),1,V) = \sum_{i=1}^{N}\sum_{j=1}^{V}(Pr(X_i = j) - p)^2 \qquad (2.1)$$

$$p = \overline{p} = \frac{\sum_{i=1}^{N}\sum_{j=1}^{V} Pr(X_i = j)}{(N-1+1)(V-1+1)} \tag{2.2}$$

$\overset{(2.1),(2.2)}{\Longrightarrow} 0 < \text{VALERR}((1,N),1,V) \leq NV \Rightarrow NV \geq \text{VALERR}((1,N),1,V)$
$\geq inter[a_{l+1},T] \geq (1+\delta)^{l-1} inter[a_2,T] \Rightarrow NV \geq (1+\delta)^{l-1} inter[a_2,t]$
$\geq \mathbf{x}(1+\delta)^{l-1} \overset{\bigstar}{\Rightarrow}$

$$\mathbf{x}^{-1} NV \geq ((1+\delta)^{\frac{2}{\delta}})^{ln(\mathbf{x}^{-1} NV)} \tag{2.3}$$

We have that: $(1+\delta)^{\frac{2}{\delta}} > e\,(0 < \delta < 1)$, so, from the above equation:
$\mathbf{x}^{-1} NV > e^{ln(\mathbf{x}^{-1} NV)}$, which is a **contradiction**. So, we have:
$l \leq 1 + 2\,\delta^{-1} ln(\mathbf{x}^{-1} NV) = 1 + 2\,\delta^{-1}(ln(\mathbf{x}^{-1} V) + ln(N)) \overset{\blacklozenge}{\Rightarrow}$
$\Rightarrow l = O(\delta^{-1} ln(N)) \underset{\delta = \frac{\varepsilon}{2T}}{\Rightarrow} \underset{(lemma\,8)}{} l = O(T\varepsilon^{-1} ln(N))$. This proves the lemma.

♦ Assuming that, asymptotically, the dominant factor is N. □

Lemma 10. *Let $\tau_2 = min\{T\varepsilon^{-1}\ln(N),\, V\}$. Each list created by intra-bucket algorithm has $O(\tau_2)$ size under sum-squared error metric.*

Proof. The proof is quite similar with the proof in Lemma 9. □

Lemma 11. *Let $\tau_3 = min\{T\varepsilon^{-1} ln(\mathbf{x}^{-1}/2),\, N\}$, where $\mathbf{x} > 0$ is a constant. Each list created by inter-bucket algorithm has $O(\tau_3)$ size, under Max-Error metric.*

Proof. Consider a list $Q[T] = \{[a_1,b_1,inter[b_1,T]],...,[a_l,b_l,inter[b_l,T]],[a_{l+1},b_{l+1},inter[b_{l+1},T]]\}$, which contains $l+1$ intervals, where $a_1 = 1$ and $a_i = b_{i-1} + 1 (2 \leq i \leq l+1)$. With respect to the inter-bucket algorithm, we have:

$$\begin{gathered} inter[b_1+1,T] = inter[a_2,T] > (1+\delta) inter[a_1,T] \\ inter[b_2+1,T] = inter[a_3,T] > (1+\delta) inter[a_2,T] \\ \vdots > \vdots \\ inter[b_l+1,T] = inter[a_{l+1},T] > (1+\delta) inter[a_l,T] \end{gathered}$$

$$inter[a_{l+1},T] > (1+\delta)^{l-1} inter[a_2,T]$$

Notice that $inter[a_2,T]$ cannot be zero, because, in that case, the first inequality cannot be satisfied.

Let the minimum possible non-zero bucket error be $0 < \text{B-OPT}^b[V, min\{V,T\}-k] = \mathbf{x}$ $(k \geq 0)$, measured with max-error metric. The k defined, as the minimum integer which gives a non-zero error. It is clear that this is also the minimum possible error of the approximate algorithm $(inter[N,T])$, since $inter[N,T] \geq \text{H-OPT}[N,T]$. **Suppose, for contradiction, that: $\mathbf{l > 1 + 2\delta^{-1}\ln(x^{-1}/2)}$** $\bigstar$. Directly, from the max-error metric properties, the maximum error of inter-bucket algorithm is $1/2$. In this case, we have:

$$1/2 \geq inter[a_{l+1},T] \geq (1+\delta)^{l-1} inter[a_2,T]$$
$$\Rightarrow 1/2 \geq (1+\delta)^{l-1} inter[a_2,t] \geq \mathbf{x}(1+\delta)^{l-1}$$

$$\overset{\bigstar}{\Rightarrow} \mathbf{x}^{-1}/2 \geq ((1+\delta)^{\frac{2}{\delta}})^{ln(\mathbf{x}^{-1}/2)} \tag{2.4}$$

We have that: $(1+\delta)^{\frac{2}{\delta}} > e\,(0<\delta<1)$, so, from the above equation:
$\mathbf{x}^{-1}/2 > e^{\ln(\mathbf{x}^{-1}/2)}$, which is a **contradiction**. So, we have:
$l \leq 1 + 2\delta^{-1}\ln(\mathbf{x}^{-1}/2) \underset{\delta=\frac{\varepsilon}{2T}\ (lemma\ 8)}{\Rightarrow} l = O(T\varepsilon^{-1} ln(\mathbf{x}^{-1}/2))$. This proves the lemma. □

In the following lemmas we don't give the proofs, since the error bounds (minimum and maximum error) are the same with those in sum-squared error metric, and so, the proofs are quite similar.

Lemma 12. *Let $\tau_1 = min\{T\varepsilon^{-1}\ln(N), N\}$. Each list created by inter-bucket algorithm has $O(\tau_1)$ size, under squared Hellinger distance.*

Lemma 13. *Let $\tau_2 = min\{T\varepsilon^{-1}\ln(N), V\}$. Each list created by intra-bucket algorithm has $O(\tau_2)$ size, under squared Hellinger distance.*

Lemma 14. *Let $\tau_1 = min\{T\varepsilon^{-1}\ln(N), N\}$. Each list created by inter-bucket algorithm has $O(\tau_1)$ size, under L1 Error metric.*

Lemma 15. *Let $\tau_2 = min\{T\varepsilon^{-1}\ln(N), V\}$. Each list created by intra-bucket algorithm has $O(\tau_2)$ size, under L1 Error metric.*

We are ready now to give the time complexity of our approximate algorithm for a variety of error metrics.

Theorem 16. *An $(1+\varepsilon)^2$-approximation, i.e., when we set the approximation scheme both to inter- and intra-bucket algorithms, of optimal T-term histogram of a probabilistic relation under sum-squared error, can be found in time $O(NTmin\{T,V\}\tau_1 + NT\tau_1 Vmin\{T,V\}\tau_2)$, where τ_1, τ_2 are the list sizes, under sum-squared error, of the approximate inter- and intra-bucket algorithms, respectively.*

Proof. The time complexity of our approximate Dynamic Programming algorithms depends on the time complexity of the core component, i.e., the time complexity of VALERR function. The VALERR function can be computed in constant time, under sum-squared error metric [8]. Our approximate inter-bucket algorithm builds an approximate $N \times T$ Dynamic Programming table. For each entry in this table, there is a inter-searching space of size $O(min\{T,V\}\ \tau_1)$ (τ_1 denotes the list size bound of inter-bucket algorithm). Thus, our approximate Dynamic Programming scheme needs $O(NT\,min\{T,V\}\tau_1)$ time to compare all possible choices (with respect to the underlying approximation scheme) of bucket boundaries and term assignments. The rest time is consumed by the intra-bucket algorithm's calls. Clearly, each call of intra-bucket algorithm has $O(V\,min\{T,V\}\ \tau_2)$ time complexity (τ_2 denotes the list size bound of intra-bucket algorithm). We need to call intra-bucket algorithm only with a space allotment of $min\{T,V\}$ terms, since there is no benefit in assignment of $t > V$ terms for a bucket (a PDF has only V terms), and additionally, the intra-bucket algorithm itself performs Dynamic Programming, so the calculation of $intra^{(b)}[V,t]$ also generates the solutions $intra^{(b)}[V,t']$ $(1 \leq t' < t)$. Naively, there is a need to carry out $O(N\,T\ \tau_1)$ evaluations of intra-bucket algorithm totally, i.e., we need to evaluate the intra-bucket algorithm for each bucket, starting at the end point of an interval in a list (plus one) and ending at the current item $i \in \mathscr{U}$. Actually, we carry out significantly fewer evaluations than $O(N\,T\ \tau_1)$ and of course fewer than the optimal algorithm which needs $O(N^2)$ evaluations. This holds, because we need to evaluate the intra-bucket algorithm only for the buckets starting at the distinct ends of the intervals within the T-1 lists. These distinct ends are obviously less than N. Hence, combining the above, the overall time complexity to build an approximate probabilistic histogram under sum-squared

error metric is $O(N\,T\,min\{T,V\}\,\tau_1\,+\,N\,T\,\tau_1\,V\,min\{T,V\}\,\tau_2)$.

□

Theorem 17. *An $(1+\varepsilon)$-approximation, i.e., when we set the approximation scheme only to the inter-bucket algorithm, of optimal T-term histogram of a probabilistic relation under sum-squared error, can be found in time $O(NTmin\{T,V\}\tau_1 + NT\tau_1V^2min\{T,V\})$, where τ_1, is the list size, under sum-squared error, of the approximate inter-bucket algorithm.*

Proof. The proof is quite similar with the proof of theorem 16. The only difference is that an intra-bucket call has $O(V^2min\{T,V\})$ time complexity, since we don't set the approximation scheme to the intra-bucket algorithm. □

Theorem 18. *An $(1+\varepsilon)^2$-approximation, i.e., when we set the approximation scheme both to inter- and intra-bucket algorithms, of optimal T-term histogram of a probabilistic relation under (squared) Hellinger distance, can be found in time $O(NT\min(T,V)\tau_1 + NT\tau_1Vmin\{T,V\}\tau_2)$, where τ_1, τ_2 are the list sizes, under (squared) Hellinger distance, of the approximate inter- and intra-bucket algorithms, respectively.*

Proof. The proof is quite similar with Theorem 16. □

Theorem 19. *An $(1+\varepsilon)$-approximation, i.e., when we set the approximation scheme only to the inter-bucket algorithm, of optimal T-term histogram of a probabilistic relation under under (squared) Hellinger distance, can be found in time $O(NTmin\{T,V\}\tau_1 + NT\tau_1V^2min\{T,V\})$, where τ_1, is the list size, under (squared) Hellinger distance, of the approximate inter-bucket algorithm.*

Proof. The proof is quite similar with the proof of theorem 18. The only difference is that an intra-bucket call has $O(V^2min\{T,V\})$ time complexity, since we don't set the approximation scheme to the intra-bucket algorithm. □

Theorem 20. *An $(1+\varepsilon)^2$-approximation, i.e., when we set the approximation scheme both to inter- and intra-bucket algorithms, of optimal T-term histogram of a probabilistic relation under L1 error metric, can be found in time $O(NT\min(T,V)\tau_1 + NT\tau_1V\tau_2min\{V,N\}log(VN))$, where τ_1, τ_2 are the list sizes, under L1 error metric, of the approximate inter- and intra-bucket algorithms, respectively.*

Proof. The proof is quite similar with the corresponding proof for L1 error metric in [8]. The only difference is the reduced searching space of our approximate inter- and intra-bucket algorithms, depending on the list sizes under L1 error metric. □

Theorem 21. *An $(1+\varepsilon)$-approximation, i.e., when we set the approximation scheme only to the inter-bucket algorithm, of optimal T-term histogram of a probabilistic relation under under L1 error metric, can be found in time $O(NTmin\{T,V\}\tau_1 + NT\tau_1 V^2 min\{V,N\}log(VN))$, where τ_1, is the list size, under L1 error metric, of the approximate inter-bucket algorithm.*

Proof. The proof is quite similar with the proof of theorem 20. The only difference is that we don't set the approximation scheme to the intra-bucket algorithm, and thus, its searching space is the same with the searching space of optimal intra-bucket algorithm. □

Theorem 22. *An $(1+\varepsilon)$-approximation of optimal T-term histogram of a probabilistic relation under Max-Error, can be found in time $O(T\,N\,min\{T,V\}\,\tau_3\,V)$, where τ_3 is the list size, under max-error, of the approximate inter-bucket algorithm.*

Proof. The proof is quite similar with the corresponding proof in [8]. The only difference is the reduced searching space of our approximate inter-bucket algorithm, depending on the list sizes under the max-error measure. □

Set up T lists Q[t] to store the intervals (initially are empty)
for t = T Down to 1 **do**
  $inter[1,t] \leftarrow intra^{(1,1)}[V, min(t,V)])$
  Insert a new interval $(1,1,inter[1,t])$ to $Q[t]$
end for
for i = 2 up to N **do**
  for t = T Down to 1 **do**
    if $t == 1$ **then**
      $inter[i,1] \leftarrow$ VALERR$((1,i),1,V)$
      //a_l is the start index of the last interval in Q[1]
      //b_l is the end index of the last interval in Q[1]
      if $inter[i,1] > (1+\delta)inter[a_l,1]$ **then**
        $a_{l+1} = b_{l+1} = i$
        Insert a new interval $[a_{l+1}, b_{l+1}, inter[i,1]]$ to $Q[1]$
      else
        Change the last interval to be: $[a_l, i, inter[i,1]]$
      end if
    else
      $inter[i,t] \leftarrow \infty$
      for k = 1 up to $min(V, t-1)$ **do**
        for e = each end point b of interval list Q[t-k] **do**
          $inter[i,t] \leftarrow \min(inter[i,t],\ inter[e,t-k] + intra^{(e+1,i)}[V,k])$
          //a_l is the start index of the last interval in Q[t]
          //b_l is the end index of the last interval in Q[t]
          if $inter[i,t] > (1+\delta)inter[a_l,t]$ **then**
            $a_{l+1} = b_{l+1} = i$
            Insert a new interval $[a_{l+1}, b_{l+1}, inter[i,t]]$ to $Q[t]$
          else
            Change the last interval to be: $[a_l, i, inter[i,t]]$
          end if
        end for
      end for
    end if
  end for
end for

Figure 2.1: The Inter-Bucket Algorithm: $inter[N,T]$

Set up T lists Q[t] to store the intervals (initially are empty)
for t = T Down to 1 **do**
  $intra^{(b)}[1,t] = \text{VALERR}(b,1,1)$
  Insert a new interval $[1,1,intra^{(b)}[1,t]]$ to $Q[t]$
end for
for i = 2 up to V **do**
  for t = T Down to 1 **do**
    if $t == 1$ **then**
      $intra^{(b)}[i,1] \leftarrow \text{VALERR}(b,1,i)$
      //a_l is the start index of the last interval in Q[1]
      //b_l is the end index of the last interval in Q[1]
      if $intra^{(b)}[i,1] > (1+\delta)intra^{(b)}[a_l,1]$ **then**
        $a_{l+1} = b_{l+1} = i$
        Insert a new interval $[a_{l+1},b_{l+1},intra^{(b)}[i,1]]$ to $Q[1]$
      else
        Change the last interval to be: $[a_l,i,intra^{(b)}[i,1]]$
      end if
    else
      $intra^{(b)}[i,t] \leftarrow \infty$
      for e = each end point in interval list Q[t-1] **do**
        $intra^{(b)}[i,t] \leftarrow \min(intra^{(b)}[i,t],\ intra^{(b)}[e,t-1] + \text{VALERR}(b,e+1,i))$
        //a_l is the start index of the last interval in Q[t]
        //b_l is the end index of the last interval in Q[t]
        if $intra^{(b)}[i,t] > (1+\delta)intra^{(b)}[a_l,t]$ **then**
          $a_{l+1} = b_{l+1} = i$
          Insert a new interval $[a_{l+1},b_{l+1},intra^{(b)}[i,t]]$ to $Q[t]$
        else
          Change the last interval to be: $[a_l,i,intra^{(b)}[i,t]]$
        end if
      end for
    end if
  end for
end for

Figure 2.2: The Intra-Bucket Algorithm: $intra^{(b)}[V,T]$

2.3 Experimental Study

We implemented our algorithms for building approximate probabilistic histograms in C, and carried out a set of experiments to compare the quality and scalability of our results against the optimal (in terms of overall error) probabilistic histograms. The experiments were performed on a server equipped with 4 CPUs clocked at 2.3 GHz, and 4GB of RAM. Each experiment was run on a single CPU.

Data Set. We experimented using a real data set. The real data set came from the MystiQ project[1] which includes approximately $127,000$ tuples describing $27,700$ distinct items. These correspond to links between a movie database and an e-commerce inventory, so the tuples for each item define the distribution of the number of expected matches, formed by combining individual tuple linkage probabilities into PDFs. In this data set the maximum frequency of any item was 13, thus requiring us to estimate $V = 14$ frequency probabilities for each item (i.e., the probability that the frequency of each item is 0, 1,...,13).

Testing the approximation scheme. We experimented with the real data set and the following error metrics: Sum-Squared Error, (Squared) Hellinger Distance, L1 error, and Max-Error metric. We explore the effect of the approximation in construction time costs and in accuracy of the produced histograms. We contrast our algorithm with the optimal algorithm (Section 1.2), termed as phist.

We build histograms over N items using T terms and we study two basic cases. In the first case we set the approximation scheme only to the inter-bucket algorithm to produce $(1+\varepsilon)$-approximate Probabilistic Histograms. This case make more sense when the item domain $\mathscr{U}$ is huge and value domain $\mathscr{V}$ is small. We experimented with various domain $\mathscr{U}$ sizes (N=10000, N=27703, N=50000), to see how our approximate scheme behaves against the optimal algorithm as the size of $\mathscr{U}$ increases. In that experiments, the size of value domain was fixed to $|\mathscr{V}| = 14$. Also, we experimented with various choices of number of available terms (T=50, T=500, T=1000). For L1 error metric, we used the first 2000 items and 20 terms, because the computational cost for this algorithm is much higher.

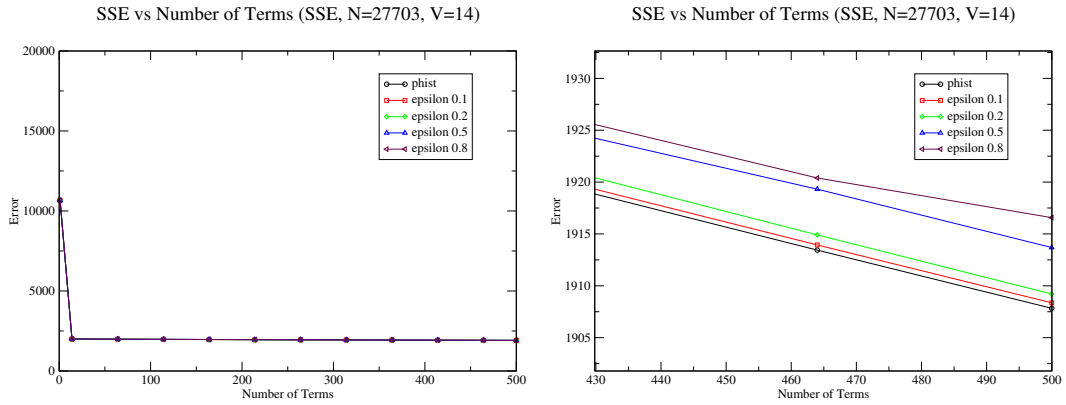
[1] `http://www.cs.washington.edu/homes/suciu/project-mystiq.html`

In the second case, we set the approximation scheme to both algorithms (inter- and intra-bucket) to produce $(1+\varepsilon)^2$-approximate Probabilistic Histograms. This case make more sense when the item domain $\mathscr{U}$ and the corresponding value domain $\mathscr{V}$ are huge. We experimented with a item domain of size $N = 4000$ and a value domain of size $|\mathscr{V}| = 131$, to see how our approximate scheme behaves against the optimal algorithm, when the value domain $\mathscr{V}$ is big. In that experiments, the number of available terms was $T = 200$.

We produce three graphs for each experiment. In the first graph, we show the error of the probabilistic histogram, according to the number of terms we used. In the second graph, we show the time taken to build the histogram (with a given number of terms), as the number of items (N) increases. Finally, the third graph, presents the time taken to build the histogram (with a given number of items), as the number of terms (T) increases.

In Figures 2.3 - 2.14, we show the results for the Sum-Squared Error metric. In Figures 2.15 - 2.20, we present the results for the (Squared) Hellinger Distance. The Figures 2.21 - 2.23 are for L1 Error metric and the Figures 2.24 - 2.26 are for Max Error metric.

In most cases, we see that our approximate algorithm produce almost-optimal Probabilistic Histograms. Moreover, in most cases, we see the clear benefits of our approximate algorithms in construction time costs.



(a) The overall picture (b) Zoomed picture

Figure 2.3: Sum-Squared Error vs Number of Terms: $(1+\varepsilon)$-approximation, N=27703, V=14

2.3.1 Sum-Squared Error Metric

1st Case Study:

In this case study, we use N=27703 item PDFs with a value domain of size $|\mathscr{V}| = 14$. The number of available terms, for histogram construction, varies from $T = 1$ to $T = 500$.We set the approximation scheme only to the inter-bucket algorithm, since the value domain $\mathscr{V}$ is small, i.e., we produce $(1+\varepsilon)$-approximations of optimal solution, where $\varepsilon \in \{0.1, 0.2, 0.5, 0.8\}$.

In Figure 2.3, we present the sum-squared error of our probabilistic histograms over N items, as the number of available terms increases. The Figure 2.3(a) is the overall picture. We see that the error decreases gradually as more terms are allowed, but in high rates for the first terms and in low rates as the number of terms becoming large, suggesting that there is a little benefit in using a large number of terms for this data set. The important observation in this graph, is that our approximate algorithm produces, even for large ε, i.e., $\varepsilon = 0.8$, almost-optimal probabilistic histograms. We show the negligible differences in sum-squared errors, between our approximate algorithm and the optimal algorithm (phist), in the zoomed picture 2.3(b). These negligible differences in sum-squared errors, making us to remember that the error bounds in the inductive proofs, was for worst case scenarios. As we expected, as the precision parameter ε increases, we have larger errors.

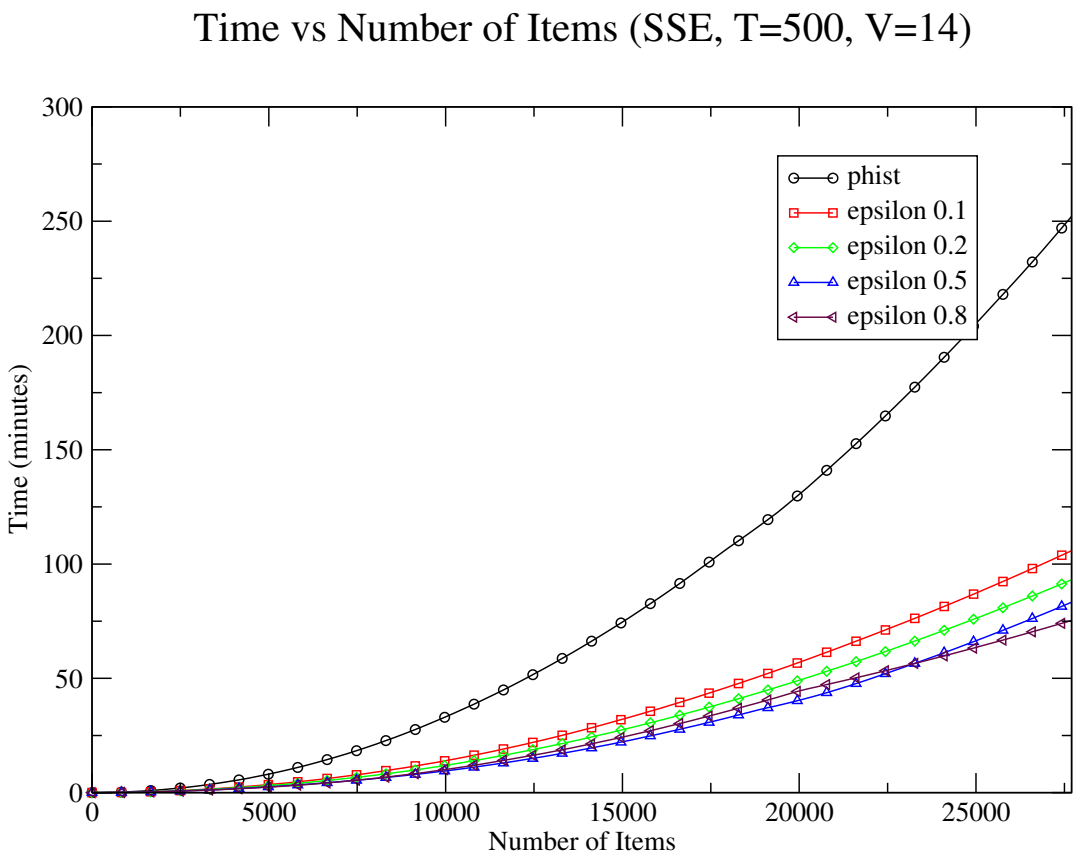


Figure 2.4: Sum-Squared Error Metric: Time as the number of items N varies, $(1+\varepsilon)$-approximation, T=500, V=14

In Figure 2.4, we present the time taken to build the optimal T-term probabilistic histogram and the corresponding $(1+\varepsilon)$-approximate T-term probabilistic histograms, under sum-squared error metric, over N items, as the number of items N increases. The number of available terms is fixed to $T = 500$. We observe the clear benefits of our approximation scheme, in construction time costs, against the optimal algorithm (phist). Even for small ε, i.e., $\varepsilon = 0.1$, our approximate algorithm builds the probabilistic histogram in less than the half time required by the optimal algorithm. As we expected, the benefits in construction time cost are larger, as the precision parameter ε increases.

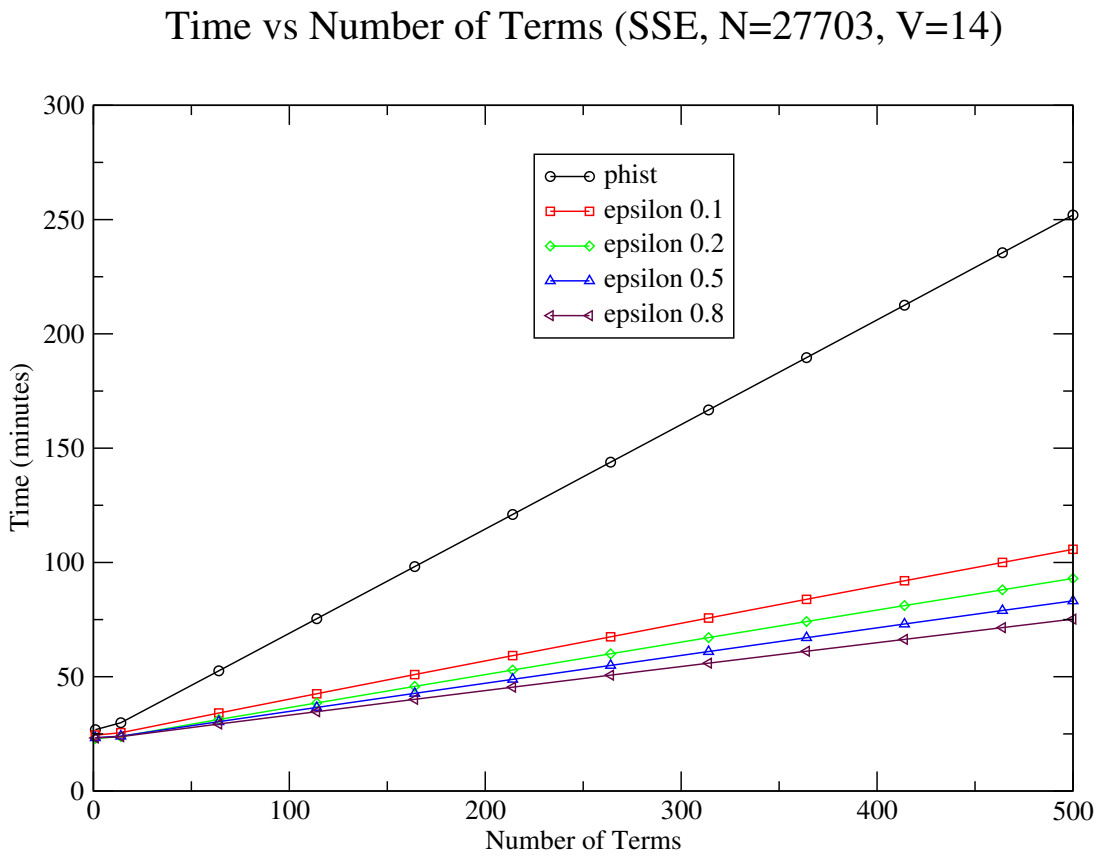
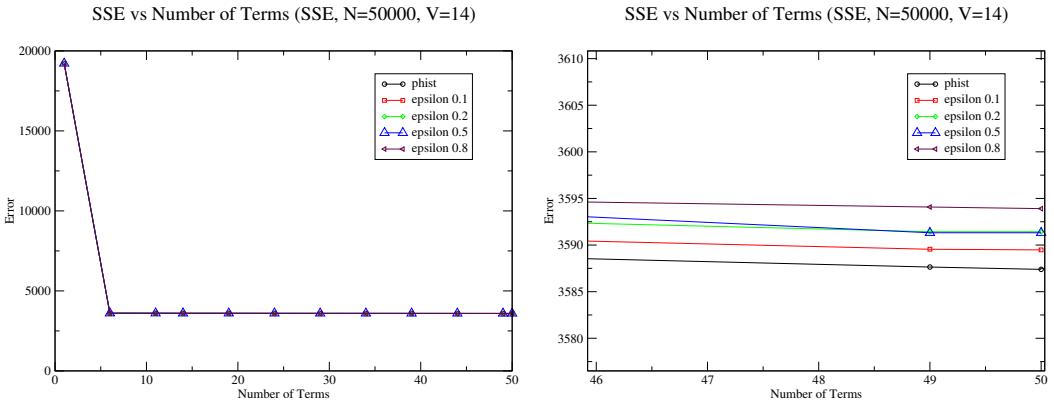


Figure 2.5: Sum-Squared Error Metric: Time as the number of terms T varies, $(1+\varepsilon)$-approximation, N=27703, V=14

In Figure 2.5, we present the time taken to build the optimal T-term probabilistic histogram and the corresponding $(1+\varepsilon)$-approximate T-term probabilistic histograms, under sum-squared error metric, over N items, as the number of terms T increases. The number items N is fixed to $N = 27703$. We observe the clear benefits of our approximation scheme, in construction time costs, against the optimal algorithm (phist). Even for small ε, i.e., $\varepsilon = 0.1$, our approximate algorithm builds the probabilistic histogram in less than the half time required by the optimal algorithm. As we expected, the benefits in construction time cost are larger, as the precision parameter ε increases. We also see that the time costs grow in a linear fashion as the number of terms increases.

2nd Case Study:

In this case study we increase the item domain size, i.e., we use N=50000 item PDFs with a value domain of size $|\mathscr{V}| = 14$, to see how our approximate algorithm behaves with large item domain sizes. Our purpose is to show the $O(NlogN)$ nature of our approximate algorithm against the $O(N^2)$ nature of the optimal algorithm, as the item domain $\mathscr{U}$ becoming huge. The benefit in construction time costs, of our approximate algorithm, is clearer, as the size of item domain $\mathscr{U}$ increases. Also, we have clearer benefits, when the number of available terms is small, because the δ factor of our approximate algorithm was fixed by the inductive proofs to $\delta = \varepsilon/2T$. Thus, small T produces large δ, leading to small sizes of interval lists maintained by our approximate algorithm. In this case study, the number of available terms varies from $T = 1$ to $T = 50$. We set the approximation scheme only to the inter-bucket algorithm, i.e., we produce $(1+\varepsilon)$-approximations of optimal solution.



(a) The overall picture (b) Zoomed picture

Figure 2.6: Sum-Squared Error vs Number of Terms: $(1+\varepsilon)$-approximation, N=50000, V=14

In Figure 2.6, we present the sum-squared error of our $(1+\varepsilon)$-probabilistic histograms, over N items, as the number of available terms increases. From this figure, someone can extract similar observations with the previous experiment (Fig. 2.3), i.e., that our approximation scheme produces almost-optimal probabilistic histograms.

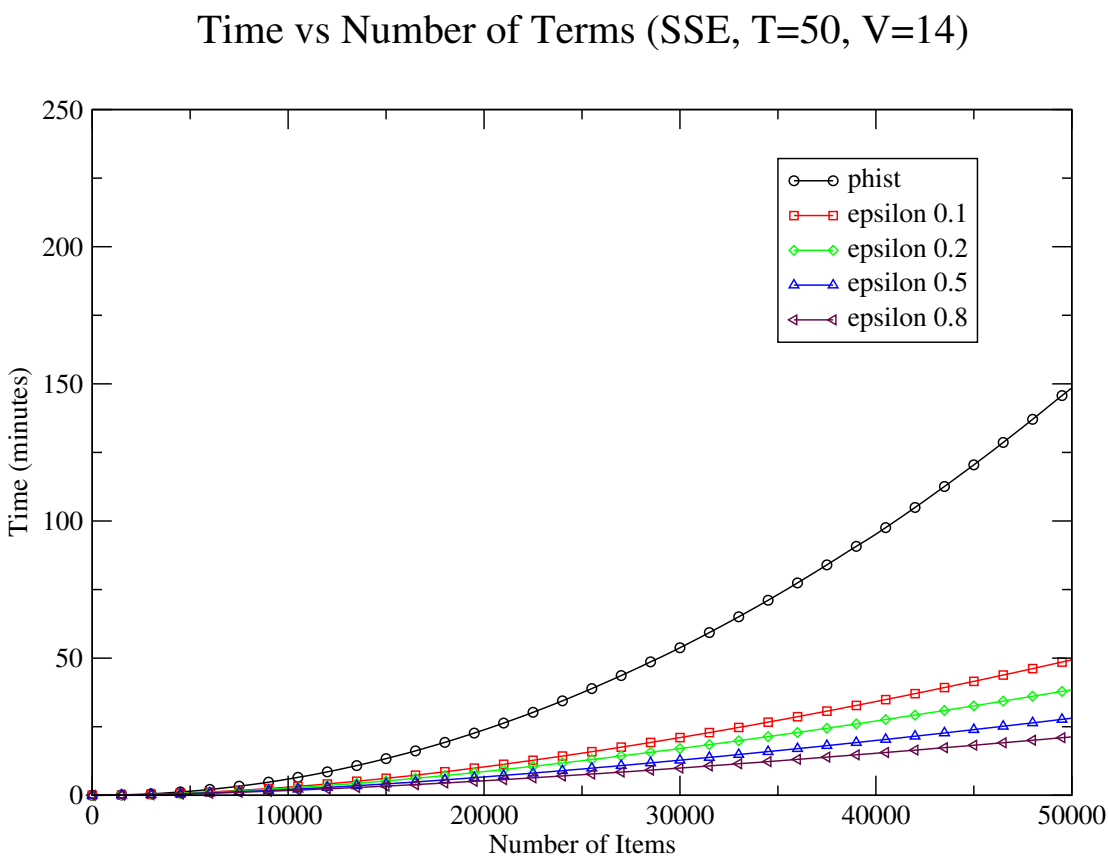


Figure 2.7: Sum-Squared Error Metric: Time as the number of items N varies, $(1+\varepsilon)$-approximation, T=50, V=14

In Figure 2.7, we present the time taken to build the optimal T-term probabilistic histogram and the corresponding $(1+\varepsilon)$-approximate T-term probabilistic histograms, under sum-squared error metric, over N items, as the number of items N increases. The number of available terms is fixed to $T = 50$. We observe the clear benefits of our approximation scheme, in construction time costs, against the optimal algorithm (phist). Even for small ε, i.e., $\varepsilon = 0.1$, our approximate algorithm builds the probabilistic histogram in $1/3$ of time required by the optimal algorithm. When $\varepsilon = 0.8$, our algorithm builds the histogram in $1/6$ of time required by the optimal algorithm. The benefits in construction time cost are larger, as the precision parameter ε increases.

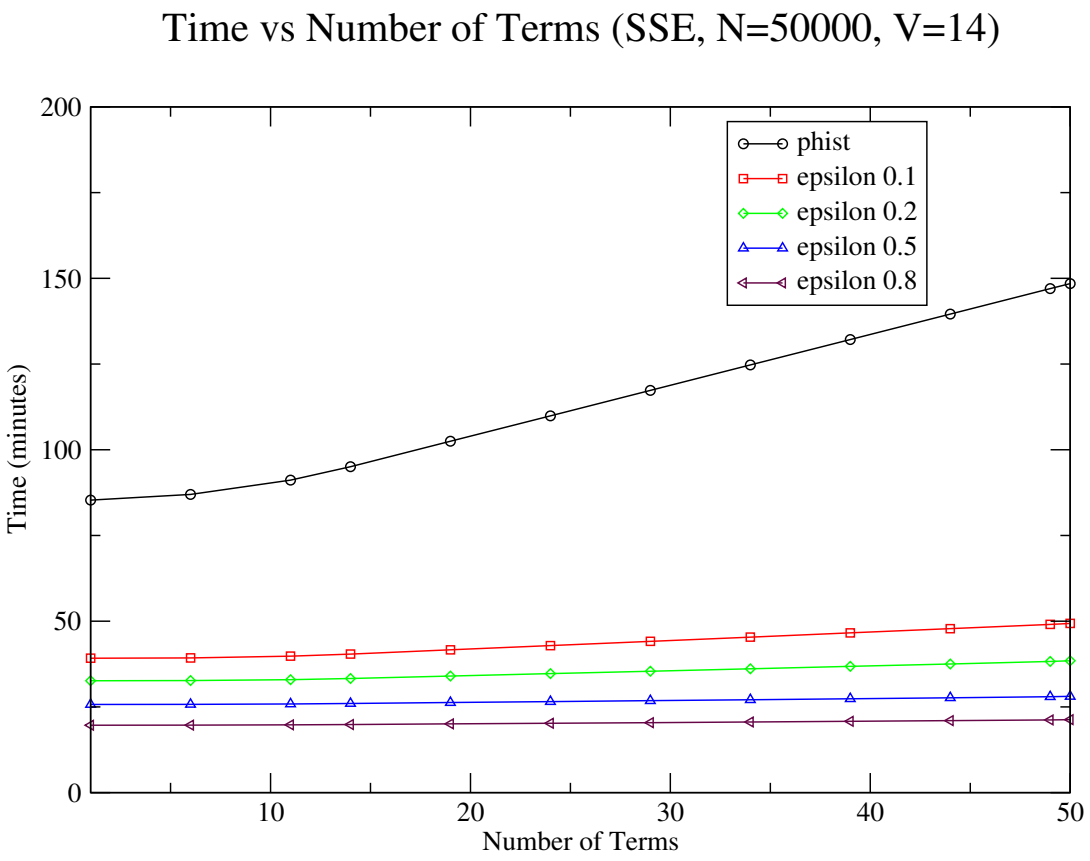
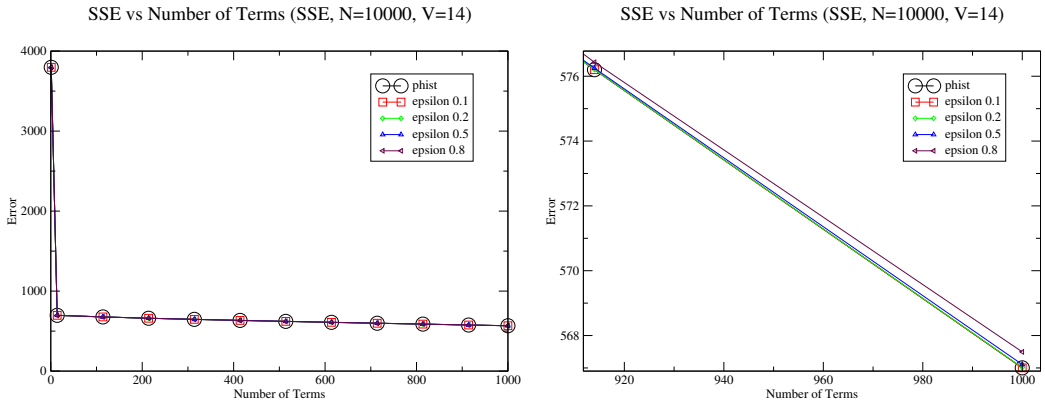


Figure 2.8: Sum-Squared Error Metric: Time as the number of terms T varies, $(1+\varepsilon)$-approximation, N=50000, V=14

In Figure 2.8, we present the time taken to build the optimal T-term probabilistic histogram and the corresponding $(1+\varepsilon)$-approximate T-term probabilistic histograms, under sum-squared error metric, over N items, as the number of terms T increases. In this experiment, the number items N is fixed to $N = 50000$. We observe again (as in Fig. 2.7) the clear benefits of our approximation scheme, in construction time costs, against the optimal algorithm (phist). We observe again (as in Fig. 2.5) that the time costs grow in a linear fashion as the number of terms increases.

3rd Case Study:



(a) The overall picture (b) Zoomed picture

Figure 2.9: Sum-Squared Error vs Number of Terms: $(1+\varepsilon)$-approximation, N=10000, V=14

In this case study, we allow a large number of available terms, i.e., $T = 1$ to $T = 1000$, to see how our approximation scheme behaves with large numbers of available terms. We use N=10000 item PDFs with a value domain of size $|\mathcal{V}| = 14$. We set the approximation scheme only to the inter-bucket algorithm, i.e., we produce $(1+\varepsilon)$-approximations of optimal solution.

In Figure 2.9, we present the sum-squared error of our $(1+\varepsilon)$-probabilistic histograms, over N items, as the number of available terms increases. Again, someone can extract similar observations with the Fig. 2.3 (negligible differences in sum-squared errors).

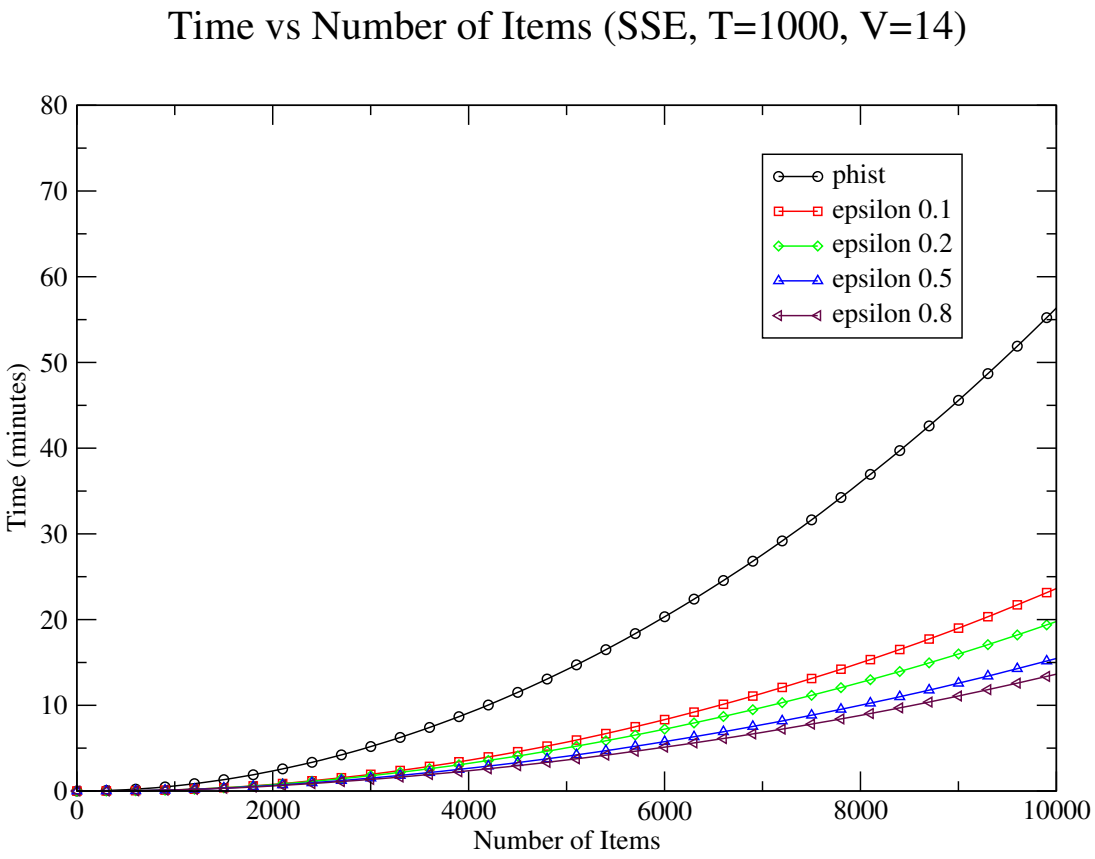


Figure 2.10: Sum-Squared Error Metric: Time as the number of items N varies, $(1+\varepsilon)$-approximation, T=1000, V=14

In Figure 2.10, we present the time taken to build the optimal T-term probabilistic histogram and the corresponding $(1+\varepsilon)$-approximate T-term probabilistic histograms, under sum-squared error metric, over N items, as the number of items N increases. The number of available terms is fixed to $T = 1000$, i.e., 10% of total number of items. Even for small ε, i.e., $\varepsilon = 0.1$, our approximate algorithm builds the probabilistic histogram in less than the half of time required by the optimal algorithm. Again, the benefits in construction time cost are larger, as the precision parameter ε increases.

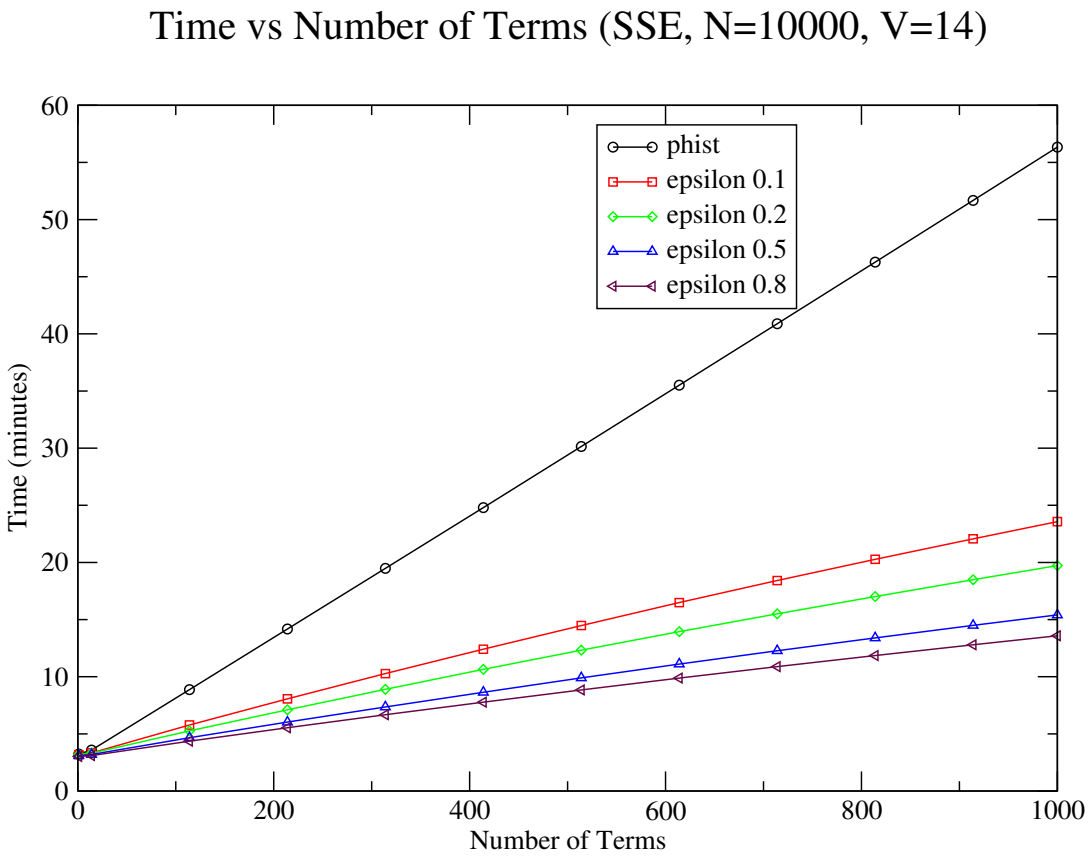
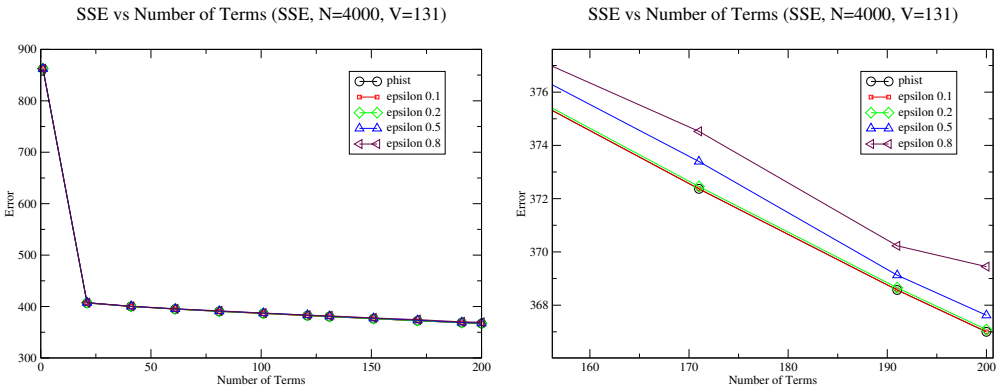


Figure 2.11: Sum-Squared Error Metric: Time as the number of terms T varies, $(1+\varepsilon)$-approximation, N=10000, V=14

In Figure 2.11, we present the time taken to build the optimal T-term probabilistic histogram and the corresponding $(1+\varepsilon)$-approximate T-term probabilistic histograms, under sum-squared error metric, over N items, as the number of terms T increases from $T = 1$ to $T = 1000$. The number items N is fixed to $N = 10000$. We observe again (as in Fig. 2.10) the benefits of our approximation scheme, in construction time costs, against the optimal algorithm (phist). Also, we see (as in Fig. 2.5) that the time costs grow in a linear fashion as the number of terms increases.



(a) The overall picture (b) Zoomed picture

Figure 2.12: Sum-Squared Error vs Number of Terms: $(1+\varepsilon)^2$-approximation, N=4000, V=131

4th Case Study:

In this case study, we use N=4000 "heavy" item PDFs with a value domain of size $|\mathscr{V}| = 131$. We set the approximation scheme both to the inter- and intra-bucket algorithms, since the value domain $\mathscr{V}$ is relatively big, i.e., we produce $(1+\varepsilon)^2$-approximations of optimal solution, where $\varepsilon \in \{0.1, 0.2, 0.5, 0.8\}$. Thus, we explore the effect of $(1+\varepsilon)^2$-approximation in quality and scalability of the produced histograms, when the value domain is big and make more sense to set the approximation scheme both to the inter- and intra-bucket algorithms.

In Figure 2.12, we present the sum-squared error of our probabilistic histograms over N items as the number of available terms increases. Even for large choices for ε, $\varepsilon = 0.8$, we see again negligible differences, in sum-squared errors, between our $(1+\varepsilon)^2$-approximations and the corresponding optimal errors.

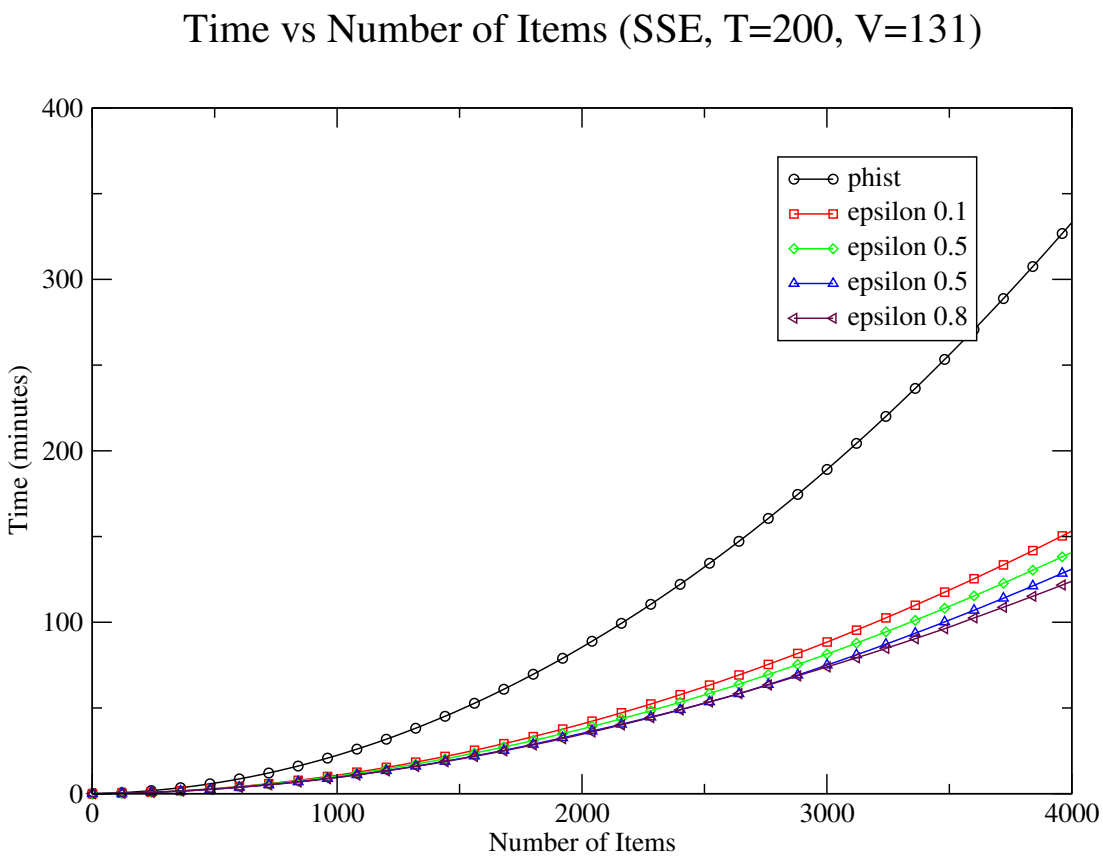


Figure 2.13: Sum-Squared Error Metric: Time as the number of items N varies, $(1+\varepsilon)^2$-approximation, T=200, V=14

In Figure 2.13, we present the time taken to build the optimal T-term probabilistic histogram and the corresponding $(1+\varepsilon)^2$-approximate T-term probabilistic histograms, under sum-squared error metric, over the N "heavy" item PDFs, as the number of items N increases. The number of available terms is fixed to $T = 200$. As in the case of $(1+\varepsilon)$-approximation, when we set the approximation scheme both to inter- and intra bucket algorithms, we see again the clear benefits, in construction time costs, against the optimal algorithm (phist). Even for small ε, i.e., $\varepsilon = 0.1$, our approximate algorithm builds the probabilistic histogram in less than the half time required by the optimal algorithm. Again, as we expected, the benefits in construction time cost are larger, as the precision parameter ε increases.

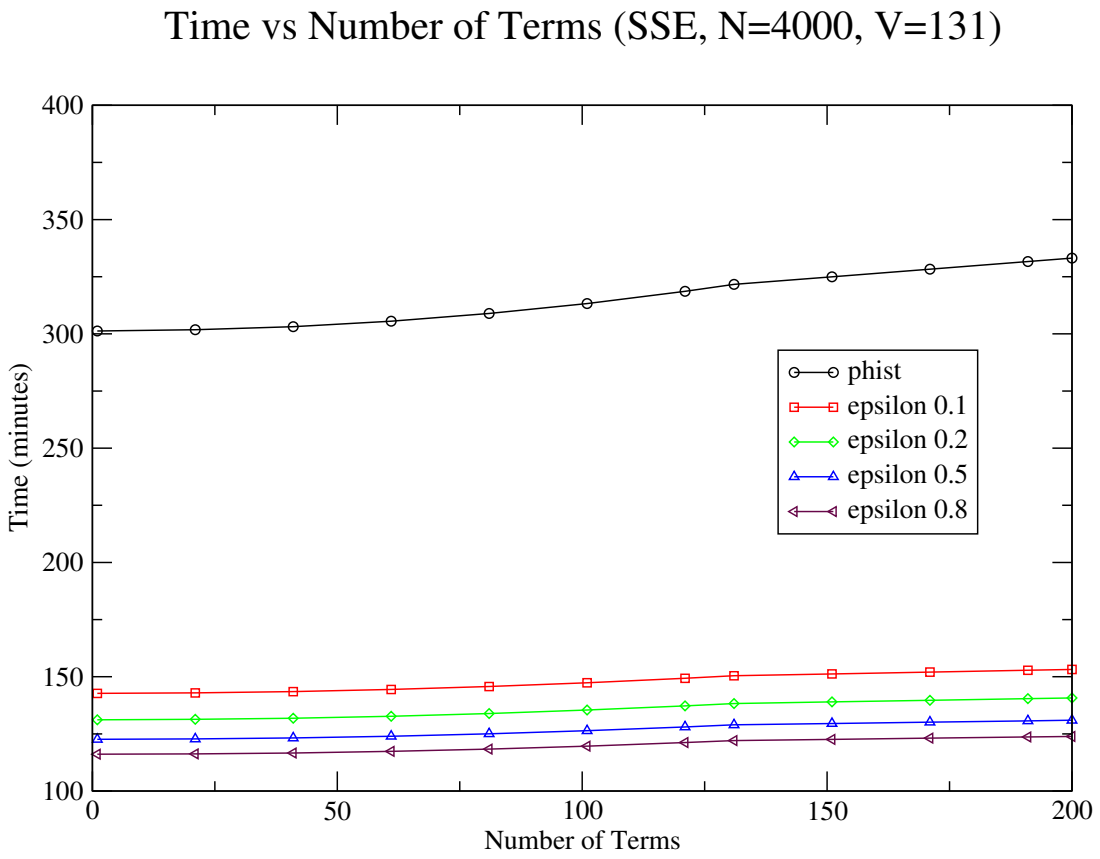


Figure 2.14: Sum-Squared Error Metric: Time as the number of terms T varies, $(1+\varepsilon)^2$-approximation, N=4000, V=14

In Figure 2.14, we present the time taken to build the optimal T-term probabilistic histogram and the corresponding $(1+\varepsilon)^2$-approximate T-term probabilistic histograms, under sum-squared error metric, over N "heavy" items, as the number of terms T increases. The number items N is fixed to $N = 4000$. We observe again (as in Fig. 2.13) the clear benefits of our approximation scheme, in construction time costs, against the optimal algorithm (phist). Again, the time costs grow in a linear fashion as the number of terms increases.

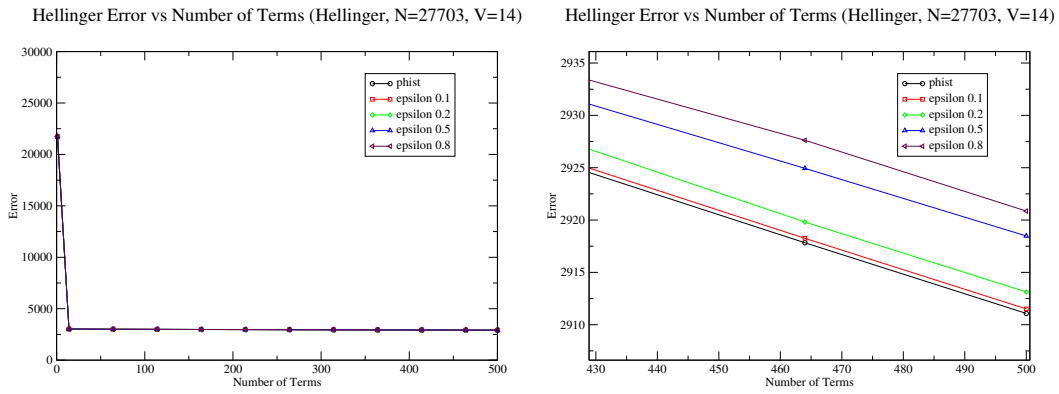
2.3.2 (Squared) Hellinger Distance

The (squared) Hellinger distance is a similar error metric with the sum-squared error metric. Thus, the results for this metric are quite similar with the corresponding results of sum-squared error metric. Here, we present two experiments for the (Squared) Hellinger Distance.

In the first experiment (1st case study: Figs 2.15 - 2.17), we use N=27703 items with a value domain of size $|\mathcal{V}| = 14$, and we build $(1+\varepsilon)$-approximate probabilistic histograms over the N items, using at most T=500 terms. The results are quite similar with the corresponding case study for sum-squared error metric, i.e., Figs 2.3 - 2.5.

In the second experiment (2nd case study: Figs 2.18 - 2.20), we use N=4000 "heavy" items with a large value domain of size $|\mathcal{V}| = 131$, and we build $(1+\varepsilon)^2$-approximate probabilistic histograms over the N items, using at most T=200 terms. The results are quite similar with the corresponding case study for sum-squared error metric, i.e., Figs 2.12 - 2.14.

1st Case Study:



(a) The overall picture (b) Zoomed picture

Figure 2.15: (Squared) Hellinger Distance vs Number of Terms: $(1+\varepsilon)$-approximation, N=27703, V=14

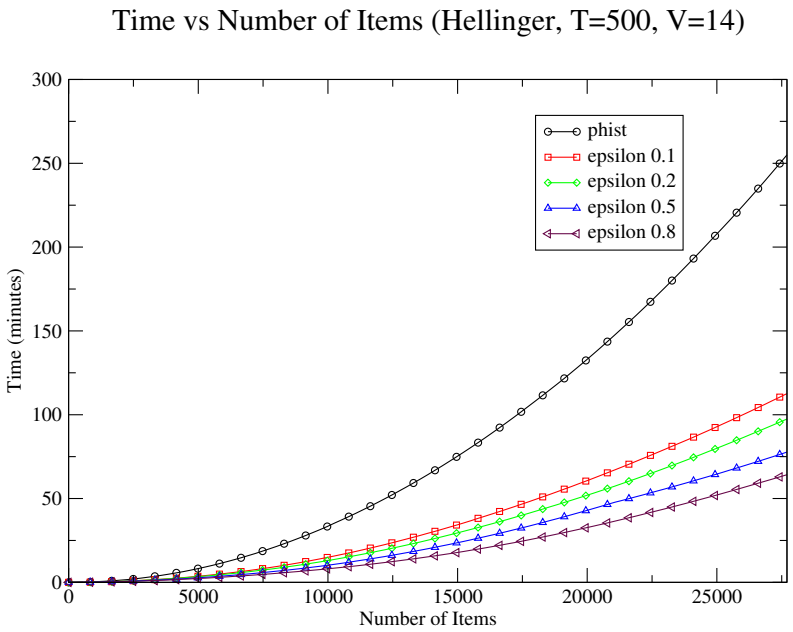


Figure 2.16: (Squared) Hellinger Distance: Time as the number of items N varies, $(1+\varepsilon)$-approximation, T=500, V=14

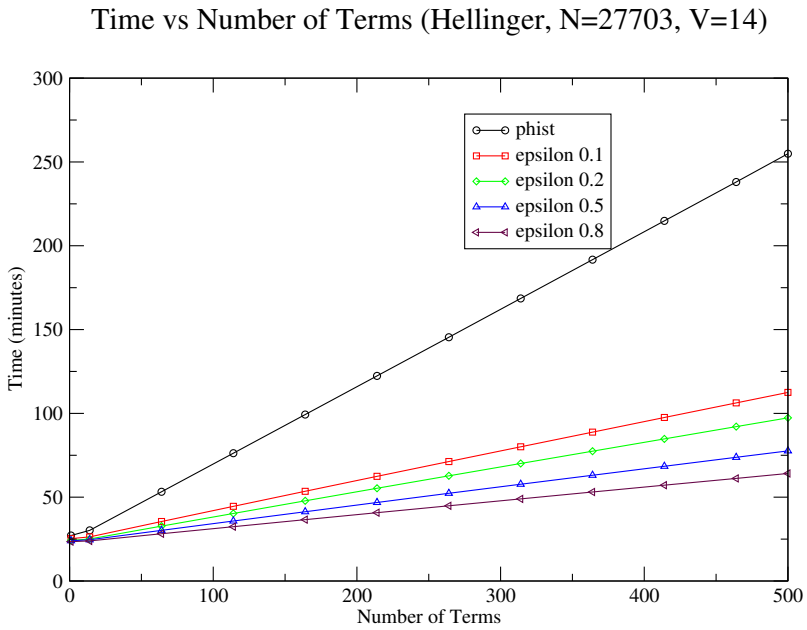
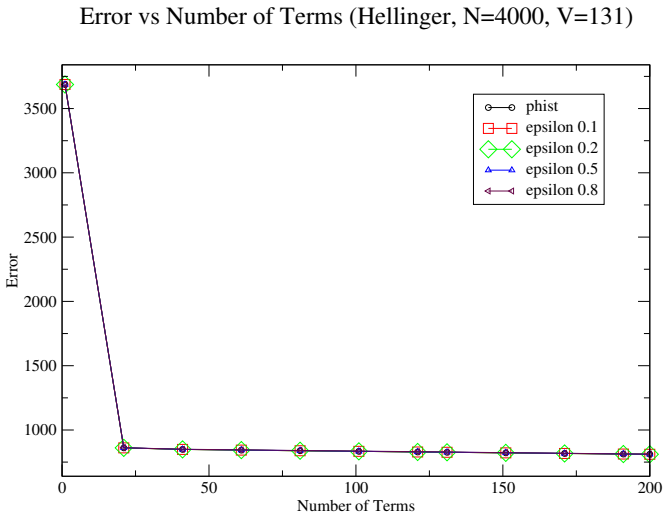
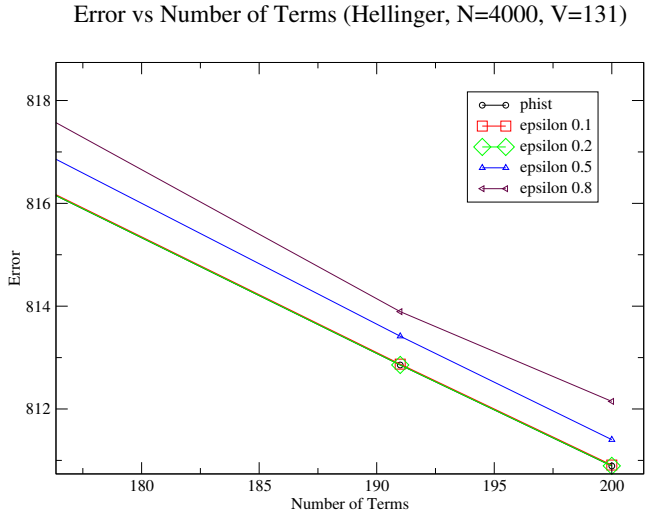


Figure 2.17: (Squared) Hellinger Distance: Time as the number of terms T varies, $(1+\varepsilon)$-approximation, N=27703, V=14

2nd Case Study:



(a) The overall picture



(b) Zoomed picture

Figure 2.18: (Squared) Hellinger Distance vs Number of Terms: $(1+\varepsilon)^2$-approximation, N=4000, V=131

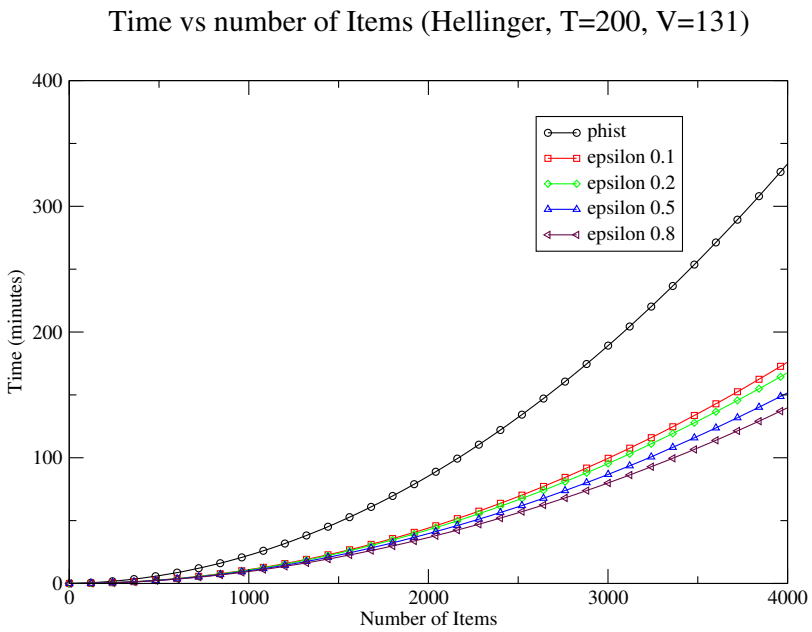


Figure 2.19: (Squared) Hellinger Distance: Time as the number of items N varies, $(1+\varepsilon)^2$-approximation, T=200, V=131

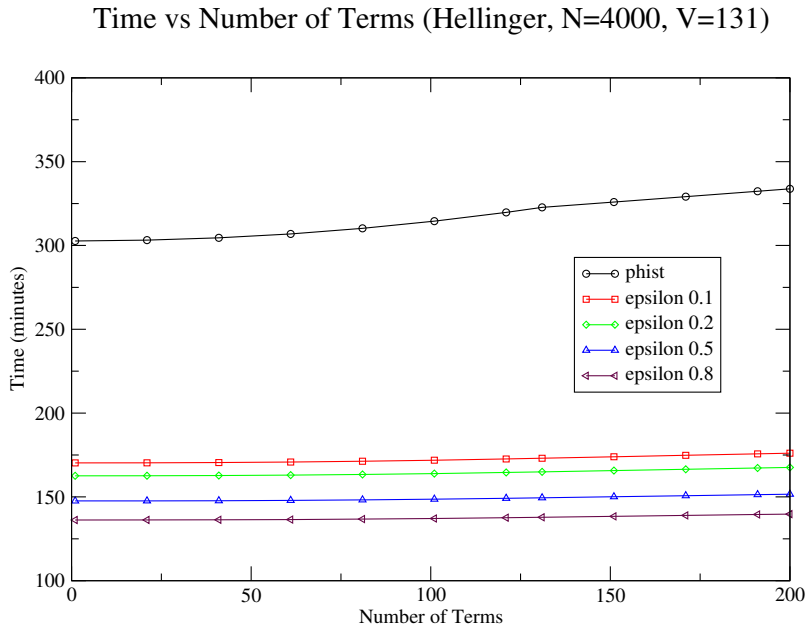
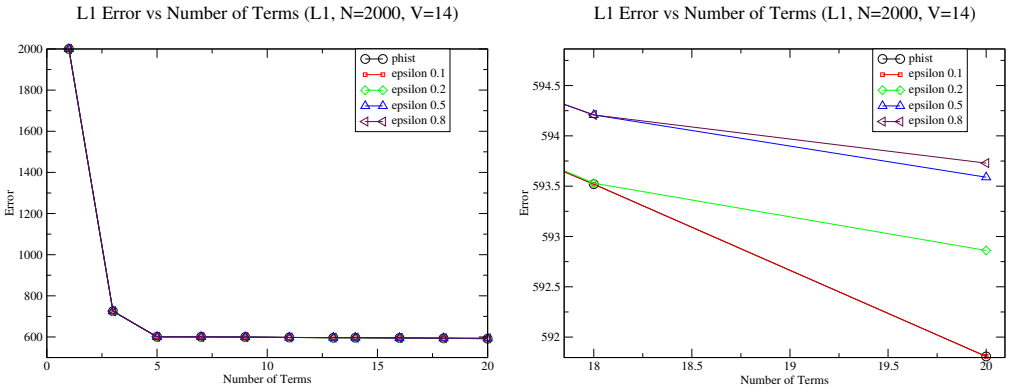


Figure 2.20: (Squared) Hellinger Distance: Time as the number of terms T varies, $(1+\varepsilon)^2$-approximation, N=4000, V=131

2.3.3 L1 Error Metric

The L1 error metric is the most time consuming metric relatively with the other metrics explored. This metric has much higher computational cost than the other metrics considered, since the computation of VALERR function for L1 error metric, is associated with the time consuming two-dimensional range-sum-median problem [8].

Thus, in the case study presented here, we use a relatively small dataset, consisting of N=2000 items with a value domain of size $|\mathscr{V}| = 14$. We build T-term $(1+\varepsilon)$-approximate probabilistic histograms over N items using at most 20 terms. We set the approximation scheme only to the inter-bucket algorithm, i.e., we produce $(1+\varepsilon)$-approximations of optimal solution. We contrast our results, i.e., scalability and quality of the produced $(1+\varepsilon)$-approximate histograms, against the optimal histogram.



(a) The overall picture (b) Zoomed picture

Figure 2.21: L1 Error vs Number of Terms: $(1+\varepsilon)$-approximation, N=2000, V=14

In Figure 2.21, we present the L1 error of our $(1+\varepsilon)$-probabilistic histograms, over N items, as the number of available terms increases. From this figure, someone can extract similar observations with the other metrics, i.e., our approximate scheme produces almost-optimal probabilistic histograms (negligible differences between the $(1+\varepsilon)$-approximations and the optimal L1 error).

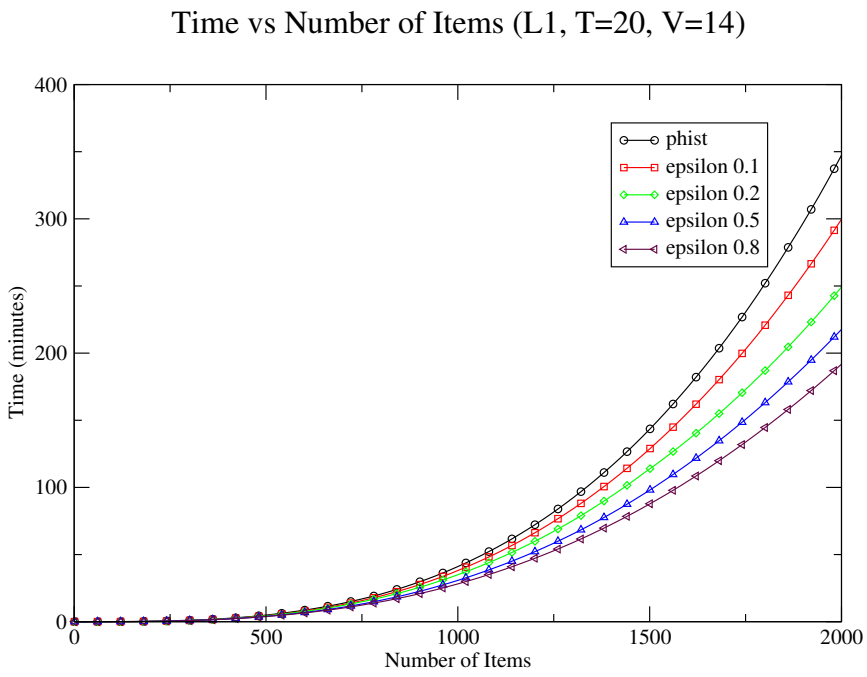


Figure 2.22: L1 Error Metric: Time as the number of items N varies, $(1+\varepsilon)$-approximation, T=20, V=14

In Figure 2.22, we present the time taken to build the optimal T-term probabilistic histogram and the corresponding $(1+\varepsilon)$-approximate T-term probabilistic histograms, under L1 error metric, over N items, as the number of items N increases. The number of available terms is fixed to $T = 20$. We observe the benefits of our approximation scheme, in construction time costs, against the optimal algorithm (phist). The results for this experiment are not such good as in the other metrics, since the benefits of our approximation scheme are clearer as the domain of items increases. But, because of high computational cost of this metric, explained before, it was not possible to examine larger data sets. However, even with a small dataset of 2000 items, we observe that our approximate algorithm when $\varepsilon = 0.8$, builds the histogram almost in half of the time required by the optimal algorithm (phist). As in the other metrics, the benefits in construction time cost are larger, as the precision parameter ε increases.

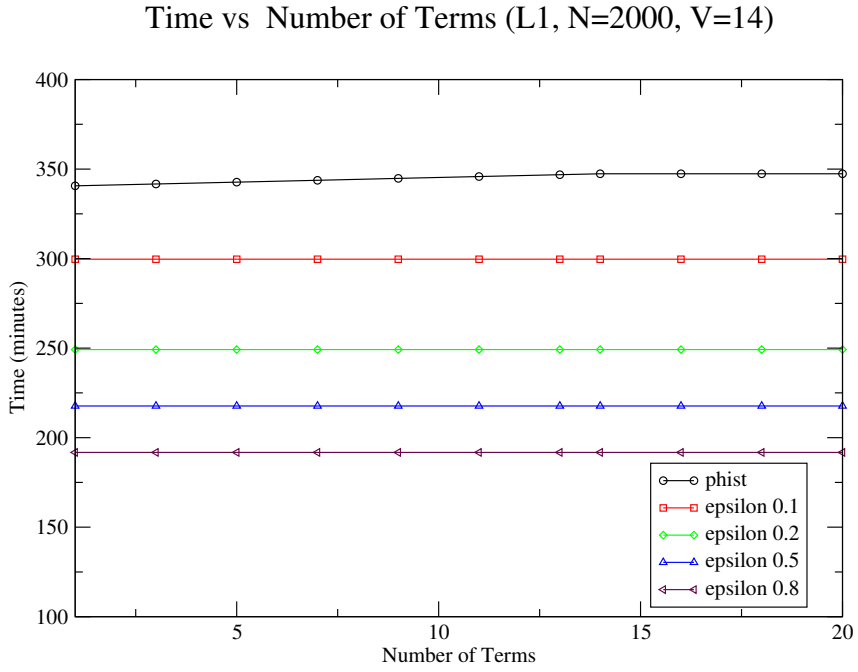


Figure 2.23: L1 Error Metric: Time as the number of terms T varies, $(1+\varepsilon)$-approximation, N=2000, V=14

In Figure 2.23, we present the time taken to build the optimal T-term probabilistic histogram and the corresponding $(1+\varepsilon)$-approximate T-term probabilistic histograms, under L1 error metric, over N items, as the number of terms T increases from $T = 1$ to $T = 20$. The number of items N is fixed to $N = 2000$. We observe again (as in Fig. 2.22) the benefits of our approximation scheme, in construction time costs, against the optimal algorithm (phist).

2.3.4 Max-Error Metric

The best results came from Max-Error metric. Clearly, from the Max-Error metric properties, there are large ranges with the same max-error. Specifically, let $maxErr(i,t)$ denote the max-error in the t-term representation up to domain point $i \in \mathscr{U}$. Also, let the current measured max-error caused by item $j \in \mathscr{U}$. For a given number of t terms, this error will be the same with the errors $maxErr(j+1,t), maxErr(j+2,t), \ldots, maxErr(j+k,t)$, until a next item (let be j+k+1) contributes to a larger error, i.e., $maxErr(j+k+1,t) > maxErr(j,t)$. From our approximation scheme properties, all these ranges will be stored in the interval lists maintained by our algorithm. Thus, the searching space of our approximate algorithm is drastically reduced relatively with the searching space of optimal algorithm, since we need to remember only the max-errors at the ends of such large intervals.

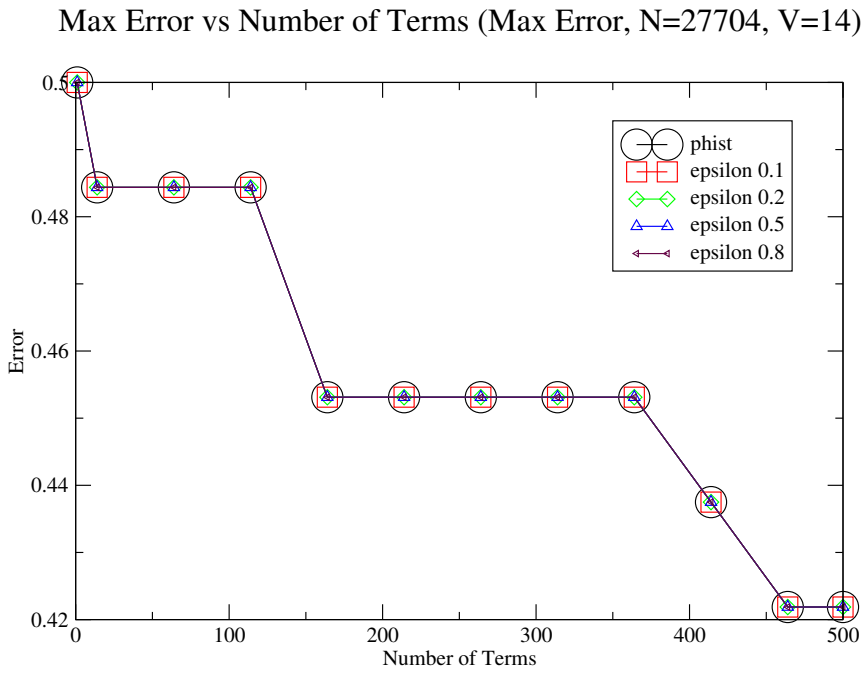


Figure 2.24: Max Error vs Number of Terms, , $(1+\varepsilon)$-approximation, N=27703, V=14

We use N=27703 item PDFs with a value domain of size $|\mathscr{V}| = 14$. The number of available terms, for histogram construction, varies from $T = 1$ to $T = 500$. We produce $(1+\varepsilon)$-approximations of optimal solution, where $\varepsilon \in \{0.1, 0.2, 0.5, 0.8\}$.

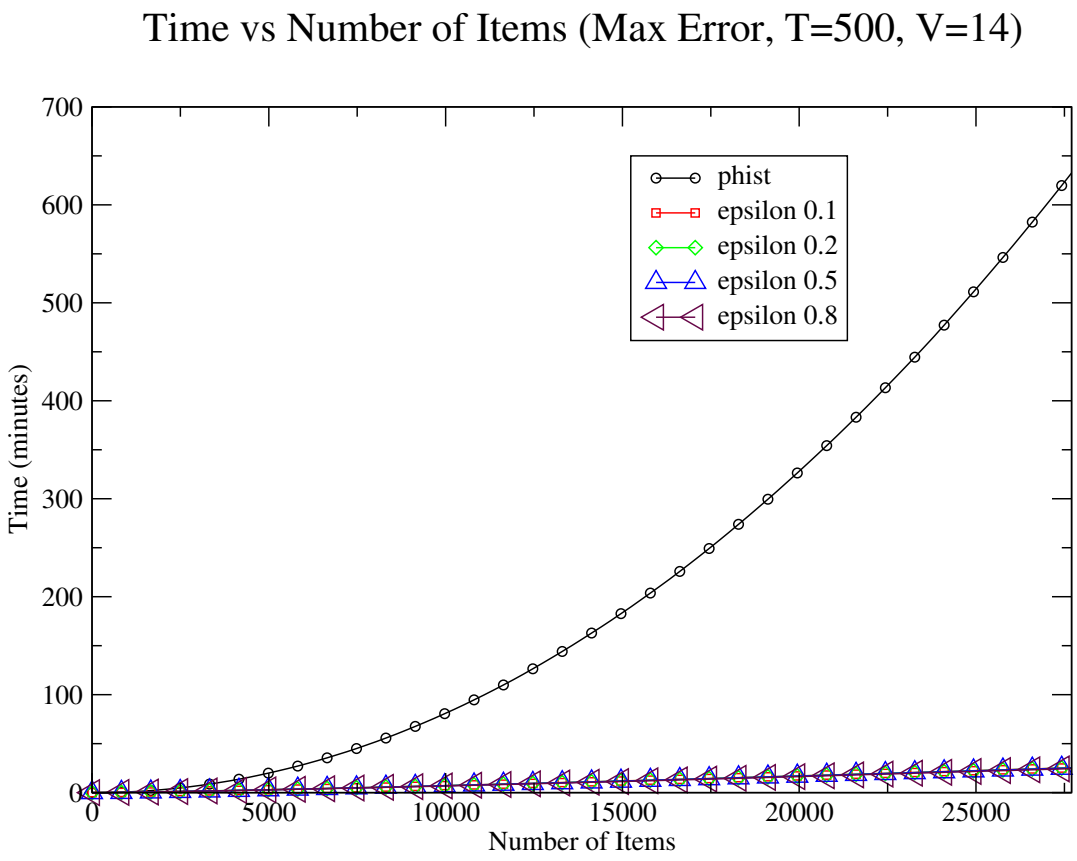


Figure 2.25: Max Error Metric: Time as the number of items N varies, $(1+\varepsilon)$-approximation, T=500, V=14

In Figure 2.24, we present the max error of our probabilistic histograms over N items as the number of available terms increases. Even for large choices for the precision parameter ε, the $(1+\varepsilon)$-approximate max-error curves are identical with the corresponding optimal error curve.

In Figure 2.25, we present the time taken to build the optimal T-term probabilistic histogram and the corresponding $(1+\varepsilon)$-approximate T-term probabilistic histograms, under max-error metric, over N items, as the number of items N increases. In this experiment, the number of available terms is fixed to $T = 500$. We observe the extraordinary benefits of our approximation scheme, in construction time costs, against the optimal algorithm (phist). It is remarkable that time costs, of our approximate algorithm, grow in a linear fashion as the number of items increases.

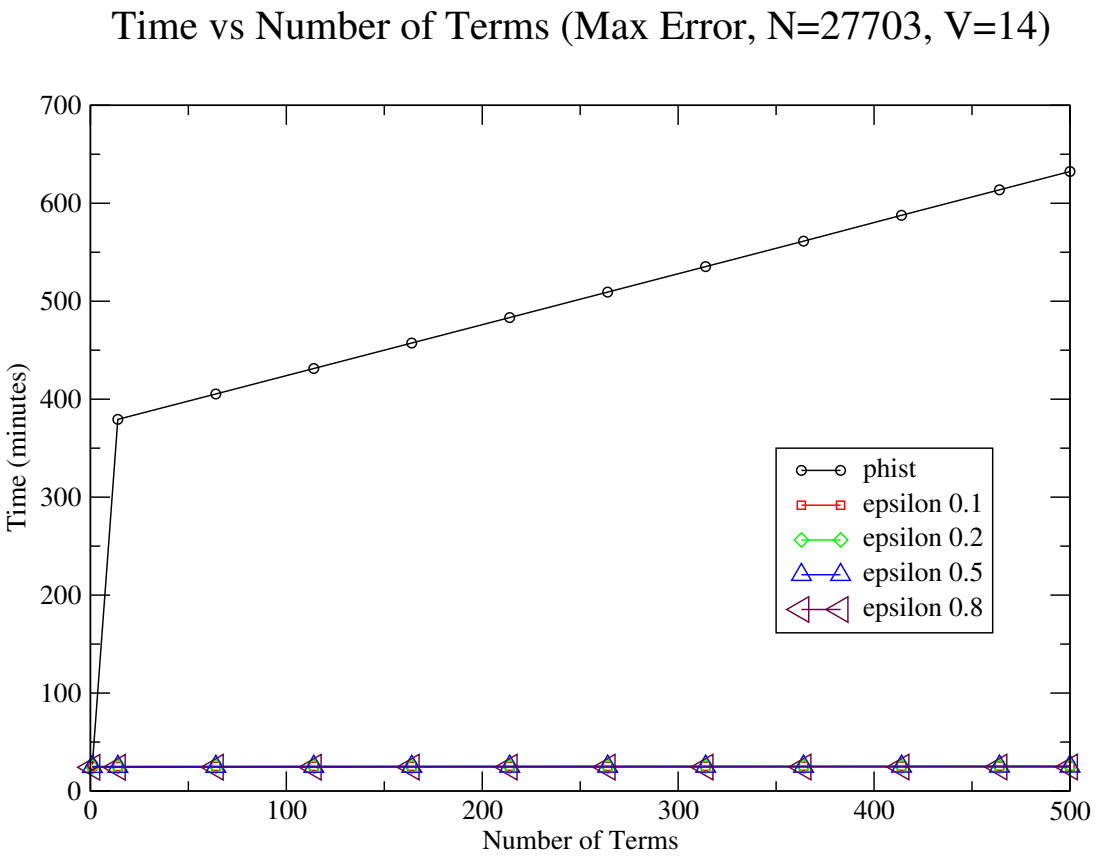


Figure 2.26: Max Error Metric: Time as the number of terms T varies, $(1+\varepsilon)$-approximation, N=27703, V=14

In Figure 2.26, we present the time taken to build the optimal T-term probabilistic histogram and the corresponding $(1+\varepsilon)$-approximate T-term probabilistic histograms, under max-error metric, over N items, as the number of terms T increases from $T = 1$ to $T = 500$. In this experiment, the number of items N is fixed to $N = 27703$. We observe again (as in Fig. 2.25) the extraordinary benefits of our approximation scheme, in construction time costs, against the optimal algorithm (phist). As in the other metrics, we observe that the time costs grow in a linear fashion as the number of terms increases.

Chapter 3

Concluding Remarks and Future Work

Histograms have proven to be a very effective summarization mechanism, and are widely used to capture data distributions. Currently, they are an important part in commercial query engines.

Moreover, there is a growing realization that modern DBMSs must be able to manage data that contain *uncertainties* that are represented in the form of probabilistic relations. However, it is difficult to work with the probabilistic data, because of their high complexity in contrast with the deterministic case. Even for simple queries, there is a $\#P$ hard complexity [9], and thus effective probabilistic histogram construction arises.

The authors of [8] proposed optimal dynamic programming algorithms to build optimal probabilistic histograms for probabilistic data. Although, their algorithms are optimal in terms of overall error, they have high time-complexity.

Since, histograms are themselves approximation schemes, an extra approximation factor for speed up can be tolerable. The main contribution of this thesis, was the incorporation of $(1+\varepsilon)$-approximation schemes, in the case of probabilistic data. In order to reduce the high computational cost of optimal probabilistic histogram construction, we set an approximation scheme to the general DP framework. The speed up is obtained, because our approximate algorithm takes into consideration only a small set of

sub-problems in the DP framework. Our algorithm provides guarantees to the overall error of the histogram, and gives the user a useful trade-off between the accuracy of the histogram and the computational cost. Our experimental study, to explore the effect of such approximation, shows that our approximation scheme produces almost-optimal probabilistic histograms, with clear benefits in time-complexity against the optimal algorithm.

It would be interesting to adjust more sophisticated approximation schemes, like those in [16], to the two-dimensional case of probabilistic data. Another idea for speed up, is to use map-reduce techniques to have parallel computation. Finally, it would be interesting to examine other error metrics, e.g., Earth Movers Distance.

Bibliography

[1] C. Aggarwal, editor. *Managing and Mining Uncertain Data*. Springer, 2009.

[2] N. Alon, Y. Matias, and M. Szegedy. The space complexity of approximating the frequency moments. In *ACM Symposium on Theory of Computing*, pages 20–29, 1996. Journal version in *Journal of Computer and System Sciences*, 58:137–147, 1999.

[3] L. Antova, T. Jansen, C. Koch, and D. Olteanu. Fast and simple relational processing of uncertain data. In *IEEE International Conference on Data Engineering*, 2008.

[4] O. Benjelloun, A. D. Sarma, C. Hayworth, and J. Widomn. An introduction to ULDBs and the Trio system. *IEEE Data Engineering Bulletin*, 29(1):5–16, Mar. 2006.

[5] J. Boulos, N. Dalvi, B. Mandhani, S. Mathur, C. Re, and D. Suciu. Mystiq: A system for finding more answers by using probabilities. In *ACM SIGMOD International Conference on Management of Data*, 2005.

[6] G. Cormode and M. Garofalakis. Sketching probabilistic data streams. In *ACM SIGMOD International Conference on Management of Data*, 2007.

[7] G. Cormode and M. Garofalakis. *Histograms and wavelets on probabilistic data*. In IEEE International Conference on Data Engineering, 2009.

[8] G. Cormode, A. Deligiannakis, M. Garofalakis, A. McGregor. *Probabilistic Histograms for Probabilistic Data*. In the Proceedings of VLDB'2009 (PVLDB, Vol. 2), Lyon, France, August, 2009.

[9] N. Dalvi and D. Suciu. Efficient query evaluation on probabilistic databases. In *International Conference on Very Large Data Bases*, 2004.

[10] N. Dalvi and D. Suciu. Management of probabilistic data: foundations and challenges. In *ACM Principles of Database Systems*, 2007.

[11] M. Datar, A. Gionis, P. Indyk, and R. Motwani. Maintaining stream statistics over sliding windows. Proc. of SODA, pages 635-644, 2002.

[12] J. Feigenbaum, S. Kannan, M. Strauss, and M. Viswanathan. An approximate L1-difference algorithm for massive data streams. SIAM J. Comput., 32(1):131-151, 2002.

[13] P. Gibbons and Y. Matias. Synopsis data structures for massive data sets. Proc. of SODA, pages 909-910, 1999.

[14] A. C. Gilbert, S. Guha, P. Indyk, Y. Kotidis, S. Muthukrishnan, and Martin Strauss. Fast, small-space algorithms for approximate histogram maintenance. Proc. of ACM STOC, pages 389-398, 2002.

[15] Sudipto Guha, Nick Koudas, Kyuseok Shim. *Approximation and Streaming Algorithms for Histogram Construction Problems*. ACM Transactions on Database Systems, Vol. 31, No. 1, March 2006, Pages 396-438.

[16] Sudipto Guha, Nick Koudas, Kyuseok Shim. *Data-Streams and Histograms*. STOC01, July 6-8, 2001, Hersonissos, Crete, Greece.

[17] H. V Jagadish, N. Koudas, S. Muthukrishnan, V. Poosala, K. C. Sevcik, and T. Suel. *Optimal Histograms with Quality Guarantees*. In the Proceedings of the VLDB Conference, pages 275-286, 1998.