Consistent Histograms In The Presence of Distinct Value Counts

Raghav Kaushik
Microsoft Research
skaushi@microsoft.com

Dan Suciu
University of Washington
suciu@cs.washington.edu

ABSTRACT

Self-tuning histograms have been proposed in the past as an attempt to leverage feedback from query execution. However, the focus thus far has been on histograms that only store cardinalities. In this paper, we study consistent histogram construction from query feedback that also takes distinct value counts into account.

We first show how the entropy maximization (EM) principle can be leveraged to identify a distribution that approximates the data given the execution feedback making the least additional assumptions. This EM model that takes both distinct value counts and cardinalities into account. However, we find that it is computationally prohibitively expensive.

We thus consider an alternative formulation for consistency – for a given query workload, the goal is to minimize the L_2 distance between the true and estimated cardinalities. This approach also handles both cardinalities and distinct values counts. We propose an efficient one-pass algorithm with several theoretical properties modeling this formulation. Our experiments show that this approach produces similar improvements in accuracy as the EM based approach while being computationally significantly more efficient.

1. INTRODUCTION

Histograms are the primary data structures used in commercial query optimizers. They are meant to approximate the data distribution in the best manner possible within a bounded amount of space. Accordingly, several algorithms have been proposed for building histograms from the data [19].

It is often desirable to focus histograms toward a given workload of queries. For instance, if there are more sales queries about the New York region, we would desire better cardinality estimation for queries over this region. Since we are working within a sharply bounded space, histograms that seek to model the entire data distribution may have reduced accuracy for a specific region of the data. With this motivation, recent work [1, 23, 6, 30] has focused on the problem of *self-tuning* histograms where the idea is to bias the histogram toward a workload of queries. In this approach, statistics are gathered at a low overhead during query execution [31, 9] and folded into the histogram either online [1, 6, 23] or offline [30]. This way, we obtain higher accuracy precisely in those regions that are frequently queried by the given workload. Of course self-tuning histograms are inaccurate for parts of the data that are never queried but this problem can be mitigated by for example starting with a histogram that at least coarsely approximates the data. This approach is targeted toward data warehouse environments where the data can be assumed to be largely static.

Histograms are used not only for cardinality estimation but also for estimation of other statistics such as the number of distinct values and the number of distinct pages. Thus, histograms constructed from the data store not only cardinalities but also such additional statistics. On the other hand, all of the prior work on self-tuning histograms focuses exclusively on cardinalities. Therefore, histograms built over the data have the advantage of being able to model additional statistics. In this paper, we attempt to bridge this gap by designing a self-tuning histogram that models not only cardinalities but also distinct value counts.

The question arises whether it is even possible to monitor distinct value counts during query execution. There are many known single-pass techniques that approximate the number of distinct values in a data set [3]. These techniques make it feasible to obtain distinct value counts at a low overhead during query execution. Indeed, in this spirit recent work [9] has addressed the problem of obtaining distinct page counts at a low overhead as a part of execution feedback.

In designing such a self-tuning histogram, we seek a principled notion of *consistency* that formalizes how well the histogram models the information obtained from execution feedback. Recent work [30] has identified the Entropy Maximization principle for this purpose.

Accordingly, we start by applying the Entropy Maximization (EM) principle to the case when we have both cardinalities and distinct values. Capturing both cardinalities and distinct values introduces a new challenge since we need a different probability space than that considered in [30]. This leads to a more difficult EM problem. We show how the EM solution can be obtained by solving a non-linear system of equations with a number of variables proportional to the size of the workload. However, this system of equations is considerably more complex and therefore more difficult to solve than the corresponding one for cardinalities alone [30]. Indeed, we show empirically that the state of the art method for solving this system of equations is prohibitively expensive. For example, even for equations consisting of five variables, the running time is in the order of a few hours. Thus even though Entropy Maximization provides a principled method of constructing a self-tuning histogram, it does not yield an efficient self-tuning histogram in the presence of multiple statistics.

VLDB '08, August 24-30, 2008, Auckland, New Zealand

We therefore consider a different principle: to minimize the squared (L_2) distance between the histogram's estimates and the query feedback viewed as vectors. This measure of error has been successfully used in prior work on histogram construction [20, 29, 21, 16]. We call this self-tuning histogram L_2-Optimal. This approach models consistency even in the presence of multiple statistics (such as cardinalities and distinct value counts). The main challenge we address is to efficiently compute this histogram. We develop in this paper several advanced linear algebra techniques that allow us to compute the L_2-optimal histogram efficiently. In fact, we describe a single-pass algorithm for computing L_2-optimal histogram, which processes one query at a time and computes the histogram incrementally (Sections 4.1.2 and 4.2). This, of course, allows the L_2-optimal histogram to scale to larger workloads.

In summary, we make the following contributions in this paper.

- We present and solve the Entropy Maximization model for histograms with both cardinalities and distinct values (Section 3)
- We empirically show that the EM approach is prohibitively expensive (Section 3.3).
- We formulate a new notion of consistency based on the L_2 distance (Section 4.1).
- We describe a one-pass algorithm for computing the L_2-optimal histogram (Sections 4.1.2 and 4.2).
- We conduct an empirical study that shows that the L_2-optimal histogram yields significant benefits in query result size estimation just as the EM based histogram while being computationally much more efficient (Section 5).

2. PRELIMINARIES

We first describe our setting for self-tuning histograms and then introduce the basic definitions we use in the rest of the paper.

2.1 Setting

Prior work has considered two kinds of self-tuning histograms. On the one hand, we have *online* self-tuning histograms [1, 6, 23] that modify the histogram as queries execute in the database system. Given that this could impose significant overhead at the time of query execution, *offline* self-tuning histograms [30] have also been investigated where as queries execute, there is a very light-weight collection of execution feedback that is logged. This logged information is then used offline to build and modify a self-tuning histogram. In this paper, we focus on the offline setting.

To the best of our knowledge, there has been little prior work on storing distinct value estimates in multi-dimensional histograms. In fact, the number of column combinations for which distinct value estimates can be stored is exponential in the number of dimensions. We therefore focus on single-dimensional histograms in this paper.

Finally, for ease of exposition, we assume in this paper that the database is static. We do note that our techniques can be extended to handle database updates in ways similar to prior work [30].

2.2 Problem Statement

We assume that the database is a bag I of values from a finite, ordered domain $\mathcal{D}$ (the notations used in this paper are shown in Fig. 1). For any bag I, we denote $|I|$ its set size, and $||I||$ its bag size; e.g. for $I = \{a, a, b, c, c, c\}$, $|I| = 3$, $||I|| = 6$. A *cell*, or *bucket*, C is a set consisting of an interval $[a, b] \subseteq \mathcal{D}$, and we denote its volume as $\texttt{volume}(C)$; if the domain $\mathcal{D}$ is of integers, then $\texttt{volume}(C) = b - a + 1$.

$I, \mathcal{D}$	instance (a bag) and domain: $I \subseteq \mathcal{D}$
m, n	number of queries, number of buckets
i, j	indices: $i = 1, m$, $j = 1, n$
B, C	bucket, cell; $B, C \subseteq D$
S	statistics type: $S \in \{\texttt{set}, \texttt{bag}\}$
X_j	bucket value (set-size of bag-size)
r_i	record value (set-size of bag-size)
ρ_i	query statistic: $\rho_i = (C_i, r_i, S_i)$
f_i	weight of query ρ_i
W	workload of query statistics
μ	probability distribution
ω, α, β	parameters of the EM probability
$\mathbf{R}, \mathbf{X}$	column matrices (X_j), (r_i)
$\mathbf{Q}, \mathbf{F}$	query matrix ($m \times n$), diagonal weight matrix ($m \times m$)
$\mathbf{M}$	influence matrix ($n \times n$)
$\mathbf{p}$	statistics column matrix ($n \times 1$)

Figure 1: Notations used in the paper

In this paper we consider range queries with or without distinct values. More precisely, to each cell C we associate two queries: one that counts the number of elements in the instance I that are in C, called the *bag-size*, and the other that counts the number of distinct values in I that are in C, called the *set size*. We denote the answers to these two queries as:

$$\begin{aligned} \texttt{answer}_I^{\texttt{bag}}(C) &= ||I \cap C|| \\ \texttt{answer}_I^{\texttt{set}}(C) &= |I \cap C| \end{aligned}$$

Here the intersection of a bag I and a set C is a bag that preserves the multiplicities in I, e.g. $\{a, a, a, b, b, c\} \cap \{a, c\} = \{a, a, a, c\}$.

A *histogram*, H, consists of a partition of the space $\mathcal{D}$ into buckets $B_1, B_2, \ldots, B_n$, and two statistics per bucket B_i: a set-size $X_i^{\texttt{set}}$ and a bag-size $X_i^{\texttt{bag}}$. The histogram is used to estimate the set-size or the bag-size of a query associated to a cell C, as follows:

$$Est^S(C, H) = \sum_j \frac{\texttt{volume}(C \cap B_j)}{\texttt{volume}(B_j)} X_j^S \qquad (1)$$

for $S \in \{\texttt{set}, \texttt{bag}\}$. In particular, for every bucket B_j:

$$Est^S(B_j, H) = X_j^S$$

A *query statistic* (QS) is $\rho = (C, r, S)$ where C is a cell, $r = \texttt{answer}_I^S(C)$ is called the *record value*, and $S \in \{\texttt{set}, \texttt{bag}\}$ is the type of statistic. With some abuse of notation we write ρ as (C^S, r^S), to indicate that the cell and record value refer to the type S of statistic. When both set and bag statistics are obtained from the same query execution, we assume that two separate statistics are issued, one for the set size and one for the bag size. A *workload* W is a set $\{\rho_1, \ldots, \rho_m\}$, where each ρ_i is a query statistic.

We study in this paper the following problem: given a query workload W, find a histogram H that is *consistent* with the workload. Ideally, we would like H to satisfy the following for every query statistic $(C^S, r^S) \in W$:

$$Est^S(C, H) = r^S \qquad (2)$$

On the other hand, histograms are constrained to be small usually through an explicit space budget. Thus, perfect consistency of the form captured by the above equation may not be attainable. In this paper we discuss different ways of realistically modeling this consistency condition (2).

3. EM-MODEL FOR SET/BAG STATISTICS

A consistent histogram for bag-counts only is described in [30] based on the Entropy Maximization (EM) principle. This is a very general principle, and it raises the question whether the same principle can be applied to obtain consistent histograms for both bag-count and set-count statistics. We show that this is possible, by developing a more elaborate probability space.

To best understand our model it helps to briefly review the probabilistic model in [30]. There, the database instance I is fixed, and a single element is selected at random from the bag I. This defines a probability space whose outcomes are the elements from the domain $\mathcal{D}$: one outcome is one element. The bag-size of a bucket B is simply the probability that the output element belongs to B, times the cardinality of I, which is fixed and known. This model captures only one statistic: either the cardinality, or distinct values, but not both. To capture both, we need a probability space whose outcomes are instances I: then we can measure both the bag-size and set-size.

Let $\mathcal{I}$ be the set of all finite bags over the finite domain $\mathcal{D}$. This set is infinite (for example, it contains $\{a\}, \{a, a\}, \{a, a, a\}, \ldots$), and is countable. A *discrete probability space* over $\mathcal{I}$ is a function $\mu : \mathcal{I} \to [0, 1]$ such that

$$\sum_{I \in \mathcal{I}} \mu(I) \quad = \quad 1 \tag{3}$$

The expected set-size, bag-size, and the entropy are defined as:

$$\begin{aligned} E_\mu[|I|] &= \sum_{I \in \mathcal{I}} |I| \mu(I) \\ E_\mu[||I||] &= \sum_{I \in \mathcal{I}} ||I|| \mu(I) \\ H(\mu) &= -\sum_{I \in \mathcal{I}} \mu(I) \log \mu(I) \end{aligned}$$

Note that, in general, these sums may diverge: throughout the paper we consider only probability spaces where all three sums converge. We define the answer to the set-size and bag-size query associated to a cell C to be:

$$\begin{aligned} \texttt{answer}_\mu^{\texttt{bag}}(C) &= E[||I \cap C||] \\ \texttt{answer}_\mu^{\texttt{set}}(C) &= E[|I \cap C|] \end{aligned} \tag{4}$$

In this approach, we model the consistency condition (2) as follows:

DEFINITION 3.1. *Let W be a workload. A probability space μ* is consistent *with the workload W if for every query statistic $(C^S, r^S) \in W$, $\texttt{answer}_\mu^S(C^S) = r^S$.*

Note that once we have a consistent probability space μ then we can compute a consistent histogram H over a given set of buckets $B_1, \ldots, B_n$ as follows: associate to each B_i the statistic $X_i^S = \texttt{answer}_\mu^S(B_i)$, for $S \in \{\texttt{set}, \texttt{bag}\}$. Assume that the buckets are chosen such that they partition every cell C^S of the query workload, i.e. $\bigcup_{B_i \subseteq C^S} B_i = C^S$: then consistency condition Eq (2) holds in expectation. Indeed, in this case for every query statistic $(C^S, r^S) \in W$ we have $Est^S(C^S, H) = \sum_{B_i \subseteq C^S} \texttt{answer}_\mu^S(B_i) = \texttt{answer}_\mu^S(C)$.

Thus, in this section we study the following problem: given a workload W, find a consistent probability space μ. Note that, in general, there may be no solution at all. Worse, when a solution exists, in general there are many (infinitely) solutions for μ, and we need to choose one without making ad-hoc assumptions about the data.

The *entropy-maximization* principle states that this is achieved by the probability distribution that has the maximum entropy.

DEFINITION 3.2. *A probability space μ is an* EM-model*, or an EM-solution for a workload W, if (1) it is consistent with the workload, and (2) if μ' is any probability space that is consistent with the workload, then $H(\mu) \geq H(\mu')$.*

We show in this section how to obtain the EM-model for a workload W. We start by reviewing a classic result for the EM model [22]. Here, and in the sequel, we denote $m^{\texttt{set}}$ the number of set statistics in W, and $m^{\texttt{bag}}$ the number of bag statistics.

THEOREM 3.3. *If μ is an EM-model for a workload W, then the following property holds:*

$(*)$ *There exists $1 + m^{set} + m^{bag}$ positive constants ω, α_i, β_j, where $i = 1, m^{set}$ and $j = 1, m^{bag}$, such that $\forall I \in \mathcal{I}$:*

$$\mu(I) \quad = \quad \omega \prod_{i=1,m^{set}} \alpha_i^{|I \cap C_i^{set}|} \prod_{j=1,m^{bag}} \beta_j^{||I \cap C_j^{bag}||} \tag{5}$$

Conversely, if a probability space μ has property $(*)$ and is consistent with the workload W, then it is an EM model for W.

We refer to [22] for the proof. The result does not give us a solution to the EM model, and does not tell us when such a solution exists, but it restricts our search for an EM solution to probability functions that have a particular, simple expression. We illustrate with an example.

Example 3.4 Consider the domain $D = [1, 100]$, and assume a workload with three statistics: $([1, 100], r_1, \texttt{bag})$, $([1, 75], r_2, \texttt{set})$, $([26, 100], r_3, \texttt{set})$. Thus, the expected number of values in the bag is r_1, the expected number of distinct values in $[1, 75]$ is r_2 and in $[26, 100]$ is r_3. Then, the EM model, if it exists, must have the following form:

$$\mu(I) \quad = \quad \omega \alpha_2^{|I \cap [1,75]|} \alpha_3^{|I \cap [26,100]|} \beta_1^{||I||}$$

for some positive constants $\omega, \alpha_2, \alpha_3, \beta_1$.

We show next how to compute the parameters of the EM model, in two steps.

3.1 Partitioned Workloads

A *partitioned workload* is a workload whose cells $B_1, \ldots, B_m$ form a partition of the domain, and there is exactly one set-size and one a bag-size statistic for each bucket, $r_i^{\texttt{set}}$ and $r_i^{\texttt{bag}}$, for $i = 1, m$.

We first solve the EM-model for the case $m = 1$, called the *single bucket workload*. Thus $W = \{(\mathcal{D}, r^{\texttt{set}}), (\mathcal{D}, r^{\texttt{bag}})\}$ and we need to find μ that satisfies:

$$E_\mu[|I|] = r^{\texttt{set}} \qquad E_\mu[||I||] = r^{\texttt{bag}} \tag{6}$$

and has the maximum entropy. By Theorem 3.3 the function μ is given by $\mu(I) = \omega \alpha^{|I|} \beta^{||I||}$, where ω, α, β are three positive unknowns: we need to find these unknowns such that Equations (6) hold. Denote:

$$N = |\mathcal{D}| \quad p = \frac{r^{\texttt{set}}}{N} \quad f = \frac{r^{\texttt{bag}}}{r^{\texttt{set}}}$$

N is the size of the domain, and we call f the *fanout*. The value p has an interesting interpretation: it is the probability that some fixed value v in the domain belongs to the random instance I, in notation $\mu(v \in I)$. To see this, notice that we can express the expected size of I as $E[|I|] = \sum_{v \in \Delta} \mu(v \in I) = N\mu(v \in I)$, since the probability $\mu(v \in I)$ is the same for all constants $v \in \mathcal{D}$. Since the expected set-size of I is $r^{\texttt{set}}$, we obtain $\mu(v \in I) = r^{\texttt{set}}/N = p$.

THEOREM 3.5. *If $p < 1 < f$ then the EM-model for the single bucket workload has a unique solution given by:*

$$\omega = (1-p)^N \quad \alpha = \frac{p}{(f-1)(1-p)} \quad \beta = 1 - \frac{1}{f}$$

Otherwise, the EM-model has no solution.

Next we turn to partitioned workloads with $m \geq 1$. In this case it can be shown that the EM-model consists of m independent EM models, one for each bucket. More precisely, denoting:

$$N_i = |B_i| \quad p_i = \frac{r_i^{\texttt{set}}}{N_i} \quad f_i = \frac{r_i^{\texttt{bag}}}{r_i^{\texttt{set}}} \tag{7}$$

for each bucket $i = 1, m$, we have:

THEOREM 3.6. *If for all $i = 1, m$, $p_i < 1 < f_i$, then the EM-model for the partitioned workload is given by $1 + 2m$ parameters $\omega, \alpha_i, \beta_i$, where $\omega = \prod_{i=1m} \omega_i$ and for every $i = 1, m$:*

$$\omega_i = (1-p_i)^{N_i} \quad \alpha_i = \frac{p_i}{(f_i-1)(1-p_i)} \quad \beta_i = 1 - \frac{1}{f_i} \tag{8}$$

Otherwise, the EM model has no solution.

3.2 EM-Model for General Workload

Now we consider a general workload $\bar{W}$ over arbitrary cells with set-size statistics $(\bar{C}_i^{\texttt{set}}, \bar{r}_i^{\texttt{set}})$, $i = 1, \bar{m}^{\texttt{set}}$, and bag-size statistics $(\bar{C}_j^{\texttt{bag}}, \bar{r}_j^{\texttt{bag}})$, $j = 1, \bar{m}^{\texttt{bag}}$; we overline this workload, $\bar{W}$, to distinguish it from a partitioned workload W, which we construct shortly, and assume $\mathcal{D} = \bigcup \bar{C}_i^{\texttt{set}} = \bigcup \bar{C}_i^{\texttt{bag}}$. The difficulty here is that the cells may be overlapping, and the EM model no longer consists of independent models. We show that here the solution to the EM model is given by the solution of a certain partitioned workload W, which satisfies some additional constraints. We describe the partitioned workload next.

Let $B_1, \ldots, B_l$ ($l \geq m$) be the coarsest partition of the domain that is a refinement of all the cells in the workload $\bar{W}$; more precisely, for every cell $\bar{C}_j$ in $\bar{W}$ and every bucket B_i in W, either $B_i \subseteq \bar{C}_j$ or $B_i \cap \bar{C}_j = \emptyset$. Next, for each bucket B_i we introduce two variables, $r_i^{\texttt{set}}$ and $r_i^{\texttt{bag}}$, representing the (unknown) set-size and bag-size statistics for B_i, subject to the constraints:

$$\sum_{i:B_i \subseteq \bar{C}_j^{\texttt{set}}} r_i^{\texttt{set}} = \bar{r}_j^{\texttt{set}} \qquad \sum_{i:B_i \subseteq \bar{C}_j^{\texttt{bag}}} r_i^{\texttt{bag}} = \bar{r}_j^{\texttt{bag}} \tag{9}$$

Define the partitioned workload $W = \{(B_i, r_i^{\texttt{set}}) \mid i = 1, l\} \cup \{(B_i, r_i^{\texttt{bag}}) \mid i = 1, l\}$. Consider the associated EM-model, and denote $\omega, \alpha_i, \beta_i$, $i = 1, l$ its parameters, which are related to the unknowns $r_i^{\texttt{set}}, r_i^{\texttt{bag}}$, through Eq.(8).

THEOREM 3.7. *The EM-model for a general workload $\bar{W}$ has the parameters $\omega, \bar{\alpha}_j, \bar{\beta}_j$ obtained by solving simultaneously the following system of non-linear equations: Equations (8), (9), and the equations below:*

$$\alpha_i = \prod_{j:B_i \subseteq \bar{C}_j^{\texttt{set}}} \bar{\alpha}_j \qquad \beta_i = \prod_{j:B_i \subseteq \bar{C}_j^{\texttt{set}}} \bar{\beta}_j \tag{10}$$

We can use this theorem to solve the EM model as follows. Equation (8) yields $2l + 1$ equations since the equations for ω_i can be folded into a single equation for ω. We can see that Equation (9) yields $2m$ equations. Finally, Equation (10) also yields $2l$ equations. Thus, we have $2m + 4l + 1$ equations. The number of unknowns adds up as follows — there are $2l$ unknowns corresponding to α_i, β_i (parameters associated with B_i), $2m$ unknowns corresponding to $\bar{\alpha}_j, \bar{\beta}_j$ (parameters associated with cell $\bar{C}_j$), $2l$ parameters $r_i^{\texttt{set}}, r_i^{\texttt{bag}}$, and finally ω. Observe that we are not counting p_i and f_i since they can be directly expressed in terms of $r_i^{\texttt{set}}, r_i^{\texttt{bag}}$. We thus have $2m + 4l + 1$ unknowns. This system of non-linear equations needs to be solved to obtain the EM model.

Example 3.8 Consider the domain $\mathcal{D} = [1, 100]$ and the two overlapping cells $C_1 = [1, 75]$, $C_2 = [26, 100]$. We illustrate with an example with a workload $\bar{W}$ consisting of four statistics:

$$(C_1, \texttt{set}, \bar{r}_1^{\texttt{set}}) \quad (C_2, \texttt{set}, \bar{r}_2^{\texttt{set}}) \quad (C_1, \texttt{bag}, \bar{r}_1^{\texttt{bag}}) \quad (C_2, \texttt{bag}, \bar{r}_2^{\texttt{bag}})$$

We need to find five unknowns $\omega, \alpha_1, \alpha_2, \beta_1, \beta_2$. For that we associate the partitioned workload W defined for the following three buckets: $B_1 = [1, 25]$, $B_3 = [26, 75]$, $B_2 = [76, 100]$. We introduce twelve variables $\alpha_j, \beta_j, r_j^{\texttt{set}}, r_j^{\texttt{bag}}$, $j = 1, 2, 3$, and solve the following seventeen equations, where $N_1 = 25, N_2 = 50, N_3 = 25$ are the sizes of the three buckets, and

$$\begin{array}{ll}
\omega = \prod_{j=1,3}(1-p_j)^{N_j} & \\
\alpha_1 = \frac{p_i}{(f_1-1)(1-p_1)} & \beta_1 = 1 - \frac{1}{f_1} \\
\alpha_2 = \frac{p_i}{(f_2-1)(1-p_2)} & \beta_2 = 1 - \frac{1}{f_2} \\
\alpha_3 = \frac{p_i}{(f_3-1)(1-p_3)} & \beta_3 = 1 - \frac{1}{f_3} \\
\alpha_1 = \bar{\alpha}_1 & \beta_1 = \bar{\beta}_1 \\
\alpha_2 = \bar{\alpha}_2 & \beta_2 = \bar{\beta}_2 \\
\alpha_3 = \bar{\alpha}_1\bar{\alpha}_2 & \beta_3 = \bar{\beta}_1\bar{\beta}_2 \\
\bar{r}_1^{\texttt{set}} = r_1^{\texttt{set}} + r_3^{\texttt{set}} & \bar{r}_1^{\texttt{bag}} = r_1^{\texttt{bag}} + r_3^{\texttt{bag}} \\
\bar{r}_2^{\texttt{set}} = r_2^{\texttt{set}} + r_3^{\texttt{set}} & \bar{r}_2^{\texttt{bag}} = r_2^{\texttt{bag}} + r_3^{\texttt{bag}}
\end{array}$$

p_j, f_j are expressions in the unknowns $r_j^{\texttt{set}}, r_j^{\texttt{bag}}$ (Eq.(7)).

3.3 Capturing the Space Budget

The histograms we seek typically have space budgets which are explicitly provided. Note that the EM model as described so far yields a histogram that is a refinement of all the workload cells. We thus need techniques to compress the histogram. Here we follow a method similar to [30] where we use the solutions to the unknowns $\bar{\alpha}_j, \bar{\beta}_j$ associated with cell $\bar{C}_j$ to derive a measure of the importance of this statistic. We then drop statistics that are of lower importance until the space budget can be met.

3.4 Solving the Non-Linear System

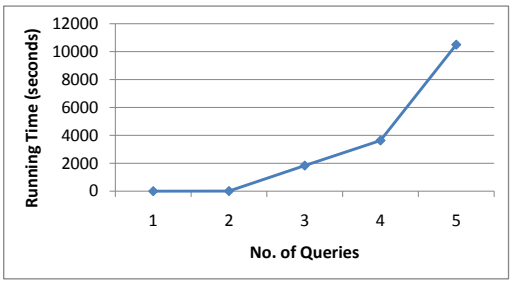


Figure 2: EM Running Time

The system of equations implied by Theorem 3.7 illustrated for Example 3.8 above is quite unlike the system obtained for the EM model in the presence of cardinality constraints alone [30]. It is not clear if methods such as iterative scaling can be used with any guarantee of convergence for the non-linear system we obtain. We did implement an adaptation of iterative scaling [27] but found it to not converge to any meaningful result. The state of the art method that solves non-linear equations of the form we obtain with good convergence properties is homotopy continuation [18]. Unfortunately,

we find that this approach scales very poorly with the number of variables. We illustrate here the running time of this method with increasing number of query statistics fed (which corresponds to increasing number of variables) — the data and the rest of the experimental setup is explained in Section 5. The X-axis plots the number of query statistics input and the Y-axis, the running time to solve the non-linear system using homotopy continuation [18]. We can see that the running times increase sharply with the number of queries reaching the order of hours even for 5 queries, whereas we are interested in processing workloads of hundreds or even thousands of queries. This makes this approach prohibitively expensive.

Solving non-linear systems of equations is an active area of research and it is possible that practical methods to solve systems of the form we generate will be discovered in the future. However, given the state of the art, it is not clear how the EM model can be applied for consistent histogram construction in the presence of distinct value counts. For these reasons we describe next a completely different approach for constructing histograms for a workload W.

4. CONSISTENCY USING L_2 DISTANCE

In this section we describe an alternative construction of an adaptive histogram. Instead on insisting that the histogram H be consistent for a workload W (Equation (2)), we aim at reducing the squared, or L_2, error between the statistics in the workload and those estimated by the histogram, viewed as vectors. This measure of error has been successfully used in prior work on histogram construction [20, 29, 21, 16].

More precisely, given a histogram H and query statistic $\rho = (C^S, r^S)$, define the error of query ρ:

$$\texttt{Err}_\rho(H) \quad = \quad (Est^S(C^S, H) - r^S)^2$$

Consider a workload of query statistics $W = \{\rho_1, \ldots, \rho_m\}$, and assume we have an optional sequence of positive weights $f_1, f_2, \ldots, f_m$ (by default $f_1 = f_2 = \ldots = f_m = 1$). Given a histogram H, the error for the entire workload W is

$$\texttt{Err}_W(H) \quad = \quad \sum_{i=1,m} f_i \cdot \texttt{Err}_{\rho_i}(H) \qquad (11)$$

DEFINITION 4.1. *Let W be a workload, and n be a number. The L_2-optimal histogram for the workload W and the budget n is a histogram H s.t. (a) H has at most n buckets, and (b) $\texttt{Err}_W(H)$ is minimal among all histograms with at most n buckets.*

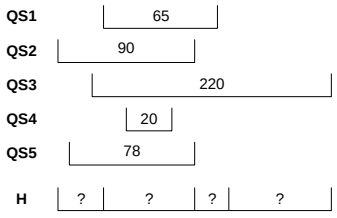


Figure 3: Illustration of the L_2-optimality problem.

Figure 3 illustrates the setting of the L_2-optimal histogram: there are five queries in the workload W (suppose all five queries are bag-statistics), and we have a budget of four buckets. The goal is to determine the bucket boundaries, and the bucket values of a histogram H s.t. $\texttt{Err}_W(H)$ is minimal. Note that the problem is over specified: there are more queries than buckets. As we will see, the L_2-optimal histogram is well defined only if the problem is usually over-specified; by contrast, the EM-histogram is defined only if the problem is under-specified. In practice the problem is over-specified, because there are far more queries than buckets in a histogram.

4.1 Fixed Size Buckets

We start our investigation by assuming that the buckets are fixed: we denote n_0 the number of buckets, and $B_1, \ldots, B_{n_0}$ the actual buckets. After processing the workload W, we need to compute the bucket contents X_j^S, one for each bucket B_j and each statistic S: we do not change the bucket cell B_j, or the number of buckets. This is a least-square problem with $n = 2n_0$ unknowns, $X_j^{\texttt{set}}$ and $X_j^{\texttt{bag}}$, $j = 1, n_0$. To formulate the least square problem using standard linear algebra terminology, we consider a single column matrix $\mathbf{X} = (X_j)_{j=1,n}$, of size $n \times 1$, where $X_j = X_j^{\texttt{set}}$, $X_{j+n} = X_j^{\texttt{bag}}$ for $j \le n_0$. With this translation in mind, we denote:

$$B_j^{\texttt{set}} \quad = \quad \begin{cases} B_j & \text{if } j \le n_0 \\ \emptyset & \text{if } n_0 < j \le n \end{cases}$$

$$B_j^{\texttt{bag}} \quad = \quad \begin{cases} \emptyset & \text{if } j \le n_0 \\ B_{j-n} & \text{if } n_0 < j \le n \end{cases}$$

Given a workload $W = \{(C_1, r_1, S_1), \ldots, (C_m, r_m, S_m)\}$ and weights $f_1, \ldots, f_m$, we define:

$$\begin{aligned} \mathbf{Q} \quad &= \quad (q_{ij})_{i=1,m;j=1,2n} \text{ the } m \times 2n \text{ query matrix} \\ &\quad \text{where } q_{ij} = \frac{\texttt{volume}(C_i \cap B_j^{S_i})}{\texttt{volume}(B_j^{S_i})} \\ \mathbf{F} \quad &= \quad \texttt{diag}(f_1, \ldots, f_m) \text{ the diagonal weight matrix} \\ \mathbf{r} \quad &= \quad (r_i)_{i=1,m} \text{ the column vector of record values} \end{aligned}$$

Combining Equations (11) and (1) we obtain the following expression for the error:

$$\texttt{Err}_W(\mathbf{X}) \quad = \quad (\mathbf{QX} - \mathbf{r})^T \mathbf{F} (\mathbf{QX} - \mathbf{r}) \qquad (12)$$

The problem becomes: given the workload $(\mathbf{Q}, \mathbf{F}, \mathbf{r})$ compute $\mathbf{X}$ that minimizes the error expression (12). This is a Least Square problem, and has an explicit solution, which we give after introducing the following notation:

$$\begin{aligned} \mathbf{M} \quad &= \quad \mathbf{Q}^T \mathbf{F} \mathbf{Q} \quad \text{the influence matrix, } n \times n \\ \mathbf{p} \quad &= \quad \mathbf{Q}^T \mathbf{F} \mathbf{r} \quad \text{the statistics vector, } n \times 1 \end{aligned}$$

4.1.1 Offline Algorithm

The offline algorithm follows from a standard result in linear algebra:

THEOREM 4.2. *Then the L_2-optimal histogram is given by any vector X that satisfies:*

$$\mathbf{MX} \quad = \quad \mathbf{p} \qquad (13)$$

In particular, if $\mathbf{M}$ is non-singular, then the W-histogram is:

$$\mathbf{X} \quad = \quad \mathbf{M}^{-1}\mathbf{p} \qquad (14)$$

□

The theorem gives a naive approach to compute the L_2-optimal histogram: if $\mathbf{M}$ is non-singular, then solve Eq. (14), which gives the unique solutions; otherwise, if the matrix is singular, then pick any solution to (13) (which can be shown to always have a solution)

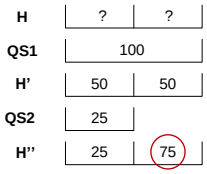


Figure 4: The need to remember the history. After processing the first query, the histogram is $(50, 50)$. The second query tells us that there are 25 tuples in B_1. Because of the influence reflected in QS1, in addition to updating the first bucket to 25 we also need to update the second bucket to 75.

because it is guaranteed to be an L_2-optimal histogram. We discuss how we maintain the non-singularity of $\mathbf{M}$ in Section 4.1.3.

In the rest of this section we discuss how to improve this naive algorithm, but first we discuss the intuition behind the influence matrix $\mathbf{M}$ and statistics vector $\mathbf{r}$.

The influence matrix stores an "influence" value between each pair of buckets in the histogram; the statistics vector adds for each bucket all query results that affect that bucket:

$$M_{jk} = \sum_{i=1,m} f_i q_{ij} q_{ik}$$
$$p_j = \sum_{i=1,m} f_i q_{ij} r_i$$

The simplest case is when $f_1 = \ldots = f_m = 1$ and every query either completely covers a bucket or is disjoint form that bucket ($q_{ij} = 1$ or $q_{ij} = 0$). In that case M_{jk} represents the number of queries in the workload that are common to both buckets j and k. Thus, M_{jk} stands for the number of common queries between buckets j and k, hence "influence".

4.1.2 Online Algorithm

Next, we describe an online algorithm for computing the L_2-optimal histogram: given the L_2-optimal histogram $\mathbf{X}$ for a workload W and a new query statistic $\rho = (C, r, S)$, we will compute the new L_2-optimal histogram $\mathbf{X}'$ for the extended workload $W \cup \{\rho\}$. The challenge is to compute $\mathbf{X}'$ by inspecting only $\mathbf{X}$ and ρ, and not the prior workload W. In general this is not possible, as shown in the following example.

Example 4.3 Consider the histogram in Fig. 4. The domain $\mathcal{D} = [a, b]$ is partitioned in two equal buckets, $B_1 = [a, \frac{a+b}{2}]$, $B_2 = [\frac{a+b}{2}, b]$, thus $\texttt{volume}(B_1) = \texttt{volume}(B_2)$. There are two query statistics, saying:

QS1: "There are $r_1 = 100$ tuples in $B_1 \cup B_2$".

QS2: "There are $r_2 = 25$ tuples in B_1".

The L_2-optimal histogram minimizes $(X_1 + X_2 - r_1)^2 + (X_1 - r_2)^2$. From Theorem 4.2 we obtain:

$$\mathbf{Q} = \begin{pmatrix} 1 & 1 \\ 1 & 0 \end{pmatrix} \quad \mathbf{M} = \begin{pmatrix} 2 & 1 \\ 1 & 1 \end{pmatrix} \quad \mathbf{p} = \begin{pmatrix} 125 \\ 100 \end{pmatrix}$$

Then Eq. (13) is $2X_1 + X_2 = 125$ and $X_1 + X_2 = 100$, which solves to $X_1 = 25, X_2 = 75$.

But now let's examine what happens if we attempt to compute the histogram incrementally. If we start from QS1, then we first obtain a singular matrix $\mathbf{M} = \begin{pmatrix} 1 & 1 \\ 1 & 1 \end{pmatrix}$, while the statistics vector is $\mathbf{p} = \begin{pmatrix} 100 \\ 100 \end{pmatrix}$. The system $\mathbf{MX} = \mathbf{p}$ is under-specified; assume we decide to spread the 100 tuples evenly across B_1 and B_2, hence $X_1 = X_2 = 50$. Next we have to process QS2, which says "there are 25 tuples in B_1". If we ignore the history (which is here QS1) then we cannot achieve the same correct answer as in the offline approach: depending on whether we ignore the current bucket content X_1 or include in the Least Square calculation, we arrive at either $X_1 = 25$, $X_2 = 50$ or at $X_1 = 37.5$, $X_2 = 50$. In either cases we fail to update X_2. In order to compute the correct L_2-optimal histogram we need to remember that there is an "influence" induced by the QS1 between B_1 and B_2, which should lead us to increase X_2 whenever we decrease X_1. It is important to store this influence in order to compute the histogram online. □

Thus, we cannot ignore the entire history W. However, instead of storing W, we store only $\mathbf{M}$ and $\mathbf{p}$: the space requirement is $O(n^2)$ (which is fixed for the histogram), and is typically is much smaller than the space $O(m)$ needed for W. We will show next how to maintain $\mathbf{M}$ and $\mathbf{p}$ incrementally.

We need some notation. Let $\mathbf{M}$, $\mathbf{p}$, and $\mathbf{X}$ correspond to the past workload W, with m queries. Let ΔW be the incremental workload, consisting of Δm queries. Denote $\mathbf{M}'$, $\mathbf{p}'$, and $\mathbf{X}'$ the values corresponding to the combined workload $W' = W \cup \Delta W$.

PROPOSITION 4.4. *Consider the matrix representation of ΔW:*

$$\Delta W = (\Delta\mathbf{Q}, \Delta\mathbf{F}, \Delta\mathbf{r})$$

Then:

$$\mathbf{M}' = \mathbf{M} + (\Delta\mathbf{Q})^T \Delta\mathbf{F} \Delta\mathbf{Q} \quad (15)$$
$$\mathbf{p}' = \mathbf{p} + (\Delta\mathbf{Q})^T \Delta\mathbf{F} \Delta\mathbf{r} \quad (16)$$

Thus, $\mathbf{M}'$ and $\mathbf{p}'$ can be computed incrementally, in time $O(n^2 + n(\Delta m)^2)$, and the new L_2-optimal histogram is obtained by solving:

$$\mathbf{M}'\mathbf{X}' = \mathbf{p}'$$

The proof is omitted for lack of space.

Example 4.5 Continuing the Example 4.3, we show how to process the two QSs online. The first QS has $q_{11} = q_{12} = 1$, $r_1 = 100$, and the influence matrix and statistics vector are:

$$\mathbf{M} = \begin{pmatrix} 1 \\ 1 \end{pmatrix} \begin{pmatrix} 1 \end{pmatrix} \begin{pmatrix} 1 & 1 \end{pmatrix} = \begin{pmatrix} 1 & 1 \\ 1 & 1 \end{pmatrix}$$
$$\mathbf{p} = \begin{pmatrix} 1 \\ 1 \end{pmatrix} \begin{pmatrix} 1 \end{pmatrix} (100) = \begin{pmatrix} 100 \\ 100 \end{pmatrix}$$

Note that $\mathbf{M}$ is singular. The equation $\mathbf{MX} = \mathbf{p}$ becomes $X_1 + X_2 = 100$, which is non-determined. We discuss below how to address this in general; here we simply choose $X_1 = X_2 = 50$, thus deciding to split the 100 tuples evenly between the two buckets. Next, we process the second QS, where $q_{21} = 1$, $q_{22} = 0$, $r_2 = 25$. We compute the new influence matrix and statistics vector incrementally:

$$\mathbf{M}' = \mathbf{M} + \begin{pmatrix} 1 \\ 0 \end{pmatrix} \begin{pmatrix} 1 \end{pmatrix} \begin{pmatrix} 1 & 0 \end{pmatrix} = \begin{pmatrix} 2 & 1 \\ 1 & 1 \end{pmatrix}$$
$$\mathbf{p}' = \begin{pmatrix} 1 & 1 \\ 1 & 1 \end{pmatrix} \begin{pmatrix} 50 \\ 50 \end{pmatrix} + \begin{pmatrix} 1 \\ 0 \end{pmatrix} \begin{pmatrix} 25 \end{pmatrix} = \begin{pmatrix} 125 \\ 100 \end{pmatrix}$$

Algorithm 4.1 Online L_2-`Optimal` Algorithm for Fixed Buckets

Input: $\mathbf{M}, \mathbf{p}, \mathbf{X}, (\mathbf{M})^{-1}$ for some workload W
A new query statistic $\rho = (q, f, r)$
Output: $\mathbf{M}', \mathbf{p}', \mathbf{X}', (\mathbf{M}')^{-1}$ for the workload $W' = W \cup \{\rho\}$

for $j = 1, n; k = 1, n$ **do**
  $M'_{jk} = M_{jk} + q_j f q_k$
end for
for $j = 1, n$ **do**
  $p'_j = p_j + f r q_j$
end for
for $j = 1, n$ **do**
  $a_j = \sum_{k=1,n} q_k (M^{-1})_{kj}$
end for
$g = f \sum_i q_i^2$
for $j = 1, n; k = 1, n$ **do**
  $((M')^{-1})_{jk} = (M^{-1})_{jk} - a_j a_k / g$
end for
for $j = 1, n$ **do**
  $X'_j = \sum_k ((M')^{-1})_{jk} p'_k$
end for

The new state of the histogram is given by the solution to the equations $2X_1 + X_2 = 125$ and $X_1 + X_2 = 100$, which is $X_1 = 25, X_2 = 75$. This is the same as the offline histogram in Example 4.3. □

Once we determine $\mathbf{M}'$ and $\mathbf{p}'$ we still need to solve the system $\mathbf{M}'\mathbf{X}' = \mathbf{p}'$. First we will assume that $\mathbf{M}$ is non-singular, and show how to compute its inverse incrementally; then we discuss how to ensure that it is indeed non-singular.

THEOREM 4.6. *Assume* $\mathbf{M}$ *is non-singular and denote:*

$$\begin{aligned} \mathbf{A} &= \Delta\mathbf{Q}(\mathbf{M})^{-1} : \Delta m \times n \\ \mathbf{G} &= (\Delta\mathbf{F})^{-1} + \Delta\mathbf{Q}\mathbf{M}^{-1}(\Delta\mathbf{Q})^T : \Delta m \times \Delta m \end{aligned}$$

Then (1) $\mathbf{G}$ *is nonsingular. (2) The influence matrix* $\mathbf{M}'$ *is non-singular, and its inverse is given by:*

$$(\mathbf{M}')^{-1} = \mathbf{M}^{-1} - \mathbf{A}^T\mathbf{G}^{-1}\mathbf{A} \quad (17)$$

In particular, $(\mathbf{M}')^{-1}$ *can be computed in time* $O(n^2 + n(\Delta m)^2 + (\Delta m)^3)$.

PROOF. (Sketch) For (1) we note that the quadratic form associated to the symmetric matrix $\mathbf{G}$ is positively defined, because it is the sum of $(\Delta\mathbf{F})^{-1}$, which is positively defined[1], and $\Delta\mathbf{Q}\mathbf{M}^{-1}(\Delta\mathbf{Q})^T$, which is semi-positively defined. (2) is Woodbury's matrix identity. □

4.1.3 Non-Singularity of $\mathbf{M}$

Now we explain how we ensure that $\mathbf{M}$ is non-singular. First, we initialize the self-tuning histogram such that the first value of $\mathbf{M}$ is non-singular. We do this by starting with a workload of exactly n queries, one per bucket, which amounts to computing the entire histogram offline. After this initialization phase, $\mathbf{M} = \mathbf{I}_m$ (the identity matrix), hence it is non-singular. Next, we process workloads incrementally, either one query at a time or in small batches ΔW. Theorem 4.6 guarantees that $\mathbf{M}$ continues to be non-singular.

Our discussion leads Algorithm 4.1, where we process one new query at a time, i.e. $\Delta m = 1$.

THEOREM 4.7. *Algorithm 4.1 maintains the* L_2*-optimal histogram among the class of histograms with a fixed set of* n *buckets online using* $O(n^2)$ *space and* $O(n^2)$ *time per QS.* □

[1] $\Delta\mathbf{F}$ is diagonal, and all its diagonal elements are > 0.

4.2 Adjusting Bucket Boundaries

So far we have assumed that the bucket boundaries are fixed. In practice we need to adjust them, in response to the queries in the workload. Given bucket base $\bar{B} = \{B_1, \ldots, B_n\}$ and a query statistic $\rho = (C, r, S)$, we allow the histogram to change its bucket base to $\bar{B}' = \{B'_1, \ldots, B'_n\}$, as follows. Define *candidate bucket* to be a bucket obtained from $\bar{B} \cup \{C\}$ using set intersections, unions, and differences. For example, given ten buckets $\bar{B} = \{[1, 10], [11, 20], \ldots, [91, 100]\}$ and the cell $[26, 45]$, there are 66 candidate buckets, namely all intervals of the form $[x+1, y]$, where $x, y \in \{0, 10, 20, 25, 30, 40, 45, 50, \ldots, 100\}$. We restrict the search space to histograms H' whose bucket base consists only of candidate buckets. We denote $\mathcal{B}$ the set of bucket bases that form our search space: the problem is to find a histogram H' s.t. $\texttt{buckets}(H') \in \mathcal{B}$ and $\texttt{Err}_W(H')$ is minimal. Our search algorithm restricts the search space $\mathcal{B}$ by using a greedy algorithm: start with the histogram containing all minimal candidate buckets, then search histograms with a smaller number of buckets, by merging adjacent buckets: $\mathcal{B}$ denotes the set of histograms considered by the algorithm. We give the details in the remainder of this section.

We assume the following two operations on the cells:

- $\texttt{split}_C(B) = \{B_0, \ldots, B_k\}$ partitions the cell B by the cell C. The result is a set of $k+1$ disjoint cells $B_0, \ldots, B_k$ s.t. $B = B_0 \cup \ldots \cup B_k$ and forall i, either $B_i \subseteq C$ or $B_i \cap C = \emptyset$.
- $\texttt{merge}(B_0, \ldots, B_k)$ returns either $B_0 \cup \ldots \cup B_k$, if this is a valid cell, or is undefined otherwise.

Since our domain is one dimensional, $\texttt{split}_C(B)$ returns 1, 2, or 3 intervals. For example:

$$\begin{aligned} \texttt{split}_{[5,6]}([1,4]) &= \{[1,4]\} \\ \texttt{split}_{[3,5]}([1,4]) &= \{[1,3],[3,4]\} \\ \texttt{split}_{[2,3]}([1,4]) &= \{[1,2],[2,3],[3,4]\} \end{aligned}$$

The online algorithm is shown in Algorithm 4.2. First it splits all histogram buckets that intersect with the query, thus creating a larger histogram H'', which may exceed the number of buckets allowed by our budget: Sec 4.2.2 describes how to compute H''. Next, it adds the query statistic QS to H'', without further changing the bucket boundaries of H'': for this it uses Algorithm 4.1. Finally, it greedily merges buckets in H'' to decrease them to $\leq n$: Sec. 4.2.3 describes the matrix manipulation for the merge phase. The result consists of the new histogram H', and its associated data: $\mathbf{M}', (\mathbf{M}')^{-1}, \mathbf{p}', \mathbf{X}'$. We prove the following nontrivial fact in the remainder of this section:

THEOREM 4.8. *Algorithm 4.2 runs in time* $O(n^2 + n|\mathcal{B}|)$.

The importance of this result is that we need to spent only $O(n)$ per candidate solution inspected, plus an extra global time $O(n^2)$. The crux of the theorem consists of several matrix manipulation techniques that avoid completely the need to compute an inverse matrix, and further reduce the matrix manipulation complexity. We describe these next.

4.2.1 Bucket Transformations

DEFINITION 4.9. *A* bucket transformation *is a pair of bucket basis:* $\bar{B} = \{B_1, \ldots, B_n\}$ *and* $\bar{B}' = \{B'_1, \ldots, B'_p\}$. *The* bucket transformation matrix, $\mathbf{S}$, *is:*

$$S_{jk} = \texttt{volume}(B'_k \cap B_j)/\texttt{volume}(B'_k)$$

Algorithm 4.2 Online Algorithm for Variable Buckets

Input: A histogram H, and it's matrices $\mathbf{M}, \mathbf{M}^{-1}, \mathbf{p}, \mathbf{X}$;
A new query statistic: (C, r).
Output: A new histogram H', and $\mathbf{M}', (\mathbf{M}')^{-1}, \mathbf{p}', \mathbf{X}'$
1: Split Phase (Sec. 4.2.2)
Compute $H'' = \texttt{split}_C(H)$
Compute $\mathbf{M}''$, $(\mathbf{M}'')^{-1}$, $\mathbf{p}''$, $\mathbf{X}''$.
2: Merge Phase (Sec. 4.2.3). Find H' s.t.:
(a) $\texttt{buckets}(H') \in \mathcal{B}$,
(b) $|\texttt{buckets}(H')| \leq n$,
(c) $\texttt{Err}_W(H')$ is minimal Compute $\mathbf{M}', (\mathbf{M}')^{-1}, \mathbf{p}', \mathbf{X}'$ for H'.

Both `split` and `merge` define bucket transformations: `split` replaces one bucket in the first base with $k+1$ buckets, while `merge` replaces $k+1$ buckets with one bucket. The transformation matrices for `merge` and `split` are shown in Fig. 5.

Consider a workload $W = (\mathbf{Q}, \mathbf{F}, \mathbf{r})$ defined in terms of the bucket base $\bar{B}$. Our goal is to represent the same workload in terms of the new bucket base $\bar{B}'$. Define:

$$\mathbf{Q}' = \mathbf{QS} \tag{18}$$

and, as a consequence:

$$\mathbf{M}' = \mathbf{S}^T\mathbf{MS} \tag{19}$$
$$\mathbf{p}' = \mathbf{S}^T\mathbf{p} \tag{20}$$

For a `merge` transformation, formula (18) gives the correct query matrix for the new basis. In the case of a `split` transformation, formula (18) is only an approximation, which holds "on average", in the following sense. Call two cells B, C *independent* if $\texttt{volume}(C \cap B)/\texttt{volume}(B) = \texttt{volume}(C)/\texttt{volume}(\mathcal{D})$. Suppose we split according to some cell C that is not in the workload, and suppose all buckets returned by the `split` are independent of all the cells in the workload: then formula (18) gives the correct query matrix. Hence, the equation holds "on average", assuming that the buckets obtained by splitting according to a random cell C are independent of the workload.

The following is a simple application of linear algebra:

PROPOSITION 4.10. *Let* $\mathbf{S}$ *be the transformation matrix for* $(\bar{B}, \bar{B}')$ *and* $\mathbf{S}'$ *the transformation matrix for* $(\bar{B}', \bar{B}'')$. *Then the transformation matrix for* $(\bar{B}, \bar{B}'')$ *is their product,* $\mathbf{S}' \cdot \mathbf{S}$.

4.2.2 The `split` Phase

The transformation matrix for `split` is given in Fig. 5. We show here how to compute $(\mathbf{M}')^{-1}$ efficiently, where $\mathbf{M}'$ is the influence matrix given by Eq.(19).

However, $\mathbf{M}'$ is always singular, because the newly introduced buckets are indistinguishable from the point of view of the workload. For example, if bucket B_j splits into B'_j and B'_{j+1}, then $\mathbf{M}'$ is obtained from $\mathbf{M}$ by copying row j and copying column j, hence it is singular.

We address this as follows. Suppose $\texttt{split}(B)$ returns $k+1$ buckets. Designated one *distinguished child* and the other k as *undistinguished children*. Create k new query statistics (B'_i, p^e_i) that estimate the sizes of the undistinguished buckets. To estimate p^e_i we use the histogram, $p^e_i = X_i\texttt{volume}(B'_i)/\texttt{volume}(B)$. Thus, we have created k new queries in order to help the system distinguish between the newly created buckets. Denote $\mathbf{Q}^e$ the $k \times n$ query matrix for these k estimate queries: thus $(\mathbf{Q}^e)^T\mathbf{Q}^e$ is a $n \times n$ matrix with exactly k 1's on the diagonal, corresponding to the k non-distinguished buckets.

Define the matrices $\mathbf{T}, \mathbf{K}$ of types $n \times (n+k)$, $(n+k) \times (n+k)$:

$$T_{jk} = \begin{cases} 1 & \text{if } B_j = B'_k, \text{ or if } B'_k \text{ is a distinguished child of } B_j \\ 0 & \text{otherwise} \end{cases}$$

$$K_{j_1j_2} = \begin{cases} -1 & \text{if } j_1 \text{ is distinguished and } j_1 = j_2 \\ 1 & \text{if } j_1 \text{ is distinguished and } j_2 \text{ is a sibling} \\ -1 & \text{if } j_1 \text{ is undistinguished and } j_1 = j_2 \\ 1 & \text{if } j_2 \text{ is distinguished and } j_1 \text{ is a sibling} \\ 0 & \text{in all other cases} \end{cases}$$

THEOREM 4.11. *Let* $\mathbf{M}' = \mathbf{S}^T\mathbf{MS} + (\mathbf{Q}^e)^T\mathbf{Q}^e$. *Then* $(\mathbf{M}')^{-1} = \mathbf{T}^T\mathbf{M}^{-1}\mathbf{T} - \mathbf{K}$.

The proof uses techniques from linear algebra and is omitted for lack of space.

Example 4.12 Consider three buckets B_1, B_2, B_3, where B_2 is split into two buckets. The new base is B'_1, B'_2, B'_3, B'_4, where $B_1 = B'_1$, $B_2 = B'_2 \cup B'_3$, $B_3 = B'_4$. We designated B'_2 the distinguished child, thus B'_3 is undistinguished. Then we add a single estimate query, for bucket B'_3 thus:

$$\mathbf{q}^e = \begin{pmatrix} 0 & 0 & 1 & 0 \end{pmatrix}$$

$$\mathbf{S} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \qquad (\mathbf{q}^e)^T\mathbf{q}^e = \begin{pmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix}$$

$$\mathbf{T} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \qquad \mathbf{K} = \begin{pmatrix} 0 & 0 & 0 & 0 \\ 0 & -1 & 1 & 0 \\ 0 & 1 & -1 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix}$$

If we have the matrix $\mathbf{M}^{-1}$ for the bucket base B_1, B_2, B_3, then the theorem gives us an expression for $(\mathbf{M}')^{-1}$ for the new bucket base B'_1, B'_2, B'_3, B'_4. □

Note that the matrices $\mathbf{T}$ and $\mathbf{K}$ are sparse: we use this to optimize matrix operations: for `split`, $(\mathbf{M}')^{-1}$ can be computed in time $O(n^2)$.

To summarize, in order to perform a `split` operation on bucket B_j we need to (1) compute $\mathbf{M}' = \mathbf{S}^T\mathbf{MS} + (\mathbf{Q}^e)^T\mathbf{Q}^e$ where $\mathbf{Q}^e$ are the k estimate queries, (2) compute $\mathbf{p}' = \mathbf{S}^T\mathbf{p} + (\mathbf{Q}^e)^T\mathbf{r}^e$, (3) compute $(\mathbf{M}')^{-1}$ using Theorem 4.11.

4.2.3 The `merge` Phase

The transformation matrix for `merge` is also shown in Fig. 5. We start by proving that the inverse $(\mathbf{M}')^{-1}$ can also be computed incrementally, in time $O(n^2)$. However, here we can push the optimization further: we can avoid computing the inverse altogether during the main loop of Algorithm 4.2 (lines 4(a), (b), and (c)), and compute $(\mathbf{M}')^{-1}$ only at the end. We give the details next.

THEOREM 4.13. *Let* $\mathbf{M}$ *be an* $n \times n$ *non-singular matrix with inverse* $\mathbf{M}^{-1}$, *let* $\mathbf{S}$ *be a* $n \times p$ *matrix, and let* $\mathbf{T}$, $\mathbf{K}$ *be two matrices with the following properties (we denote* $k = n - p$*):*

$$\mathbf{T} : p \times n \text{ is a left inverse of } \mathbf{S}: \quad \mathbf{TS} = \mathbf{I}_p$$
$$\mathbf{K} : n \times k \text{ is a kernel of } \mathbf{S}^T: \quad \mathbf{S}^T\mathbf{K} = 0$$

Denote:

$$\mathbf{G} = \mathbf{K}^T\mathbf{M}^{-1}\mathbf{K} \quad : k \times k$$
$$\mathbf{L}_1 = \mathbf{K}^T\mathbf{M}^{-1} \quad : k \times n$$
$$\mathbf{L}_2 = \mathbf{K}^T(\mathbf{M}^T)^{-1} \quad : k \times n$$

$$\mathbf{S}_{\texttt{merge}} = \begin{pmatrix} 1 & & 0 & & 0 \\ & \vdots & & \vdots & \\ 0 & \dots & f_0 & \dots & 0 \\ 0 & \dots & f_1 & \dots & 0 \\ & \vdots & & \vdots & \\ 0 & \dots & f_k & \dots & 0 \\ & \vdots & & \vdots & \\ 0 & & 0 & & 1 \end{pmatrix} \qquad \mathbf{S}_{\texttt{split}} = \begin{pmatrix} 1 & & 0 & \dots & 0 & & 0 \\ & \vdots & & & \vdots & & \\ 0 & \dots & 1 & \dots & 1 & \dots & 0 \\ & \vdots & & & \vdots & & \\ 0 & & 0 & \dots & 0 & & 1 \end{pmatrix}$$

Figure 5: Tranformation matrices for `merge` and `split`. Here $f_0, f_1, \dots$ represent the ratio of the buckets being merged and the new bucket.

(When $\mathbf{M}$ is symmetric then $\mathbf{L}_1 = \mathbf{L}_2^T$, denote it $\mathbf{L}$). Then the matrix $\mathbf{M}' = \mathbf{S}^T\mathbf{M}\mathbf{S}$ is invertible, and its inverse is:

$$(\mathbf{M}')^{-1} = \mathbf{T}(\mathbf{M}^{-1} - \mathbf{L}_2^T\mathbf{G}^{-1}\mathbf{L}_1)\mathbf{T}^T$$

We omit the proof, but only mention that it is even more involved than Theorem 4.11. The theorem allows us to compute the inverse $(\mathbf{M}')^{-1}$ for a `merge` operation, because it is easy to construct both a left inverse and a kernel, as we illustrate in the next example:

Example 4.14 Suppose we have $n = 4$ buckets B_1, B_2, B_3, B_4 and we merge them into $p = 3$ buckets $B_1, B_2 \cup B_3, B_4$. Denote:

$$f_0 = \texttt{volume}(B_2)/\texttt{volume}(B_2 + B_3)$$
$$f_1 = \texttt{volume}(B_3)/\texttt{volume}(B_2 + B_3)$$

Then:

$$\mathbf{S} = \begin{pmatrix} 1 & 0 & 0 \\ 0 & f_0 & 0 \\ 0 & f_1 & 0 \\ 0 & 0 & 1 \end{pmatrix} \quad \mathbf{T} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \quad \mathbf{K} = \begin{pmatrix} 0 \\ f_1 \\ -f_0 \\ 0 \end{pmatrix}$$

One can check that $\mathbf{T}$ is indeed a left inverse for $\mathbf{S}$ and $\mathbf{K}$ a kernel for $\mathbf{S}^T$. Denote $\mathbf{M}^{-1} = (\bar{m}_{ij})_{i,j=1,4}$. Then:

1. $\mathbf{G}$ is a single number, denoted G:
$$G = f_1^2\bar{M}_{22} - f_1f_0\bar{m}_{23} - f_1f_0\bar{m}_{32} + f_0^2\bar{m}_{33}$$
2. $\mathbf{L}_1$ is a 1×4 row matrix, obtained as a linear combination of rows 2 and 3 in $\mathbf{M}^{-1}$:
$$\mathbf{L}_1 = \begin{pmatrix} f_1\bar{m}_{21} - f_0\bar{m}_{31} & \dots & f_1\bar{m}_{24} - f_0\bar{m}_{34} \end{pmatrix}$$
3. Similarly, $\mathbf{L}_2$ is a row matrix obtained as a linear combination of the columns 2 and 3 in $\mathbf{M}^{-1}$:
$$\mathbf{L}_2 = \begin{pmatrix} f_1\bar{m}_{12} - f_0\bar{m}_{13} & \dots & f_1\bar{m}_{42} - f_0\bar{m}_{43} \end{pmatrix}$$
When $\mathbf{M}$ is an influence matrix then it is symmetric, hence $\mathbf{L}_1 = \mathbf{L}_2$, because $\bar{m}_{ij} = \bar{m}_{ji}$.
4. We compute the rank-1 matrix $\mathbf{L}_2^T\mathbf{G}^{-1}\mathbf{L}_1$ directly, then subtract it from $\mathbf{M}^{-1}$.
5. Finally, the result $\mathbf{T}(\mathbf{M}^{-1} - \mathbf{L}_2^T\mathbf{G}^{-1}\mathbf{L}_1)T^T$ is obtained by adding rows 2 and 3 and adding columns 2 and 3 in the matrix $\mathbf{M}^{-1} - \mathbf{L}_2^TG^{-1}\mathbf{L}_1$ (thus, transforming it from a 4×4 to a 3×3 matrix). □

We generalize the example to a general `merge` operation: $\mathbf{T}$ has a row containing k 1's, and $\mathbf{K}$ is:

$$\mathbf{K} = \begin{pmatrix} 0 & 0 & \dots & 0 \\ & & \dots & \\ f_0 & f_1 & \dots & f_k \\ -f_0 & 0 & \dots & 0 \\ 0 & -f_0 & \dots & 0 \\ & & \dots & \\ 0 & 0 & \dots & -f_0 \end{pmatrix}$$

COROLLARY 4.15. *The inverse $(\mathbf{M}')^{-1}$ of the influence matrix resulting after a `merge` can be computed in time $O(n^2)$.*

The proof is by generalization of Example 4.14: step 1 takes $O(k^2)$ time, steps 2 and 3 take $O(kn)$ time, and steps 4 and 5 take $O(n^2)$ time.

Finally, we show how to avoid computing $(\mathbf{M}')^{-1}$ during the main loop of the algorithm. During this loop we examine several alternative histograms $H' \in \mathcal{B}$ (obtained by merging buckets), and choose the one with the minimal cost. We show that we can compute their cost without computing $(\mathbf{M}')^{-1}$: this allows us to find the optimal histogram H' in time $O(n|\mathcal{B}|)$. Once we identified the optimal $H' \in \mathcal{B}$, we still need to compute the inverse, which do incrementally, using Theorem 4.13, but we only need to do this for the histogram with optimal cost.

THEOREM 4.16. *Let H be a histogram with n buckets for a workload $W = (\mathbf{Q}, \mathbf{F}, \mathbf{r})$ where the matrix storing the statistic is $\mathbf{X}$. We assume to have precomputed the quantities $\mathbf{M}$, $\mathbf{M}^{-1}$, $\mathbf{p}(= \mathbf{Q}^T\mathbf{F}\mathbf{r})$. Let H' be histogram with $n - k$ buckets, obtained from H through some transformation matrix $\mathbf{S}$. Let $\mathbf{T}, \mathbf{L}, \mathbf{G}$ be the matrices defined in Theorem 4.13. Denote $\mathbf{U} = \mathbf{S}\mathbf{T} - \mathbf{I}_n$ (the non-zero entries in $\mathbf{U}$ form a $(k+1) \times (k+1)$ block, and its rank is $\leq k+1$). Denote $\mathbf{v} = \mathbf{L}\mathbf{p} - \mathbf{L}\mathbf{U}^T\mathbf{p}$: $\mathbf{v}$ is a $k \times 1$ column vector. Denote:*

$$err_1 = \mathbf{p}^T\mathbf{U}\mathbf{X}$$
$$err_2 = \mathbf{p}^T\mathbf{U}\mathbf{M}^{-1}\mathbf{U}^T\mathbf{p}$$
$$err_3 = \mathbf{v}^T\mathbf{G}^{-1}\mathbf{v}$$

Then:

$$Err_W(H') = Err_W(H) + 2err_1 - err_2 + err_3$$

In particular, if H' is obtained as the result of a `merge` operation, then $\mathbf{v}$ can be computed in time $O(n)$, and the difference $Err_W(H') - Err_W(H)$ can be computed in time $O(n + k^3)$.

This allows us to find the candidate histogram H' with the minimal error: simply inspect all candidate histograms H', compute the quantity $2\texttt{err}_1 - \texttt{err}_2 + \texttt{err}_3$ for each of them, and select the one with minimal error.

Example 4.17 We illustrate the theorem on Example 4.14. First, we show $\mathbf{ST}$ and $\mathbf{U} = \mathbf{I}_4 - \mathbf{ST}$:

$$\mathbf{ST} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & f_0 & f_0 & 0 \\ 0 & f_1 & f_1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \quad \mathbf{U} = \begin{pmatrix} 0 & 0 & 0 & 0 \\ 0 & 1-f_0 & -f_0 & 0 \\ 0 & -f_1 & 1-f_1 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix}$$

For $\texttt{err}_1 = \mathbf{p}^T\mathbf{U}\mathbf{p}$ we note that:

$$\mathbf{p}^T\mathbf{U} = \begin{pmatrix} 0 & p_2(1-f_0) - p_3 f_1 & -p_2 f_0 + p_3(1-f_1) & 0 \end{pmatrix}$$

It has only $k+1 = 2$ non-zero entries, hence $\texttt{err}_1$ requires two multiplications. Similarly, $\texttt{err}_2 = \mathbf{p}^T\mathbf{U}\mathbf{M}^{-1}\mathbf{U}^T\mathbf{p}$ requires four multiplications (using the same vector $\mathbf{p}^T\mathbf{U}$). Finally, for $\texttt{err}_3$ we need to compute $v = \mathbf{L}\mathbf{p} - \mathbf{L}\mathbf{U}^T\mathbf{p}$ (which is a number) then obtain $\texttt{err}_3 = \mathbf{v}^T G^{-1}\mathbf{v}$ (where, as shown earlier, G is also a number).□

5. EXPERIMENTS

We now describe the results of our preliminary empirical evaluation. The goals of our study are (1) to compare the EM and L2-distance based approaches for query size estimation, (2) since none of the previously proposed self-tuning histograms addresses distinct values, we primarily compare the accuracy of our approaches to the query optimizer, and (3) to measure the execution efficiency of building our self-tuning histograms.

5.1 Setup

All our experiments are conducted using the TPCH benchmark [12]. The data size is 1 GB. The benchmark data itself is uniformly generated which is the case where the query optimizer is known to have high accuracy. Thus in order to evaluate our techniques, we generate TPCH data that is skewed [8]. This data has a zipfian distribution where the zipfian parameter is set to $z = 1$.

One of the applications of self-tuning histograms is in computing statistics over views [4]. Creating these statistics is known to be challenging and prior approaches [5] focus on heuristically computing samples over joins. Sampling is known to result in poor accuracy for distinct value estimation. This problem is only exacerbated by the difficulty of sampling over joins [7]. Thus, building statistics over views by using execution feedback is attractive. Of course, there are significant challenges to be addressed in using execution feedback for building statistics over views. However, the first question that arises is whether such an approach is even feasible and whether we obtain the improvement in accuracy we seek. Our study addresses this question of feasibility.

As such we conduct our empirical study in the setting of building a statistic over views. We focus on the view obtained by joining LINEITEM with ORDERS. Our column of interest is O_TOTALPRICE. We generate a workload of queries following a uniform distribution, involving both `count` and `count distinct` queries with different selections on O_TOTALPRICE generated uniformly. The cardinality and distinct values computed using execution feedback are fed into the self-tuning histogram. (In our experiments, we compute the exact cardinalities and distinct values by running the query against SQL Server.)

5.2 EM Vs L2

As noted in Section 3, the execution efficiency of computing the EM model given cardinality and distinct value counts is very poor. In fact, even for histograms with at most 20 buckets, we found that computing the histogram took longer than 7 hours. Note that in the case of the EM approach, this means that we cannot process a large number of query feedback records since it proceeds by first constructing a refined histogram and then merges buckets.

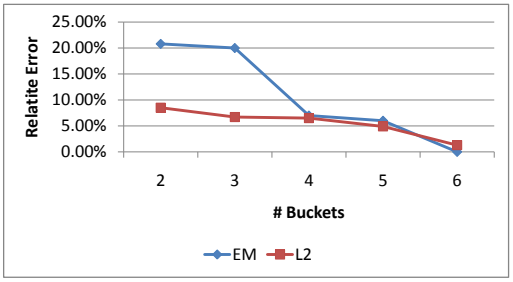


Figure 6: Comparing EM With L2.

However the question arises whether for a small number of query feedback records and a correspondingly small number of histogram buckets, how the EM approach compares with the L_2-based approach. We investigate this question here. We fix a workload of 10 queries which are used to construct the self-tuning histograms. We then test the accuracy of the histogram using a different workload (again of 10 queries) which are generated using the same distribution. We plot the average relative error of the estimation as a function of the number of buckets allocated. Figure 6 shows the results. We can see that (1) the accuracy of both EM and L_2 improves sharply as we increase the number of buckets allocated, and (2) the accuracy of L_2 is comparable to that of EM. On the other hand, the L_2-based approach has significant computational advantage over EM.

For the rest of this section, we only consider the L_2-distance based histogram.

5.3 Comparison With Query Optimizer

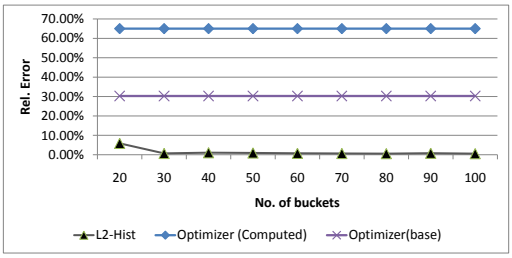
As noted above, since none of the previously proposed self-tuning histograms addresses distinct values, we primarily compare the accuracy of our approaches to the query optimizer.

For the query optimizer, we use two estimates. The first that we call `Optimizer_Computed` is one where we only have relevant statistics on the base tables and the optimizer uses its error propagation algorithms to estimate the result size.

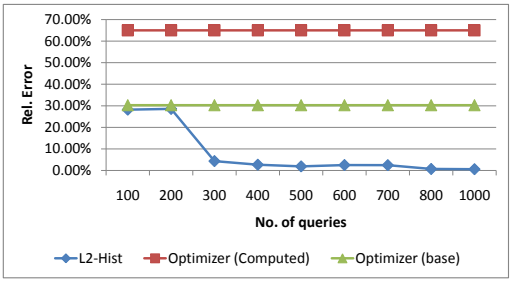
We also consider the algorithm proposed in [5] to create statistics over views. There, the approach is to approximate a sample over the view and then use the sample to compute the histogram. In order to compare against this method, we materialize the view as a base table and use sampling to compute statistics over the O_TOTALPRICE column since this is what the algorithm in [5] is trying to efficiently approximate. We call this approach `Optimizer_base`.

The self-tuning histogram is constructed using a workload of queries generated as described above (involving both `count` and `count distinct` queries). In order to measure the accuracy, we use a workload of 100 *test* queries that uses the same distribution as the "training" workload above

Figure 7 shows the results of our comparison. In Figure 7(a), we vary the number of buckets allocated to our histogram keeping the number of "training" queries fixed at 1000. In Figure 7(b), we fix the number of buckets at 100 and vary the number of "training queries". The number of buckets and queries are plotted on the X-axis. The Y-axis reports the relative error measured as the ratio between the absolute error and the true value. We use L_2-`Hist` to refer to the L_2-distance based histogram.



(a) Varying number of buckets



(b) Varying number of queries

Figure 7: Comparison With Query Optimizer

Since the data is skewed and since the optimizer uses independence and containment assumptions that are well-known to be unreliable, we find that the `Optimizer_Computed` error is significantly higher than that returned by L_2-`Hist`. Interestingly, even the `Optimizer_Base` error is higher than that of L_2-`Hist`. This happens since our test workload also involves distinct value estimation where sampling is well-known to be error-prone. Indeed, if we restrict ourselves to cardinality queries alone, the error returned by `Optimizer_base` drops to a point where it is lower than L_2-`Hist`.

In addition, as expected, the error of L_2-`Hist` reduces as more space is provided for the histogram and as the number of training queries increases.

5.4 Execution Time

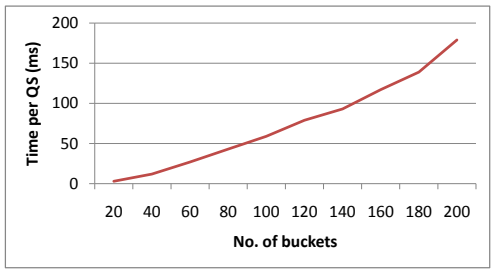


Figure 8: Execution time.

Finally, we plot the execution time of L_2-`Hist` as a function of the number of buckets allotted. The query workload size is fixed at 1000 and is generated in the same manner as in the previous subsection. The X-axis plots the number of buckets and the Y-axis, the execution time per query statistic in milli-seconds. Owing to the quadratic nature of our algorithm, we find that the running time grow super-linearly with space. In contrast, previously proposed approaches such as `STGrid` and `STHoles` take an order of magnitude smaller time and are not plotted on the graph since their time is close to 0. However, we still find our execution time — 60ms per query feedback for a histogram with 100 buckets — acceptable given the benefits of our approach, primarily the fact that it permits the maintenance of multiple statistics.

6. RELATED WORK

Various synopsis structures have been proposed for cardinality estimation in a relational query optimizer such as histograms [19], wavelets [13], samples [17] and sketches [2]. This paper focuses on histograms. This is the main data structure used by commercial database systems for cardinality estimation.

Histograms are typically constructed offline to be then used by the query optimizer at optimization. There is a vast body of prior work on constructing both one-dimensional and multi-dimensional histograms from data [19] that can be disk-resident [21, 16] or streaming [28, 32, 14, 15].

Despite this large body of prior work, significant estimation errors are commonplace in commercial query optimizers. More recently, there have been several proposals for leveraging query execution feedback in order to address this problem. This body of work focuses on two aspects of the problem — one is to incorporate mechanisms in the query execution engine to monitor and record cardinalities at a low overhead [31, 9], and the other is to use this recorded feedback to improve the estimation framework of the optimizer. The broad area of self-tuning synopsis structures [10, 1, 6, 26, 30, 23, 25] addresses the latter problem.

The most relevant prior work to this paper is the work on self-tuning histograms [1, 6, 26, 30]. STGrid [1] and SASH [26] try to explicitly approximate the data distribution by exploiting execution feedback. STHoles [6] requires extremely detailed feedback from the query execution — it needs the intersection of the query with each bucket of the histogram. This requires a modification to the query engine over and above what has been considered in [31, 9] where only the overall result size is monitored. This modification is likely to incur significant runtime overhead. The state-of-the-art self-tuning histogram is ISOMER [30] which has been shown to be empirically more accurate than STGrid. It uses the entropy maximization principle to first construct a histogram that is consistent with the execution feedback while making the fewest additional assumptions. None of the above histograms addresses distinct value counts which is the focus of this paper.

The principle of minimizing the L_2 distance is commonly used in traditional histograms. For example V-Optimal histograms use the same principle to minimize the L_2 distance [20, 29, 21, 16, 28, 23]. The principle has been applied to not only point queries [20, 29, 21] but also to range queries [16, 24], and more recently extended to build histograms over probabilistic databases [11]. Here specific classes of queries such as the class of all point queries or the class of all point and all range queries, or more restricted classes such as hierarchical range queries are considered. The primary differences in our scenario are: (1) the query workload is an

explicit input and can consist of arbitrary point and range queries, and (2) they can contain feedback about not only cardinalities but also distinct values.

7. CONCLUSIONS

Self-tuning histograms are computed and maintained from query execution feedback, obtained almost for free during normal database operation. In this paper, we considered the problem of building self-tuning histograms that are cognizant of distinct value counts. We studied two methods of consistency. One is to use the Entropy Maximization (EM) principle. We showed how this principle could be applied to take distinct value counts into account. We saw that this requires a different probability space than that proposed in prior work for cardinalities alone.

Since the equations generated by the EM principle are prohibitively hard to solve efficiently, we proposed an L_2-distance based approach to capture consistency in the presence of both cardinalities and distinct values. We showed how these L_2-distance based histograms can be computed from the query feedback records in a single pass, at a cost of $O(n^2)$ per query plus $O(n)$ per candidate histogram inspected, which translates into about 60ms for a histogram with 100 buckets. Our empirical study indicated that the L_2-distance based histogram produces comparable accuracy to EM while at the same time being computationally much more efficient.

8. REFERENCES

[1] A. Aboulnaga and S. Chaudhuri. Self-tuning Histograms: Building Histograms Without Looking at Data. In *SIGMOD*, 1999.

[2] N. Alon, P. B. Gibbons, Y. Matias, and M. Szegedy. Tracking Join and Self-Join Sizes in Limited Storage. In *PODS*, 1999.

[3] N. Alon, Y. Matias, and M. Szegedy. The space complexity of approximating the frequency moments. In *STOC*, 1996.

[4] N. Bruno and S .Chaudhuri. Exploiting statistics on query expressions for optimization. In *SIGMOD*, pages 263–274, 2001.

[5] N. Bruno and S. Chaudhuri. Efficient Creation of Statistics over Query Expressions. In *ICDE*, 2003.

[6] N. Bruno, S .Chaudhuri, and L Gravano. STHoles: A multidimensional workload-aware histogram. In *SIGMOD*, pages 211–222, 2001.

[7] S. Chaudhuri, R. Motwani, and V. R. Narasayya. On random sampling over joins. In *SIGMOD*, 1999.

[8] S. Chaudhuri and V. Narasayya. Program for TPC-D Data Generation with Skew. ftp://ftp.research.microsoft.com/users/viveknar/tpcdskew.

[9] S. Chaudhuri, V. R. Narasayya, and R. Ramamurthy. Diagnosing Estimation Errors in Page Counts Using Execution Feedback. In *ICDE*, 2008.

[10] C. Chen and N. Roussopoulos. Adaptive Selectivity Estimation Using Query Feedback. In *SIGMOD*, 1994.

[11] G. Cormode and M. Garofalakis. Histograms and wavelets on probabilistic data. In *ICDE*, 2009.

[12] Transaction Processing Performance Council. Tpc-h (ad-hoc, decision support) benchmark. http://www.tpc.org/.

[13] A. Deligiannakis, M. N. Garofalakis, and N. Roussopoulos. Extended wavelets for multiple measures. *ACM Trans. Database Syst.*, 32(2), 2007.

[14] S. Guha, P. Indyk, S. Muthukrishnan, and M. Strauss. Histogramming data streams with fast per-item processing. In *ICALP*, 2002.

[15] S. Guha, N. Koudas, and K. Shim. Approximation and streaming algorithms for histogram construction problems. *ACM Trans. Database Syst.*, 31(1), 2006.

[16] S. Guha, N. Koudas, and D. Srivastava. Fast algorithms for hierarchical range histogram construction. In *PODS*, pages 180–187, June 2002.

[17] P. J. Haas, J. F. Naughton, S. Seshadri, and A. N. Swami. Selectivity and cost estimation for joins based on random sampling. *J. Comput. Syst. Sci.*, 52(3), 1996.

[18] Homotopy Continuation Method to Solve a System of Nonlinear Algebraic Equations. http://www.math.uic.edu/ jan/PHCpack/.

[19] Y. E. Ioannidis. The History of Histograms. In *VLDB*, 2003.

[20] Y. E. Ioannidis and V. Poosala. Balancing Histogram Optimality and Practicality for Query Result Size Estimation. In *SIGMOD*, 1995.

[21] H. V. Jagadish, N. Koudas, S. Muthukrishnan, V. Poosala, K. C. Sevcik, and T. Suel. Optimal Histograms with Quality Guarantees. In *VLDB*, 1998.

[22] E.T. Jaynes. *Probability Theory: The Logic of Science*. Cambridge University Press, Cambridge, UK, 2003.

[23] A. C. König and G. Weikum. Combining Histograms and Parametric Curve Fitting for Feedback-Driven Query Result-size Estimation. In *VLDB*, 1999.

[24] N. Koudas, S. Muthukrishnan, and D. Srivastava. Optimal histograms for hierarchical range queries. In *PODS*, 2000.

[25] P. Larson, W. Lehner, J. Zhou, and P. Zabback. Cardinality estimation using sample views with quality assurance. In *SIGMOD*, 2007.

[26] L. Lim, M. Wang, and J. S. Vitter. SASH: a self-adaptive histogram set for dynamically changing workloads. In *VLDB*, 2003.

[27] V. Markl, N. Megiddo, M. Kutsch, T. M. Tran, P. J. Haas, and U. Srivastava. Consistently estimating the selectivity of conjuncts of predicates. In *VLDB*, 2005.

[28] S. Muthukrishnan, M. Strauss, and X. Zheng. Workload-Optimal Histograms on Streams. In *ESA*, 2005.

[29] V. Poosala, Y. E. Ioannidis, P. J. Haas, and E. J. Shekita. Improved histograms for selectivity estimation of range predicates. In *SIGMOD*, 1996.

[30] U. Srivastava, P. Haas, V. Markl, M. Kutsch, and T. M. Tran. ISOMER: Consistent histogram construction using query feedback. In *ICDE*, page 39, 2006.

[31] M. Stillger, G. M. Lohman, V. Markl, and M. Kandil. LEO - DB2's LEarning Optimizer. In *VLDB*, 2001.

[32] N. Thaper, S. Guha, P. Indyk, and N. Koudas. Dynamic multidimensional histograms. In *SIGMOD*, pages 428–439. 2002.