Entropy-based Histograms for Selectivity Estimation

Hien To [*]
University of Southern California
Los Angeles, CA 90089
hto@usc.edu

Kuorong Chiang
Teradata Cooporation
601 Nash Street
El Segundo, CA 90245
Kuorong.Chiang@Teradata.com

Cyrus Shahabi [*]
University of Southern California
Los Angeles, CA 90089
shahabi@usc.edu

ABSTRACT

Histograms have been extensively used for selectivity estimation by academics and have successfully been adopted by database industry. However, the estimation error is usually large for skewed distributions and biased attributes, which are typical in real-world data. Therefore, we propose effective models to quantitatively measure bias and selectivity based on information entropy. These models together with the principles of maximum entropy are then used to develop a class of entropy-based histograms. Moreover, since entropy can be computed incrementally, we present the incremental variations of our algorithms that reduce the complexities of the histogram construction from quadratic to linear. We conducted an extensive set of experiments with both synthetic and real-world datasets to compare the accuracy and efficiency of our proposed techniques with many other histogram-based techniques, showing the superiority of the entropy-based approaches for both equality and range queries.

Categories and Subject Descriptors

H.2.4 [**Database Management**]: Systems—*Query processing*; F.2.m [**Analysis of Algorithms and Complexity**]: Miscellaneous

Keywords

Selectivity Estimation; Histograms; Entropy Models

1. INTRODUCTION

Selectivity estimation is the task of estimating the size of the result set of a relational algebra operator. For a particular query, multiple execution plans can be generated with different ordering of operators. Thus, selectivity estimation of intermediate temporary relations significantly influences the choice of a query plan chosen by a query optimizer. Accurate estimations are crucial to generate optimal execution plans while bad estimations often lead to large overhead in performance.

Several principle techniques have been proposed for selectivity estimation, including non-parametric techniques (e.g., histograms [15, 16, 10, 23, 5], kernel density estimation [1] and sampling [12]) and parametric techniques [3]. If the attribute values are not uniformly distributed, which is usually the case with real data, the histogram approach is commonly used. Moreover, histogram is the most popular technique used in commercial DBMS. However, estimating the query result size is challenging due to data dependence. Under the assumptions of uniformity and independence of attribute values, all the methods, even naïve approaches (e.g., [15]) work equally well. Nevertheless, the naïve approaches often result in large estimation errors on non-uniform datasets [9, 10]. Therefore, better histograms are needed to perform well on various datasets under different distributions and biases that exist in real-world datasets.

In this paper, we propose a class of novel entropy-based algorithms to build histograms utilizing the traditional method of best-cut algorithm. First, we observe that both histogram and entropy maximization try to maximize information content. The goal of any histogram is to capture as much information from the data as possible, which is essentially the aim of the entropy maximization. Thus, our first algorithm, called *Maximum Entropy (ME)*, minimizes reduction in the total entropy of histograms. Next, we introduce an entropy-based analysis to model *selectivity* and *bias* of equality query. Using the knowledge of bias and selectivity, we propose two other algorithms: *Minimum Selectivity Error (MSE)* and *Minimum Bias (MB)*. We also show that using bias as an additional statistics for each histogram bucket, we can significantly improve the selectivity estimations of equality queries. Moreover, by employing incremental and pruning techniques, we can further reduce the construction complexities of the entropy-based histograms from $O(N^2)$ to $O(NlogB)$, where N is the number of distinct values and B is the number of histogram buckets.

Most histograms partition data distribution into buckets based only on the differences in frequencies. For example, the V-optimal histogram that minimizes the cumulative weighted variance of the buckets has been proven to be the best in grouping similar frequency values into the same buckets [7]. Therefore, it is the best histogram in terms

[*]This author's research has been funded in part by NSF grant IIS-1115153. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of any of the National Science Foundation.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from Permissions@acm.org.
CIKM'13, Oct. 27–Nov. 1, 2013, San Francisco, CA, USA.
Copyright 2013 ACM 978-1-4503-2263-8/13/10 ...$15.00.
http://dx.doi.org/10.1145/2505515.2505756.

of minimizing the error of the equality query. However, through experiments, we discovered that the best histograms for equality queries (e.g., MSE) do not necessarily perform as well for range queries. The reason is that while equality estimation error depends only on frequency distribution within each bucket, range estimation is concerned with the spread of attribute values across consecutive buckets [16]. In [16], an approach is introduced to approximate both value and frequency sets using *area*, which is defined as the product of the frequency and the spread. This work shows that Maxdiff and V-optimal histograms using the area (instead of frequency) result in the lowest estimation error for range queries. Hence, we also incorporate area into our ME approach to capture as much information as possible for range queries.

We conducted an extensive set of experiments to compare our proposed approaches with most competitors for both range and equality queries under various data distributions using both synthetic and real-world datasets. The experiments show that our proposed incremental algorithms require only a small fraction of the construction time of the V-optimal histogram and achieve comparable estimation errors. Furthermore, the experimental results suggest that there is no single histogram that performs best for all query types and data distributions. However, the class of entropy-based techniques can offer a histogram that works best in terms of effectively balancing construction time and accuracy. In particular, ME performs best for range queries, while MSE is the best choice for equality queries. Moreover, data distribution is another determining factor. For example, MB performs best for equality queries under smooth data distributions. We have identified other determining conditions in the experiments and suggested their best corresponding techniques. In sum, the best strategy is to select the most appropriate entropy-based histogram for a given query type and data distribution.

Thus, the main contributions of this work are:

- Providing an entropy-based analysis to model bias and selectivity,
- Developing a class of incremental entropy-based histograms for selectivity estimation that are effective in accuracy and efficient in construction time,
- Making the observation that a good histogram for equality queries does not necessarily performs well for range queries and vice versa,
- Introducing bias factor and weighted mean as per-bucket statistics to improve estimation of equality query.

The remainder of the paper is organized as follows. Section 2 defines a typical histogram and how to use it for estimating selectivity of equality and range queries. Section 3 describes existing histogram techniques with which we compare our entropy-based histograms. Section 4 provides the details of our entropy-based approaches. Section 5 reports on our experimental evaluation. Sections 6 and 7 conclude and discuss the future directions of this study.

2. BACKGROUND

2.1 Histogram Definition

The data distribution of an attribute X is the set of pairs $D = \{(v_1, f_1), (v_2, f_2), ..., (v_{|D|}, f_{|D|})\}$, where v_i and f_i ($1 \leq i \leq |D|$) are possible values of attribute X and their corresponding frequencies, respectively, and $v_i < v_j$ when $i < j$. A histogram on attribute X is constructed by partitioning the data distribution D into k ($1 \leq k \leq |D|$) mutually disjoint subsets called buckets: $H(X) = \{b_1, b_2, ..., b_k\}$, each bucket b_i represents a sub-range r_i of X's domain. The larger k is, the better the histogram captures the data distribution. Each bucket b_i usually stores: 1) $s^{(i)}$ and $e^{(i)}$ are the start value and the end value of b_i, respectively (i.e., $e^{(i)}$ and $s^{(i+1)}$ are consecutive values), 2) t_i is the number of tuples t in the dataset for which $t.X \in$ sub-range r_i, and 3) dv_i is the number of distinct values of b_i. Figure 1 shows how to construct a histogram from a data distribution $D = \{(1, 5), (2, 5), (5, 7), (6, 2), (7, 3), (8, 4)\}$.

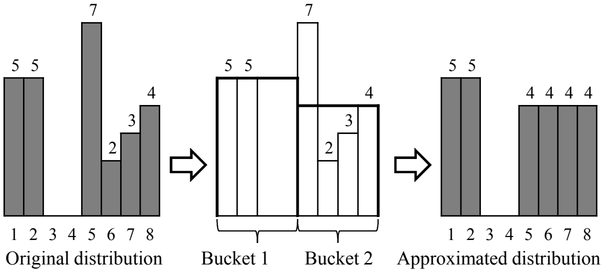


Figure 1: Equal-width histogram construction

To improve the accuracy of selectivity estimation, each bucket may store more statistics, such as min value, max value, mode value, and highest/lowest frequency values. For example, the problem of non-uniform distribution within buckets can be tackled by using additional statistics of a 32-bit information (i.e., a 4-level tree (4LT) index) for storing approximate cumulative frequencies at 7 intervals of a bucket [2]. This study shows that applying 4LT to the existing histograms such as Maxdiff and V-optimal improves the accuracy of range query significantly. We later suggest two other additional statistics namely *bias factor* and *weighted mean* that can be used to enhance equality estimation.

2.2 Selectivity Estimation using Histogram

In this section, we present two methods to estimate selectivities (i.e., result set sizes) of range and equality queries using histograms. In order to estimate the result set size of each query, we define *selectivity factor s* as the expected fraction of tuples that will satisfy the corresponding predicates. Thus, within a bucket b_i, the selectivity of the query is estimated as $t_i s_i$.

Equality Query: With assumptions of uniform frequency and continuous value, selectivity factor of an equality query $X = val$ ($val \in r_i$) is defined as:

$$s_i = \frac{1}{dv_i} \tag{1}$$

In Figure 1, the result set size of an equality query $X = 5$ is estimated as 4 while the actual value is 7. It is important to note that the accuracy of the equality query depends only on the uniformity of the data distribution within each bucket. For example, the first bucket is more uniform than the second bucket, it therefore results in a smaller estimation error in the histogram.

Range Query: The result set size of a range query is estimated based on how much the range covers the histogram buckets. With an assumption of uniform distribution between the minimum and maximum attribute values, selectivity factor of a range query $val_1 \leq X \leq val_2$ ($val_1, val_2 \in r_i$ and $val_1 \leq val_2$) is defined as $s_i = \frac{val_2 - val_1}{e^{(i)} - s^{(i)}}$. For example, on the histogram in Figure 1, a range query $1 \leq X \leq 5$ fully covers the first bucket and partially covers the second. Thus, the histogram only needs to estimate selectivity of the second bucket and has actual selectivity for the first. Since the query covers one forth the size of the second bucket ($s_i = 0.25$), the selectivity is estimated as 4 in the second bucket. Hence, the estimated result set size is 14 while the actual value is 17. In addition, the worst case example would be the result set size of another range query $3 \leq X \leq 4$ is approximated as 5 while the actual value is 0. From this example, some properties of the range query can be derived: 1) since the buckets that are fully covered by a range query do not contribute to the estimation error, the larger the range, the lower the estimation error of the query (i.e., the first example); 2) the accuracy of the range query depends not only on the uniformity of the data distribution but also the spread of attribute values (i.e., the second example); 3) equality query is a special case of the range query when its start and end points are at the same position.

3. RELATED WORK

This section presents current histogram techniques that we use to compare with our algorithms in the experiments. We group the histograms based on how their bucket boundaries are specified as follows.

Classical Histograms: Equal-width (EW) and equal-height (EH) histograms [15] are equal partitioning methods of value and frequency, respectively. While the number of attribute values associated with each bucket is the same in EW; in EH the total number of tuples having the attribute values associated with each bucket is the same. This study shows a significant improvement of EH over EW on the worst-case and average error of range query.

Variants of Biased Histograms: Through preserving separated uni-buckets that contain only one value for high bias values, the estimation accuracy is significantly improved. In [7], the class of serial histograms and its subclass of biased histograms are proposed that store the L most biased values in L uni-buckets and one multivalued bucket for the rest of the values. The paper concludes that the optimal histogram for an equality-join query is always serial. In another work [16], an efficient histogram for skew datasets, named compressed histogram is presented, in which the N highest frequencies are placed in N uni-buckets while the rest use an EH histogram.

Variants of V-optimal Histograms: The V-optimal histogram on an attribute minimizes cumulative weighted variance among all histograms using the same number of buckets [7]. However, the optimal solution to V-optimal histogram is expensive to compute. Hence, there are other efforts in either balancing histogram optimality and practicality with approximate yet faster algorithms [8, 9], or keeping the optimal solution, but with a higher memory requirement [10]. This study proposed to use dynamic programming to reduce the complexity to $O(N^2B)$, where N is the number of distinct values ($N = |D|$) and B is the number of buckets (we call this algorithm VODP). Another algorithm proposed in [16] further reduces the complexity to $O(NB)$ using an Iterative Improvement approach, termed VOII. The algorithm starts from a random state, which is a set of values representing the bucket boundary placements. Consequently, the algorithm moves each boundary until it reaches the local optimum partition that minimizes the total weighted variance of the buckets, then the algorithm places the next boundary.

Local Search Approaches: There is a set of histograms using the greedy approach to reduce the computation complexity by making local optimal decisions, thus achieving linear time complexity for partitioning. The Maxdiff (MD) histogram [16] inserts bucket boundaries between adjacent frequency values if the difference between these values is one of the $B-1$ highest differences. Another work [10] presents a one-dimensional variant of the multidimensional MHIST algorithm proposed in [17]. Both MHIST and MD iteratively choose and split the bucket that is most in need of partitioning. While MD chooses the boundary that maximizes differences between two consecutive frequency values, MHIST chooses the bucket with the highest sum square of error. A recent work proposed GDY [5], a fast and effective histogram construction based on greedy local search. GDY generates good sample boundaries, which then are used to construct B final partitions optimally using VODP. This study compares GDY variants with DnS [23] and AHistL [4] in minimizing the total error of all the buckets and shows its superiority in resolving the efficiency-quality trade-off.

There are other studies on using histograms for query optimization, such as approaches using query feedback (e.g, [22]) and multidimensional histograms (e.g, [17]).

4. ENTROPY-BASED HISTOGRAMS

In information theory, entropy is a measure of the uncertainty in a random variable, which has been extensively studied in Mathematics [20], Physics [19], Ecology [6], etc. Specifically, in Statistical Thermodynamics, entropy is a measure of the number of possible micro-states of a system [19]. In Ecology, entropy is used to measure how diverse an ecosystem is. In this work, we use Shannon entropy as measure of information content.

4.1 Preliminaries

Entropy in Selectivity Estimation: While entropy has long been used in various field of science, only a few work have studied entropy for selectivity estimation [14, 13, 18]. In [14], a maximum entropy (ME) approach is proposed to improve estimation accuracy of conjunctive predicates. This work presents an iterative scaling algorithm to estimate unknown selectivities from known selectivities by choosing the most uniform/independent selectivity model that is consistent with all of the available knowledge. The algorithm exploits all available information and avoids the bias problem. In a later work [13], the ME approach is shown to improve DB2 UDB Query Optimizer's selectivity estimation by orders of magnitudes. Recently in [18], Ré et al. model the selectivity estimation problem using probability space, then use this model to estimate selectivity of conjunctive queries.

Entropy in Histograms: Although entropy-based approaches have been used successfully in approximating selectivities of conjunctive predicates and in disciplines outside database community (e.g., image processing [11, 24]), to the best of our knowledge, this is the first study on developing entropy-based histograms for selectivity estimation. Shannon [20] introduces entropy as a measure of the uncertainty in a random variable with a probability distribution $X = (p_1, p_2, ...)$:

$$H(X) = -\sum_i p_i \, . \, log_2(p_i) \tag{2}$$

where $\sum_i p_i = 1$ and log_2 is the logarithm of base 2 (hereafter referred to as log).

Our idea of using entropy in histogram construction is inspired by the way information theory can be used in new areas of data management [21]. This tutorial presents how to compute entropy of a table column $H(X) = -\sum_{i=1}^{N} p_i(x) \, . \, log\big(p_i(x)\big)$, where X is the column, N is the number of distinct values of X and $P_i(x)$ is the value of the probability of the i^{th} value in the column X. Since X can be represented as a data distribution, we use column and distribution interchangeably. The column X can also be used as a random variable: x denotes a value taken by X and $p(x)$ is the probability that X is equal to x, $p(X = x)$.

Our observation is that histogram and entropy maximization have the same goal of maximizing information. In particular, keeping similar frequency values in the same bucket is essentially maximizing the entropy (ME) of the column of frequencies since ME tries to reach standard uniformity. Consequently, we define the entropy of a histogram bucket as the entropy of the probability distribution of the values within that bucket, in which the probability of each value is defined as its frequency divided by the total number of attribute values. For all buckets with the same number of attribute values, a bucket entropy is maximum when the frequencies are equal.

Entropy Update: The complexity of Equation (2) is $O(E)$, where E is the size of the distribution list. In fact, the entropy can be computed incrementally in $O(1)$ time when an element is added to or removed from the distribution list. This improvement is later used to design the incremental versions of the entropy-based histograms. It can be shown that the following formula is correct:

$$H(X) = p_1 H(X_1) + p_2 H(X_2) + H(p_1, p_2) \tag{3}$$

where X_1 and X_2 are non-overlapping partitions of X, and p_1 and p_2 are probabilities associated with X_1 and X_2, respectively. Then, inserting a new row R into column X increases its entropy to:

$$H(X') = H\Big(\frac{c}{t+c}, \frac{t}{t+c}\Big) + \frac{t}{t+c} H(X) \tag{4}$$

where t is the cardinality of X before insertion and c is the cardinality of R. Equation (4) can be derived from Equation (3) if we consider X' includes two non-overlapping partitions R and X with associated probabilities $\frac{c}{t+c}$ and $\frac{t}{t+c}$. Similarly, removing a value R from column X reduces its entropy as follows:

$$H(X') = \bigg(H(X) - H\Big(\frac{c}{t}, \frac{t-c}{t}\Big)\bigg) \frac{t}{t-c} \tag{5}$$

where t is the cardinality of X before deletion and c is the cardinality of R.

Besides reducing histogram construction time, another advantage of these formulas is that entropy can be added into each bucket of a histogram as additional statistics with little overhead $O(1)$ when collecting histogram statistics (Section 4.5).

4.2 A Histogram for Range Query

Naïve Best-cut Approach: In this section, we present an entropy-based histogram that is good for range queries. As mentioned in Section 4.1, histogram and the principle of maximizing entropy have the same goal of retaining as much information as possible. Particularly, using entropy maximization to construct histograms has the following three advantages: 1) group similar frequencies in the same bucket 2) maximize information content, which is the ultimate goal of any histogram—capturing as much information from the data as possible and 3) achieve standard uniformity and independence assumptions [14], which are ideal for selectivity estimation. Hence, we define entropy of a histogram as $\sum_{i=1}^{B} W(b_i)H(b_i)$, where B is predefined as the number of disjoint buckets b_i, and $W(b_i)$ and $H(b_i)$ are the weight (i.e., will be defined later) and the entropy of the bucket b_i, respectively. With this definition, the B-buckets histogram itself limits the quantity of information available in the original data. Therefore, our idea is to select the histogram boundaries that minimize reduction in total entropy. Consequently, we develop the so-called ME histogram using the idea of the best-cut approach.

Most histograms focus on grouping similar frequency values into the same bucket and ignore the closeness of attribute values, which is important for range query. In Figure 1, given a perfectly uniform distribution in the first bucket, in which equality queries (e.g., $X = 2$) can be accurately estimated, range queries (e.g., $3 \leq X \leq 4$) is, however, poorly estimated. In [16], the spread of attribute values was explicitly considered when partitioning. In order to better approximate the entire data distribution, partitioning methods should consider proximity of both the value and frequency sets. This study proposes to partition based on *area* $a_i = f_i s_i$, where f_i and s_i are the frequency and the spread of v_i, respectively. Intuitively, the *area* approach seems to match with the ME histogram since both try to capture as much information as possible. Hence, to provide the best picture of the data distribution, we extend ME to the *area* approach, which would result in a better histogram for range queries.

The pseudo-code of ME is depicted in Algorithm 1. The algorithm starts with a one-bucket histogram containing all attribute values. Subsequently, it iteratively chooses the best cuts with minimum entropy reduction, namely *global cuts* until there are enough buckets (i.e., a B-buckets histogram needs $B - 1$ cuts), (Lines (4-29)). Every global cut splits a bucket into two new disjoint buckets, the so-called left-hand-size (LHS) and right-hand-size (RHS) buckets. The global cut is represented by its position, which is computed from the local best cut in Line (27). At each iteration, the local best cut minimizes entropy reduction within each bucket, which is defined as the original bucket entropy minus the sum of the entropies of the two new buckets (Line (18)). Similarly, a *local cut* is characterized by its position, which is initialized in Line (10) and updated in Lines (19-21). In addition, if there are two buckets with the same reduction in entropy, the highly populated buckets should be split first. The algorithm hence considers *weights* of the buckets in choosing the global cut in Line (23). The *weight* of a bucket is calculated as the sum of the areas in the bucket, computed in Line (9).

Algorithm 1 Maximum Entropy

```
 1: Input: $A$ {area list}, $k$ {the number of cuts}
 2: Output: $splits$ {ordered list of boundary positions}
 3: Compute the first cut, update $splits$, init $minHeap$
 4: while $|splits| < k$ do
 5:     Delete $previousBuckets \leftarrow minHeap$
 6:     for $b$ in $previousBuckets$ do
 7:         Get area list of bucket $b$: $A_b$
 8:         if $|A_b| > 1$ then
 9:             Get weight of bucket $b$: $w_b \leftarrow sum(A_b)$
10:             Init $locCutPos \leftarrow 0, locMinH^- \leftarrow MAX$
11:             Init $T_L \leftarrow 0, T_R \leftarrow sum(A_b)$
12:             Init $H_L \leftarrow 0, H_R = H_O \leftarrow H(A_b)$
13:             for $j = 1 \rightarrow |A_b| - 1$ do
14:                 Get $j^{th}$ frequency: $x \leftarrow A_b[j]$
15:                 Update $T_L \leftarrow T_L + x, T_R \leftarrow T_R - x$
16:                 Update $H_L \leftarrow H\left(\frac{x}{x+T_L}, \frac{T_L}{x+T_L}\right) + \frac{T_L}{T_L+x} H_L$
17:                 Update $H_R \leftarrow \left(H_R - H(\frac{x}{T_R}, \frac{T_R-x}{T_R})\right) \cdot \frac{T_R}{T_R-x}$
18:                 Set $H^- \leftarrow H_O - (H_L + H_R)$
19:                 if $H^- < locMinH^-$ then
20:                     Set $locCutPos \leftarrow j, locMinH^- \leftarrow H^-$
21:                 end if
22:             end for
23:             Insert $minHeap \leftarrow \{newBuckets, w_b \cdot locMinH^-\}$
24:         end if
25:     end for
26:     Get $\{bucket, locCutPos\} \leftarrow minHeap$
27:     Set $gloCutPos = start^{(bucket)} + locCutPos$
28:     Insert $splits \leftarrow gloCutPos$
29: end while
```

Incremental and Pruning Techniques: With the naïve best-cut approach, in order to find the global cut point of all buckets, ME needs to compute all local cuts and compares these cuts with the current global cut, which results in the $O(N^2)$ complexity of the ME algorithm. However, we can further reduce the complexity of ME using local and global optimization methods.

In the local optimization phrase, to find local cuts, we can use Equations (4) and (5) to compute the entropy reduction incrementally. That is, instead of recomputing the LHS and RHS bucket entropies at every local iteration, these values can be computed one time at the beginning and then updated at each iteration. Particularly, the algorithm initially sets the position of the cut to zero in Line (10), which means that the entire column is on the right hand side (RHS) and no value is on the left-hand-side (LHS). Line (11) initializes the total frequencies of LHS and RHS, and Line (12) initializes the entropies of LHS and RHS together with the entropy of the original bucket. The loop in Lines (13-22) finds the local cut that minimizes the entropy reduction of the original bucket. It iteratively moves the cut one position to the right, which means one value is moved from RHS to LHS. Consequently, the total frequencies of LHS and RHS are computed in Line (15). Given these values, the LHS and the RHS entropies are updated in Lines (16) and (17), respectively. Finally, the corresponding reduced entropy is computed in Line (18), from which the position and the minimum entropy reduction of the local cut are updated in Lines (19-21).

In the global optimization phase, ME uses a min heap to store the minimum entropy reductions of all buckets (i.e, the local best cuts). The key observation is that most of the local best cuts are already known from the last iteration except the ones of the two new buckets resulted from the previous global best cut. Line (5) returns these buckets and the corresponding minimum entropy reduction from the min heap, which is initialized in Line (3). Line (23) inserts the minimum entropy reduction of a new bucket into the heap. As a result, by exploiting the incremental and pruning techniques, the complexity of ME is reduced by an order of magnitude (Section 4.4).

4.3 Histograms for Equality Query

In the following, we propose effective models to measure selectivity and bias based on information entropy. Based on that, we present two histograms that are good for equality queries. While the first histogram algorithm maximizes reduction in total selectivity error, the other maximizes total bias reduction. Although these methods have different objective functions, they both use the best-cut approach to solve the problems.

4.3.1 Entropy-based Models

Selectivity Factor: When attribute values follow highly skewed distributions (e.g., Zipf distributions, typically found in word frequency data for many textual databases), the error of selectivity estimation is usually large. Motivated by this problem, we develop a set of effective formulas to quantitatively measure *selectivity* as well as *bias*. Considering the knowledge of bias, effective histograms for highly biased data can be constructed.

Given the total distinct values N, the selectivity factor of an equality query on a column is defined by Equation (1). This formula is based on the assumption of uniformity of attribute values. If this assumption does not hold, the formula may not result in a good estimation. For example, if the distribution of a column is (0.6, 0.4), the weighted sum of squared error (WSSE) for using Equation (1) is $0.6(0.5 - 0.6)^2 + 0.4(0.5 - 0.4)^2 = 0.01$. It means that the selectivity estimation error is on average 0.1. Similarly, the average equality estimation error of another distribution (0.8, 0.2) is 0.3. Hence, the more bias in the data, the higher the estimation error.

To intuitively understand how entropy can be used for selectivity estimation, we illustrate our idea with some examples. When all attribute values have the same selectivity factor $s = 1/N$, using Equation (2), the total entropy is derived as: $H = -logs = logN$. It can be proved that for all distributions with the same number of distinct values, the entropy reaches the maximum value for uniform distribution. On the other hand, the more bias in a distribution, the smaller the entropy, for example $H(0.5, 0.5) = 1$, $H(0.6, 0.4) = 0.97$ and $H(0.8, 0.2) = 0.72$. Thus, entropy seems to be a good metric for capturing bias. Particularly, we propose the following formula as a replacement for Equation (1).

$$s = 2^{-H} \tag{6}$$

Two values with the same frequency have the selectivity factor $s = 2^{-1} = 0.5$. The selectivity factor increases as the bias in the data grows. For instance, for (0.6, 0.4) distribution $s = 2^{-0.97} = 0.51$, and for (0.8, 0.2) distribution $s = 2^{-0.72} = 0.61$. Using Equation (6), the weighted sum of squared errors of two distributions (0.6, 0.4) and (0.8, 0.2) are 0.0097 and 0.0625; thus, the average selectivity errors are 0.098 and 0.25, respectively. When compared to using Equation (1), the entropy approach reduces the average estimation errors by 0.002 and 0.05 for the two examples of distributions, respectively. Consequently, the more biased the data, the more the entropy model can improve upon the estimation error.

Bias Factor: In the following, we quantitatively measure bias. Our important observation is that the estimation error of equality query is likely to be underestimated if the data distribution is biased. The reason is that the high frequency values tend to get selected more often than the low frequency values. Moreover, even though estimation error is negative regardless of overestimation or underestimation, the latter can lead to worse optimizer decisions downstream. For example, the optimizer may choose an aggressive plan when selectivity is severely underestimated. Therefore, we define *bias factor (BF)* to compensate for this loss:

$$s' = s(1 + BF) \tag{7}$$

where s is the non-biased selectivity factor and s' is the selectivity factor with bias. Intuitively, s' is equal to s when the data is uniformly distributed. Substituting Equation (6) into Equation (7), we obtain:

$$BF = 2^{\Delta H} - 1 \tag{8}$$

where ΔH is the entropy deviation from the uniform distribution (i.e., $\Delta H = H - H' = logN - H$, where N is the number of distinct values). Since BF is greater than zero (i.e., $\Delta H \geq 0$), the expected selectivity factor for a biased distribution is always greater than that of uniform distribution.

4.3.2 Minimum Selectivity Error (MSE)

Using Equation (6), we develop a new algorithm called MSE to construct histograms for highly biased data. The algorithm minimizes the total mean squared error of selectivity factor, or briefly the total selectivity error, defined as $\sum_{i=1}^{B} W(b_i)E(b_i)$, where $W(b_i)$ and $E(b_i)$ are the weight and the expected squared error of selectivity factor of the bucket b_i, respectively. $E(b_i)$ is further derived as:

$$E(b_i) = \sum_{j=1}^{dv_i} p_j(p_j - s_i)^2 \tag{9}$$

where dv_i is the number of distinct values within b_i, p_j is the probability of a value in the bucket, and s_i is the selectivity factor for the bucket using Equation (6). For each value p_j, the equality estimation returns s_i, the squared difference is thus $(p_j - s_i)^2$. Moreover, since high frequency values are likely to be queried more often, they contribute more error to the bucket. Hence, we multiply the squared error by the probability that this value appears. Each value therefore contributes $p_j(p_j - s_i)^2$ to the total error of all the buckets. Thereafter, we expand the expected squared error in Equation (9) as follows: $E(b_i) = \sum_{j=1}^{dv_i} \left(p_j^3 - 2p_j^2 s + p_j s_i^2\right) = A - 2Bs_i + Cs_i^2$, where A, B, C are $\sum_{j=1}^{dv_i} p_j^3$, $\sum_{j=1}^{dv_i} p_j^2$ and $\sum_{j=1}^{dv_i} p_j$, respectively.

Consequently, a similar optimization framework as in ME, but with a different objective function can be applied for MSE. The algorithm recursively partitions a column into two disjoint columns until having enough cuts. Every cut splits a bucket into two new disjoint buckets, the so-called left-hand-size (LHS) and right-hand-size (RHS) buckets. Using Equations (4) and (5), A, B, C can be updated on the fly for both LHS and RHS. In each iteration, MSE chooses the cut that maximizes the reduction in the expected squared error of selectivity factor: $E_o - (W_l E_l + W_r E_r)$, that is, the original selectivity error minus the weighted sum of the selectivity errors of the two new buckets. Similar to ME, MSE also employs the local and the global optimization techniques to reduce the construction cost.

In summary, the more biased the data, the more the entropy approach (i.e., 2^{-H}) can improve the naïve model (Equation 1) for equality estimation.

4.3.3 Minimum Bias (MB)

Exploiting bias information, we develop another histogram algorithm that minimizes total weighted bias defined as: $\sum_{i=1}^{B} W(b_i)BF(b_i)$, where $W(b_i)$ and $BF(b_i)$ are the weight and the bias factor (Equation 8) of the bucket b_i, respectively. The *weights* are computed as the number of tuples in the bucket. The algorithm utilizes the best-cut approach that maximizes the reduction of the total bias, called MB. Every cut splits a bucket into two new disjoint buckets, from which the bias reduction can be computed by subtracting the weighted sum of two new bias factors from the original: $BF_o - (W_r BF_l + W_r BF_r)$. Using similar optimization framework as ME and MSE, but with a different objective function, MB can be solved in linear time.

4.4 Algorithm complexity

This section shows the linear complexity of our entropy-based histograms with the number of distinct values N. The cost of the algorithms at each iteration includes 1) local cost (i.e., searching local best cuts) and 2) global cost (i.e., maintaining global best cuts using a min heap). We first show that the global cost is deterministic. Since the algorithm deletes a global cut and insert two new others at each iteration, the global cost at iteration k is $3logk$, where $logk$ is the cost of insert/delete an element to/from the min heap of k elements. Hence, the global cost is $3\sum_{k}^{B-1} logk$. This series can be approximated by $\int_{1}^{B-1} logxdx = O(BlogB)$. We now consider the local costs in the worst-case, the best-case and the average-case scenarios. The worst case happens when the global cuts are chosen to be the first/last positions of the buckets. In the case of the last positions, the total local cost is: $T(N) = T(1) + T(N-1) + N - 1 = ... = (N-1)+(N-2)+...+(N-B) = NB - B(B+1)/2 = O(NB)$. Thus, the maximum local cost is $O(NB)$. The minimum local cost (i.e., the best case) is $N + BlogB$ where all the global cuts are $B - 1$ continuous values and the algorithm only searches within that range (i.e., N is the cost of seeking the first global cut). Finally, in the average case we have the following recurrence $T(B) = (N-1) + \frac{1}{N}\sum_{k=0}^{B-1}(T(k) + T(B-1-k)) = (N-1) + \frac{2}{N}\sum_{k=0}^{B-1} T(k)$. Solving the recurrence gives $T(B) = O(2NlnB) = O(NlogB)$. The detailed proof is similar to that of the average complexity of Quick Sort. In sum, the average complexity of the algorithm is $O(NlogB)$ while the worst case is $O(NB)$.

4.5 Equality Query Enhancements

In this section, we present a technique that employs additional statistics within each bucket to reduce the estimation error of equality query. As shown in Section 4.3.1, if

Method	Time	Ref
VODP	$O(N^2B)$	[10]
DnS	$O(N^{4/3}B^{5/3})$	[23]
AHistL	$O\big(n + B^3(logn + \epsilon^{-2})logn\big)$	[4]
MHIST	$O(NB)$	[10]
GDY_BDP	$O(NB)$	[5]
VOII	$O(NB)$	[16]
ME, **MSE**, **MB**	$O(NlogB)$	this
MD	$O(NlogB)$	[16]
EH	$O(N)$	[15]
EW	$O(B)$	[15]

Table 1: Construction complexity

the data distribution is biased, the estimation tends to underestimate the selectivity of equality query. One way to compensate for this loss is to use the *bias factor*, which is stored per bucket to enhance equality estimation. Another method is to use a *weighted mean* as selectivity, computed as $\big(\sum_i w_i v_i\big)/\big(\sum_i w_i\big)$, where w_i is the frequency of v_i divided by the number of attribute values within the bucket. It is straightforward to show that using the weighted mean as selectivity minimizes the weighted sum of squared error (WSSE) of a bucket. For example, the WSSE of a bucket with distribution $\{(8, 0.8), (2, 0.2)\}$ is $0.8(x-8)^2+0.2(x-2)^2$, where x is is the estimated value. This formula is minimized when its derivative is equal to zero, resulting $x = 6.8$. In fact, 6.8 is the weighted mean of the values. Although using the weighted mean minimizes the estimation error of the equality query, it is more costly in terms of storage requirement when compared with using bias factor. That is, to maintain weighted mean in an online manner, one would need to maintain the numerator and denominator separately. In contrast, the bias factor is computed from the bucket entropy, which is already given during histogram construction. Moreover, the bias factor can be recomputed incrementally in constant time when updating the bucket.

5. PERFORMANCE EVALUATION

This section reports on our extensive comparison of our approaches with various histogram algorithms for estimating the size of result sets of equality and range queries. Table 1 summarizes the complexities of these algorithms. Note that we choose GDY_BDP with 10 iterations—the best performing algorithm in [5], DnS with $\chi = (n/B)^{2/3}$, which is suggested in [23] and AHistL [4] with $\epsilon = 0.01$.

5.1 Experimental Setup

Data Distributions: We conducted experiments on two real-world time series data given in Table 2. We also use various synthetic datasets with different frequency and value sets. The frequency sets includes Zipf and randomly permuted Zipf (spike) distributions. The skew parameter z of the Zipf distributions was assigned to 1 in all the experiments. Moreover, we used the value sets that have been proposed in [16]. To have the spread of attribute values as important as the frequency sets, they need to have the same domain and range (i.e., 0 to 100K). These synthetic datasets are summarized in Table 3. A pair of value and frequency sets represents a dataset by concatenating their names. For example, *uniform_zipf* is a dataset with value set following uniform distribution and corresponding frequency set following Zipf distribution. For each dataset, T is the number of tuples and S is the sample skewness, measuring the asymmetry of the probability distribution. The higher the absolute of this value, the more skewed the dataset. With the synthetic datasets, the number of attribute values N is 1000.

Name	Size	Attr values (N)	Distribution
Phone	13664	225	Figure 2a
Balloon	4002	421	Figure 2b

Table 2: Real-world time series datasets

Value Sets	Frequency Sets			
	zipf		zipf_ran	
uniform	S=2	T=800K	S=0.2	T=800K
zipf_inc	S=38	T=700K	S=16	T=500K
zipf_dec	S=-1.5	T=700K	S=-23	T=400K
cusp_min	S=2.8	T=800K	S=-0.1	T=600K
cusp_max	S=-1.6	T=800K	S=-5	T=700K
zipf_ran	S=2.9	T=700K	S=0.6	T=400K

Table 3: Synthetic datasets

Queries: Since root mean squared error (RMSE) is a good measure of accuracy, it was used for error evaluation. For a set of queries Q, RMSE is computed as in follows:

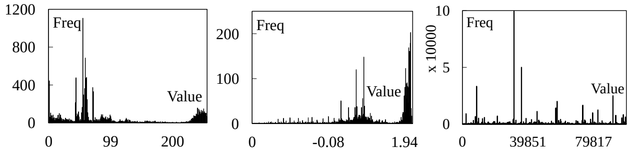
$$RMSE = \sqrt{\frac{1}{|Q|}\sum_{q \in Q}\big(S_a(q) - S_e(q)\big)^2}$$

where $S_a(q)$ and $S_e(q)$ are the actual and estimated size of the query result, respectively. Using this equation, we measured the average error of 10,000 random equality queries $X = val$, where val is in the value domain. Equation (1) was used to compute the selectivity of the queries by default.

Next, we used the *area* approach for all algorithms to improve range query estimation. Their RMSEs are measured over 10,000 random range queries whose midpoints are in the value domain and their lengths are bounded by an *offset*. When *offset* of a query is one, the size of the range query is bounded by the spread of one bucket of an equal-width histogram. To understand how well the algorithms perform through varying the size of query, we relatively categorized the range query into three types, small, medium and large queries based on the *offset* value. The *offset* of the small query is in the range of (0,.1]. Similarly, the *offsets* of the medium and large queries are between (.1,1] and (1,10], respectively. We observed that the distinction between the approaches is not clear when the query size is small (i.e., the lines intersect each other); thus, we plotted the errors of the large range queries for better visualization (i.e., more gaps between the lines).

5.2 Construction Cost

We fixed the number of buckets B at 30 and varied the number of attribute values N from 1000 to 20,000. Table 4 presents the construction time of different techniques on

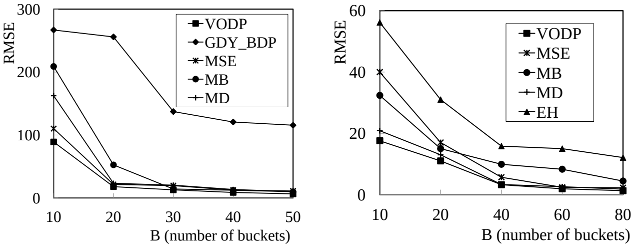


(a) Phone (b) Balloon (c) Spike

Figure 2: Some datasets

Algorithm	**Input size (N)**		
	1000	**10,000**	**20,000**
VODP	235	2325	5423
MHIST	0.547	49	211
VOII	0.047	2.376	8.885
MSE	0.024	0.033	0.045
MB	0.02	0.025	0.036
ME	0.017	0.02	0.032
MD	0.015	0.015	0.025
EH	0	0.001	0.001
EW	0	0.001	0.001

Table 4: Construction time in seconds (uniform_zipf dataset)



(a) Phone dataset

(b) Balloon dataset

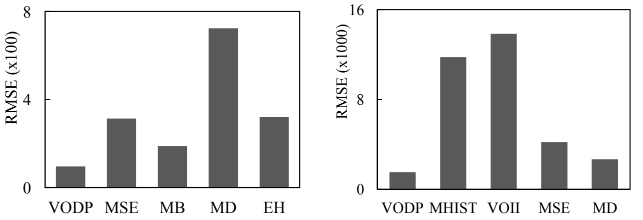
Figure 3: Quality comparision for equality query on real data (top five techniques)

the same data. The construction time of our algorithms, including ME, MSE and MB are lower than many other algorithms. We did not present GDY_BDP, DnS and AHistL in this table because these algorithms were implemented in a different language. However, Table 1 suggests that the construction time of these algorithms would be larger than that of VOII. Furthermore, EW and EH are the fastest algorithms while VODP is the slowest one. Nevertheless, in Section 5.3 we will show that VODP provides the best quality results while the classical histograms perform poorly. Thus, the challenge in building an efficient histogram is to resolve this quality-scalability tradeoff.

5.3 Estimation Error

In this section, we compare the accuracy of different techniques for equality and range queries. An important observation is that the accuracy of the range query is similar to the accuracy of an equality query when its size is close to zero. Therefore, the existing techniques that work well for equality queries are likely to perform well for small range queries. However, these algorithms do not necessarily perform well for large range queries.

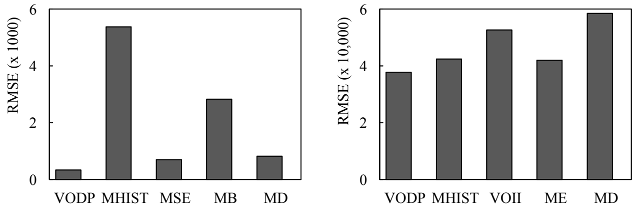
Equality Query: In the first set of experiments, we evaluated the quality of the algorithms by varying the number of buckets B. Figure 3 presents the result of our experiments on two real-world datasets. As the figures show, the estimation error reduces as we increase the number of buckets. Specifically, Figure 3a shows the error results with the phone dataset. The figure shows that MSE, MD and MB are close to the optimal line produced by VODP, in which MSE performs best. The other far-optimal algorithms (e.g., VOII, EH and EW) are not shown in this figure for better visualization. Moreover, figure 3b presents the error results with the balloon dataset. Still, MSE, MD and MB are close to the optimal line.



(a) uniform_zipf dataset

(b) uniform_zipf_ran dataset

Figure 4: Quality comparision for equality query on synthetic data (top five techniques)



(a) Equality query

(b) Range query

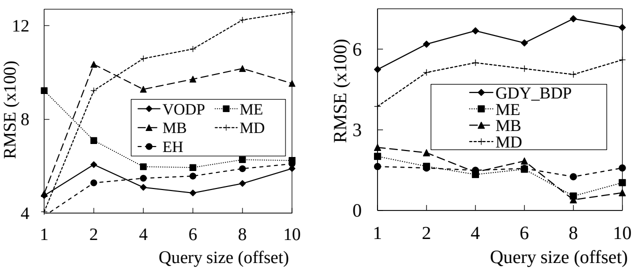
Figure 5: Average estimation errors over 12 synthetic datasets (top five techniques)

Without loosing generality, we fixed the number of buckets B at 30 to test the performance of the algorithms under various extreme Zipf distributions. Figure 4 depicts the result of our experiments on 2 synthetic datasets. In the first dataset, our algorithm MSE and MB are close to the optimal solution (VODP). Moreover, EH and MB perform very well for the smooth dataset (*uniform_zipf*); however, they are worse for the spike dataset (*uniform_zipf_ran*). Another observation is that the average error of the spike data is much higher than that of the smooth data (by 20 orders magnitude) due to the spikiness of randomly permuting the frequencies.

Consequently, to find the best overall histogram for equality query, we summarized the average errors of equality query over 12 synthetic datasets in Table 3. Figure 5a suggests that MSE is consistently close to the optimal solution. In fact, the average error of MSE is 21% smaller than that of MD while their complexities and run times are almost the same (Table 1, 4). Since computing the optimal solution of VODP is expensive, MSE has been shown to be the best technique for equality query that effectively balances construction time and quality.

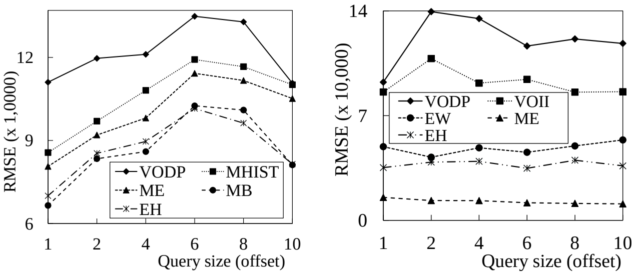
Range Query: In the next set of experiments, we evaluated the performance of the algorithms for range query estimation by varying the size of the range. Figures 6a and 6b illustrate estimation errors of the histograms on two corresponding real-world datasets. In these datasets, ME and EH have been shown to be equally effective, and they outperform most of the other techniques. Moreover, we examined the performance of the algorithms on various synthetic datasets in Table 3. Figures 7a and 7b present the errors of range estimation on the smooth (i.e., *uniform_zipf*) and the spike (i.e., *uniform_zipf_ran*) datasets, respectively. These figures show that the ME histogram consistently has low average error for range queries. Still, MB performs very well for the smooth dataset; however, it is worse for the spike dataset.

Consequently, to find the best overall algorithm for range query, we summarized the performance of the algorithms by



(a) Phone dataset (b) Balloon dataset

Figure 6: Quality comparision for range query on real data (top five techniques)



(a) uniform_zipf dataset (b) uniform_zipf_ran dataset

Figure 7: Quality comparision for range query on synthetic data (top five techniques)

finding the average errors of all sizes of range query over all of our synthetic datasets. For each size of query (small, medium and large), we choose one sample *offset*. In particular, the values of *offsets* for small, medium and large query are 0.05, 0.5 and 5, respectively. Figure 5b presents the overall range query performance of the histograms on 12 synthetic datasets. The average errors of the three best techniques VODP, MHIST and ME are almost the same (i.e., the average errors of ME and MHIST are only 8% larger than that of VODP); however, their complexities are much different. Particularly, the complexity of ME is lower than those of the competitors by an order of magnitude (Table 1). In sum, ME has been shown to be the histogram of choice for range query estimation.

Range Query vs Equality Query: Generally, the algorithms VODP, MSE, MD and MB are good at reducing error of equality query (Figures 3a-5a). However, these methods are not necessarily good for range queries (Figures 6a-7b,5b). Some of them even tend to perform worse for range queries, for example MSE and MB are not in the top 5 techniques in Figure 5b. On the other hand, ME and VOII are not good for equality queries; nevertheless, they do very well for range queries. The reason is that equality query and range query are different in characteristics. While equality query estimation has only one objective that being making the frequency within each bucket as uniform as possible, range query estimation does care about the spread of the values.

To summarize, although V-optimal gives the smallest estimation error, its complexity is too high for practical implementation. Thus, our entropy-based histograms MSE and ME perform best in terms of balancing between accuracy and construction time.

Per-Bucket Statistics: In this section, we present how much improvement per-bucket statistics, including *weighted mean* and *bias factor* can bring to the histograms on a synthetic dataset and a real dataset. We only present the results of the two datasets; however, the following observations hold for all the other datasets. The improvements are computed by relative reduction in average estimation error: $100(E_{wo} - E_w)/E_{wo}$, where the E_{wo} and E_w are computed without/with using additional statistics, respectively. Table 5 shows that weighted mean improves the accuracy of equality query 27% in average and up to two digits for most of the histograms. Besides, bias factor enhances the accuracy of equality query 18% in average. Although using weighted mean maximally reduces the average error of equality query, its maintenance cost is higher than that of bias factor (Section 4.5). Meanwhile, bias factor effectively balance the update time against the accuracy improvement, which seems to be a good choice for additional statistics. Finally, it is important to note that weighted mean assures to reduce the estimation error; however, bias factor does not guarantee this property. That is, the bias factor in Equation (7) might overcompensate the information loss in Equation (1) though this is not usually the case.

Hist	**uniform_zipf**		**Phone dataset**	
	Weighted mean	Bias factor	Weighted mean	Bias factor
VODP	20	9	15	9
DnS	37	22	29	18
AHistL	40	28	32	20
MHIST	14	5	19	9
GDY_BDP	42	36	17	10
VOII	46	34	12	11
ME	46	38	15	5
MSE	27	10	17	0
MB	4	2	17	7
MD	23	7	24	10
EH	91	88	18	10
EW	17	10	23	12

Table 5: Improvements when using per-bucket statistics

6. DISCUSSION

The key contribution of this work is the design of the incremental algorithms, which achieved comparable quality of results to that of V-optimal while being superior in time efficiency. Among the algorithms that addressed the efficiency-quality trade-off, our entropy-based histograms performed the best. Moreover, this work shows that entropy-based approaches, which have been successfully applied to selectivity estimation, can be used for creating histograms in an efficient manner. Besides, there was an assumption in our bias and selectivity models that high frequency values are queried more often than low frequency values. However, there might be the case where one may very well be interested in the values which occur infrequently, rather than common values. One way to address this issue would be to consider the importance (e.g., actual access frequencies) of the values, rather than the frequency alone. That is, the critical values require more accuracy than others, and thus these values are likely to be stored in small buckets. The extreme case of this is known as compressed histogram [16], where important values (i.e., either high frequency or high access values) are stored in separated uni-buckets. However, the variants of biased histograms are beyond the focus of this work.

7. CONCLUSIONS

In this paper, we focused on one-dimensional histograms and their applications for selectivity estimation. We observed that information entropy can be used to model bias and selectivity and thus a perfect tool to improve the accuracy of histograms for real-world datasets. Hence, a class of entropy-based histograms was proposed that achieves near-optimal quality in linear runtime. Moreover, extending ME to the *area* approach renders a histogram that can handle range queries effectively. Our experiments showed that no one histogram is the best for all data distributions and query types, that is, equality and range queries require different types of histograms. However, the entropy-based techniques always offer the best overall performance for both equality and range queries.

This work opens up new opportunities to answer some open questions including: How to measure information of a histogram tailored for a particular usage (i.e., selectivity estimation of predicates)? How to find the histogram that contains the maximum information? How to leverage the incremental histogram construction approach to improve the efficiency of update algorithms? How to extend entropy-based histograms to cases in which more data is available, such as actual query log and query feedback? How this work can be efficiently applied to multidimensional histograms? We would like to investigate some of these issues as part of our future work.

8. REFERENCES

[1] Z. Botev, J. Grotowski, and D. Kroese. Kernel density estimation via diffusion. *The Annals of Statistics*, 38(5):2916–2957, 2010.

[2] F. Buccafurri, D. Rosaci, L. Pontieri, and D. Saccà. Improving range query estimation on histograms. In *Data Engineering, 2002. Proceedings. 18th International Conference on*, pages 628–638. IEEE, 2002.

[3] C. Chen and N. Roussopoulos. *Adaptive selectivity estimation using query feedback*, volume 23. ACM, 1994.

[4] S. Guha, N. Koudas, and K. Shim. Approximation and streaming algorithms for histogram construction problems. *ACM Transactions on Database Systems (TODS)*, 31(1):396–438, 2006.

[5] F. Halim, P. Karras, and R. H. Yap. Fast and effective histogram construction. In *Proceedings of the 18th ACM conference on Information and knowledge management*, pages 1167–1176. ACM, 2009.

[6] M. O. Hill. Diversity and evenness: a unifying notation and its consequences. *Ecology*, 54(2):427–432, 1973.

[7] Y. Ioannidis and S. Christodoulakis. Optimal histograms for limiting worst-case error propagation in the size of join results. *ACM Transactions on Database Systems (TODS)*, 18(4):709–748, 1993.

[8] Y. Ioannidis and Y. Kang. Randomized algorithms for optimizing large join queries. In *ACM SIGMOD Record*, volume 19, pages 312–321. ACM, 1990.

[9] Y. Ioannidis and V. Poosala. Balancing histogram optimality and practicality for query result size estimation. *ACM SIGMOD Record*, 24(2):233–244, 1995.

[10] H. Jagadish, N. Koudas, S. Muthukrishnan, V. Poosala, K. Sevcik, and T. Suel. Optimal histograms with quality guarantees. In *Proceedings of the International Conference on Very Large Data Bases*, pages 275–286. INSTITUTE OF ELECTRICAL & ELECTRONICS ENGINEERS, 1998.

[11] J. Kapur, P. K. Sahoo, and A. Wong. A new method for gray-level picture thresholding using the entropy of the histogram. *Computer vision, graphics, and image processing*, 29(3):273–285, 1985.

[12] R. Lipton, J. Naughton, and D. Schneider. *Practical selectivity estimation through adaptive sampling*, volume 19. ACM, 1990.

[13] V. Markl, P. Haas, M. Kutsch, N. Megiddo, U. Srivastava, and T. Tran. Consistent selectivity estimation via maximum entropy. *The VLDB journal*, 16(1):55–76, 2007.

[14] V. Markl, N. Megiddo, M. Kutsch, T. Tran, P. Haas, and U. Srivastava. Consistently estimating the selectivity of conjuncts of predicates. In *Proceedings of the 31st international conference on Very large data bases*, pages 373–384. VLDB Endowment, 2005.

[15] G. Piatetsky-Shapiro and C. Connell. Accurate estimation of the number of tuples satisfying a condition. In *ACM SIGMOD Record*, volume 14, pages 256–276. ACM, 1984.

[16] V. Poosala, P. Haas, Y. Ioannidis, and E. Shekita. Improved histograms for selectivity estimation of range predicates. *ACM SIGMOD Record*, 25(2):294–305, 1996.

[17] V. Poosala and Y. Ioannidis. Selectivity estimation without the attribute value independence assumption. In *Proceedings of the International Conference on Very Large Data Bases*, pages 486–495. INSTITUTE OF ELECTRICAL & ELECTRONICS ENGINEERS (IEEE), 1997.

[18] C. Ré and D. Suciu. Understanding cardinality estimation using entropy maximization. *ACM Transactions on Database Systems (TODS)*, 37(1):6, 2012.

[19] D. V. Schroeder. *An introduction to thermal physics.* Pearson Education India, 2007.

[20] C. E. Shannon and W. Weaver. A mathematical theory of communication, 1948.

[21] D. Srivastava and S. Venkatasubramanian. Information theory for data management. In *Proceedings of the 2010 international conference on Management of data*, pages 1255–1256. ACM, 2010.

[22] U. Srivastava, P. Haas, V. Markl, M. Kutsch, and T. Tran. Isomer: Consistent histogram construction using query feedback. In *Data Engineering, 2006. ICDE'06. Proceedings of the 22nd International Conference on*, pages 39–39. IEEE, 2006.

[23] E. Terzi and P. Tsaparas. Efficient algorithms for sequence segmentation. In *SIAM SDM*. Citeseer, 2006.

[24] C. Wang and Z. Ye. Brightness preserving histogram equalization with maximum entropy: a variational perspective. *Consumer Electronics, IEEE Transactions on*, 51(4):1326–1334, 2005.