Available online at www.sciencedirect.com
ScienceDirect
Data & Knowledge Engineering 62 (2007) 156–176
DATA & KNOWLEDGE ENGINEERING
www.elsevier.com/locate/datak

Error minimization in approximate range aggregates

Xuemin Lin *, Qing Zhang, Yidong Yuan, Qing Liu

NICTA and University of New South Wales, School of Computer Science and Engineering, Sydney, NSW 2052, Australia

Received 21 November 2004; accepted 27 July 2006
Available online 28 August 2006

Abstract

Histogram techniques have been used in many commercial database management systems to estimate a query result size. Recently, it has been shown that they are very effective to support approximation of query processing especially aggregates. In this paper, we investigate the problem of minimizing average errors of approximate aggregates using histogram techniques. Firstly, we present a novel linear-spline histogram model that is more accurate than the existing models. Secondly, we propose a novel histogram construction technique for minimizing such average errors, which is shown to generate a near optimal histogram. Our experiment results demonstrate that the new histogram construction techniques lead to a great accuracy improvement on the existing techniques.

Keywords: Approximate aggregate; Data reduction; Histogram

1. Introduction

Traditional developments in query processing optimization techniques have focused on generating exact answers in a way that seeks to minimize response time and maximize throughput. However, in many recent database applications it may be too expensive to produce exact answers. For example, getting the exact answer for a complex query issued to a data warehousing may take hours or even days. Sometimes a network or disk storage failure may cause a part of data not accessible; this makes exact answers impossible. Further, in a decision support system an early response by approximate answers is quite helpful because the user can quickly determine a direction to drill down the data. Clearly, approximation is a good alternative in those applications. The quality of an approximate processing is measured by two conflicting parameters: efficiency and accuracy. A ''good'' approximate query processing usually means a good trade-off between efficiency and accuracy.

* Corresponding author.
E-mail addresses: lxue@cse.unsw.edu.au (X. Lin), qzhang@cse.unsw.edu.au (Q. Zhang), yyidong@cse.unsw.edu.au (Y. Yuan), qingl@cse.unsw.edu.au (Q. Liu).

0169-023X/$ - see front matter
doi:10.1016/j.datak.2006.07.009

Notation

T	a data distribution
B	a disjoint partition
β	the bucket number of B
B_k	one bucket of B ($1 \leqslant k \leqslant \beta$)
v_{ik+m}, f_{ik+m}	the $(m+1)$th value and frequency pair in bucket B_k ($m \geqslant 0$)
$\overline{f_k}$	the average frequency in B_k
q_k, c_k	the parameters of the linear function used in the LSLS model
q'_k, c'_k	the parameters of the linear function used in the LSCG model
q''_k, c''_k	the parameters of the linear function used in the LSCSG model
$A_T(x,y)$	the exact answer to a range aggregate over $[x,y]$ of T
$A'_{T,B}(x,y)$	the approximate answer to a range aggregate over $[x,y]$ of T, against a data partition B
$e_{T,B}(x,y)$	the absolute error between $A_T(x,y)$ and $A'_{T,B}(x,y)$
$P(x,y)$	a probability density function of range $[x,y]$
$E_{T,B}$	the average error for a range aggregation over T, against a data partition B
$M^*(X,Y)$	the optimal result for using at most Y buckets to partition the first X values of T
$M[a,b]$	the bucket containing the consecutive $(v_a,f_a),\ldots,(v_b,f_b)$ in T

Database queries have two forms – *aggregate* and *non-aggregate*. An aggregate returns a numeric value; for instance, COUNT, SUM, AVG, etc. A non-aggregate query returns a set of tuples from a database tables; for example, JOIN.

Aggregates are not only an important class of queries in conventional database system applications but also serve as an important base for *quantitative* data mining problems [6,14] in modern applications, such as financial market analysis and telecommunications. Approximate processing of aggregates, thus, has recently attracted a great deal of attention. Most research results are based on a *data size reduction* paradigm. Four techniques [7] have been developed, such as *sampling*, *wavelet*, *sketches*, and *histogram*. Sampling [1,28,8] is a popular technique of data size reduction, which takes a small portion of data as representative. To reduce approximation errors caused by applying sampling techniques to data with a big distribution skew, wavelet [2] techniques were firstly adopted by the authors in [23] to approximate query processing. The basic idea is to compress data by *important* wavelet *coefficients* [10]. The sketch techniques [11,5] have been mainly developed to tackle the query processing problems, such as quantile computation or window join size estimation, under the data stream environment. The last technique is based on histograms, which were originally used in commercial database management systems to capture the distribution statistics of attribute values in query optimizers.

Among these four techniques, histogram is the most popular data reduction technique for approximately processing range aggregates; this is partially because the histogram technique naturally suits for estimating aggregates. The basic idea of a histogram technique is to partition original data into certain number of "intervals" ("buckets"). The key issues in histogram techniques are: (1) how to partition the original data into buckets, and (2) how to approximate the original data in each bucket. The goal is to approximately process aggregates as accurate as possible. Many histogram techniques have been recently developed [9,16,15,17,19,21,22,26]. In this paper, we will focus on developing effective histogram techniques.

The current research in the area may be classified into two categories: (1) effective ways to represent each bucket [4,7,9,21] and (2) effective ways [4,7,9,12,19,20] to partition data into buckets. To represent each bucket more accurately, a "linear-spline" technique has been proposed [21] combined with the least-square [29] method in contrast to the "conventional" histogram techniques [16,15,17,19,22,26]. In [4], an "encoding" scheme has been developed as a post-process to further partition data in each bucket following a data partitioning.

Most existing techniques in partitioning data have been focused on minimizing approximation errors based on various different intuitive optimal models [15–17,19,22,26]. The paper [9] presents the first work on a for-

mal investigation of the problem of error minimization in range aggregates with respect to a "uniform" query pattern. With the applications to time series data, [9] takes the assumption that all the "values" are evenly distributed over a data set while the "frequencies" have an arbitrary distribution. The techniques in [9] are specifically developed for such an environment; and they are not quite applicable to the applications where values are not evenly distributed. In this paper, we will investigate the problem of processing approximate range aggregates on an arbitrary value space.

1.1. Our contributions

A potential problem of the linear-spline technique in [21] is that although it greatly outperforms the conventional histogram techniques (described in the next section) when the value space is evenly spanned by the values, it may not necessarily take this advantage with an arbitrarily value distribution. To resolve this, in this paper we present a new linear-spline technique to represent each data bucket in a histogram, which is shown more accurate than the existing techniques to represent data buckets in a histogram. This is the first contribution of the paper.

Based on our new linear-spline techniques, we also presented a framework for partitioning data into buckets, such that the average approximate errors in processing aggregates may be minimized. Consider that the techniques in [9] are not applicable to our framework, and the corresponding optimization problem in our framework tends to be computationally intractable. We approach this problem first by a thorough mathematic analysis of the dominant parts in our optimization model to obtain a simplified model, which is quite close to the original optimization model. Secondly, we present an efficient algorithm to deliver the optimal solution to the simplified model; such a result may serve as a near optimal solution to our original optimization model. These are the second contribution of the paper.

Our experiment results demonstrated that the new histogram construction techniques in this paper significantly improve the accuracies of the existing histogram techniques, as well as the existing wavelet techniques. Our experiment shows that the minimum improvement of accuracy is about 50%.

1.2. Organization

The rest of this paper is organized as follows. Section 2 presents the background knowledge and the motivation of this research. Section 3 presents the new linear-spline histogram techniques. Section 4 describes the details of our new data partitioning model, as well as the data partitioning algorithms. Section 5 reports our experiment results. This is followed by a conclusion and remarks.

2. Background

Given a relation R and an attribute X of R, the *domain* D of X is the set of all possible values of X, and a finite set V ($\subseteq D$) denotes the distinct values of X in an *instance* r of R. Let V be ordered; that is $V = \{v_i : 1 \leqslant i \leqslant n\}$ where $v_i < v_j$ if $i < j$. The instance r of R restricted to X is denoted by T, and can be represented as follows:

$$T = \{(v_1, f_1), (v_2, f_2), \ldots, (v_n, f_n)\}.$$

In T, each v_i is distinct and is called a *value* of T; and f_i is the occurrence of v_i in T and is called the *frequency* of v_i. Note that in this paper T may be called *data distribution* or *data set* alternately; that is, "data set" and "data distribution" will be used as synonyms in this paper. Fig. 1(a) shows an example of data distribution.

A *histogram* on data set T is constructed by the following two steps.

Step 1: Partitioning the values of T into β ($\geqslant 1$) disjoint intervals (called *buckets*) – $\{B_i : 1 \leqslant k \leqslant \beta\}$, such that each value in B_k is smaller than that in B_l if $k < l$.

Step 2: Approximately representing the frequencies and values in each bucket.

The *width* of a bucket is $v_j - v_i$ where v_j and v_i are respectively the maximal value and minimum value in the bucket. The *spread* s_i of v_i (for $1 \leqslant i \leqslant n-1$) is defined as $v_{i+1} - v_i$. The *area* a_i of v_i is defined as $f_i * s_i$ for $1 \leqslant i \leqslant n$ where we make $s_n = 1$. Fig. 1(b) shows a possible histogram of the data distribution in Fig. 1(a).

2.1. Existing histogram techniques

In a histogram, each bucket stores an approximate representation of the original data distribution in the bucket; this involves two approximate representations in a bucket: an approximation of the value distribution and an approximation of the frequency distribution. To minimize the storage space of the information to be stored in a bucket, most existing data partitioning algorithms aim to partition the data such that the data distribution in each bucket is as close to a uniform distribution as possible. Consequently, the value distribution in a bucket may be approximately represented by the *uniform-spread assumption* (that is, the values are assumed to evenly span the bucket [26]), while the frequency distribution may be approximately represented by a constant [7,25] – the average frequency in the bucket.

With the uniform-spread assumption, to represent a value distribution in a bucket we only need to store the minimum and maximum values in the bucket together with the number of distinct values in this bucket; the other values can be approximately derived according to this assumption. Fig. 3(a) shows the information to be stored for each bucket.

The existing histogram techniques for generating buckets, based on the above histogram representation, may be summarized below according to different data partitioning goals. We also give an example for each histogram. Note that all the example histograms are constructed on the data distribution in Fig. 1(a).

- *Equi-width* [22]: bucket widths equal each other. Fig. 1(b) shows an example.
- *Equi-sum* [24,22]: the sum of the frequencies in each bucket is the same. Fig. 2(a) shows an example.
- *Maxdiff* [26]: the data distribution is partitioned such that the differences of the frequencies between adjacent boundaries are maximized. Fig. 2(b) shows an example.

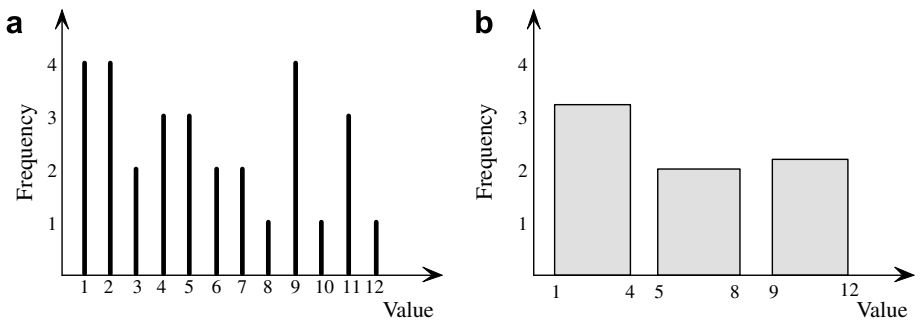


Fig. 1. Data distribution and histogram. (a) Data distribution and (b) histogram (equi-width).

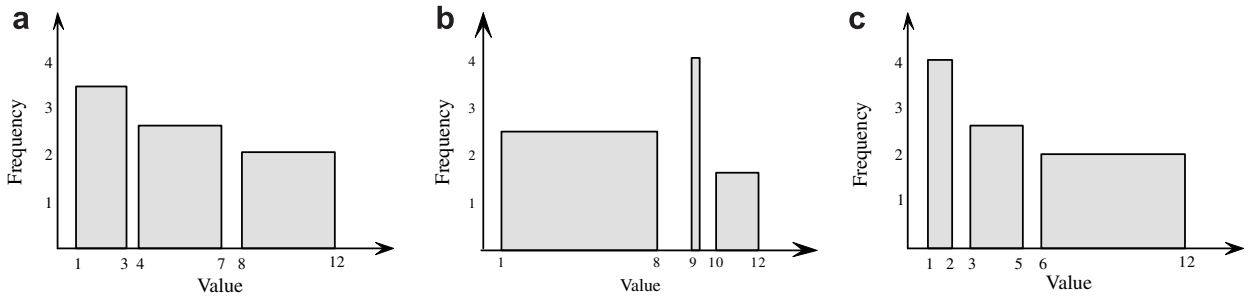


Fig. 2. Various histograms. (a) Histogram (equi-sum), (b) histogram (Maxdiff) and (c) histogram (V-optimal).

- *V-optimal* [16,15,17,21]: Partition data such that $\sum_{j=1}^{\beta}\sum_{k=1}^{n_j}(f'_{j,k} - f_{j,k})^2$ is minimized, where β is the number of buckets, n_j is the number of distinct values in the jth bucket. $f_{j,k}$ is the original frequency (or area) of kth distinct value in the jth bucket, and $f'_{j,k}$ is the corresponding approximate frequency (or area). Fig. 2(c) shows an example.

Experiment results [21] suggest that in most applications, Maxdiff and V-optimal outperform equi-width and equi-sum [26]. V-optimal usually leads to more accurate approximate aggregation results than Maxdiff [20] does.

Gilbert et al. [9] proposed to investigate the average error minimization for a *uniform* distributed query pattern; that is each range takes the same probability. In their investigation, they assume the whole value set is evenly distributed; and thus the original value set can be exactly retrieved from a histogram.

Observing the limitation of the frequency approximation in a bucket, [21] proposed a straight line $q * v + c$ to represent the frequency distribution in a bucket; this is a generalization of the conventional representation (the representation above) where $q = 0$ and c is set to the average frequency. Note that in this kind of histogram, the information required for a bucket is illustrated in Fig. 3(b).

Fig. 4 illustrates an example. With respect to the data distribution in Fig. 4, the line in Fig. 4(b) is much closer to the original data distribution than that in Fig. 4(a) where the horizontal line corresponds to the average frequency in a *conventional* histogram. Further, it is well-known that an application of the *least-square technique* [29] will minimize the errors in matching by a linear model. These motivated König et al. to develop the *linear-spline histogram with least-square method* [21] (LSLS):

- *LSLS*: In this model, the frequencies in each bucket are approximated by a linear function $l_j(v) = q_j * v + c_j$ where j represents the jth bucket. The goal is to find a histogram with β buckets, such that $\sum_{j=1}^{\beta}\sum_{k=1}^{n_j}(l_j(v_{j,k}) - f_{j,k})^2$ is minimized where n_j is the number of entries in the jth bucket and $v_{j,k}$, $f_{j,k}$ are the kth *value* and *frequency* of jth bucket. Note that in this model, for each bucket B_j, the least-square method is used to fix the variables q_j and c_j.

a

Starting Value	End Value	Number of Values	Average Frequency

b

Starting Value	End Value	Number of Values	q	c

Fig. 3. Two representations. (a) Conventional representation and (b) linear-spline representation.

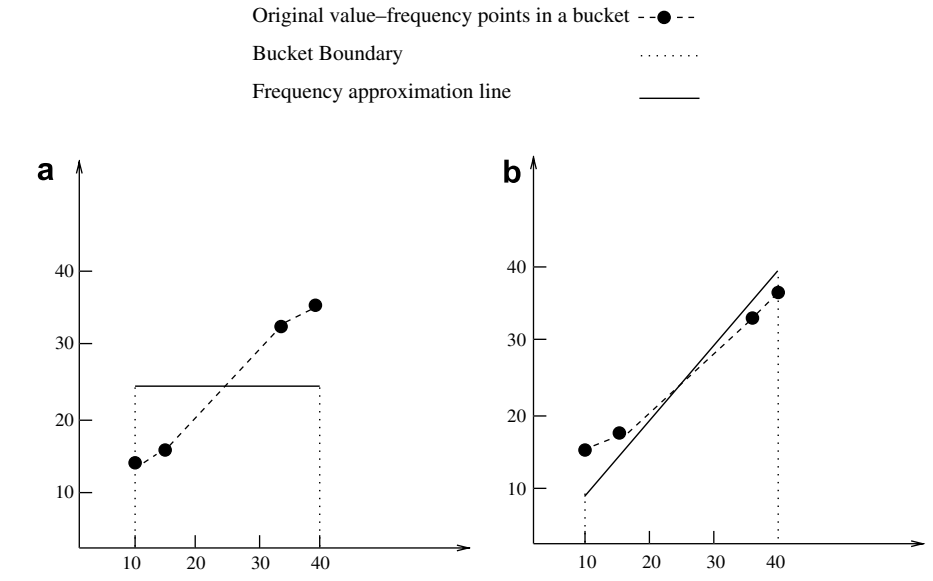


Fig. 4. *Uniform frequency* vs. *linear function*. (a) Uniform frequency assumption and (b) linear function assumption.

The paper [4] goes a step further by proposing a *continuous* model to approximately processing COUNT against a histogram; and therefore the number of distinct values is no longer required to be recorded. Taking this advantage, [4] proposed an encoding schema to represent more detailed information about the frequency distribution in a bucket within an integer space which was used to represent the number of distinct values before. Therefore, the space used in an encoded histogram is the same as that in a conventional histogram but the encoded histogram has much more information than that in a conventional histogram. The encoded scheme is proposed to be used as a post-process after data are partitioned. So it may also be applied, as a post-process, to our technique.

2.2. Motivation

In this section, we show the motivation of this research. We analyze the existing linear-spline technique and the existing optimization models for data partitioning.

2.2.1. Least-square: best alternative?

Clearly, the LSLS model can simulate an arbitrary data distribution more closely than a conventional data model does; this is because that a horizontal line is a special member in the family of linear splines. Further, LSLS usually provides the closest matching within linear models. However, it should be noted that a given data distribution does not always follow the uniform-spread distribution for its values' distribution. Consequently, LSLS may not generally bring the best approximate solutions; we will show this in our experiments. In fact, the LSLS model cannot generally guarantee the following two properties:

P1: The approximation on total frequencies in a bucket of a histogram is the same as that in the original data set.
P2: The approximation on summation of all the values from the data set restricted to a bucket are the same as that (i.e., $\sum_i v_i * f_i$) in the original data set restricted to the bucket.

Note that in contrast, any conventional model has the property P1. For example, suppose that a bucket holds the following data distribution:

$$\{(10,25),(20,45),(50,105),(60,125),(70,145)\}$$

Fig. 5(a) and (b) illustrate the information stored in a conventional model and in LSLS, respectively.

The table below summarises the results by querying the original data, the conventional histogram, and the LSLS histogram.

Count(X) with $10 \leqslant X \leqslant 70$		Sum(X) with $10 \leqslant X \leqslant 70$	
Exact answer	445	Exact answer	24 050
Conventional histogram holds P1	445	*Conventional histogram holds* P2	17 800
LSLS histogram does not hold P1	425	*LSLS histogram does not hold* P2	21 500

a

	Lowest Value	Highest Value	Number of Values	Average Frequency
Conventional Histogram	10	70	5	89

b

	Lowest Value	Highest Value	Number of Values	Slope q	Interception c
LSLS Histogram	10	70	5	2	5

Fig. 5. Bucket representations. (a) Conventional model and (b) LSLS model.

Since the properties P_1 and P_2 are also critical in histogram techniques, this motives our research on finding new linear-spline model.

2.2.2. Intuitively optimal: always best?

Note that the value distribution may also have a great impact on an approximation query processing result. This has been addressed in [26], where an *area* variance was proposed to replace a pure frequency variance. It has been shown [26] through empirical results that the area variance may greatly improve the accuracy comparing with a pure frequency variance. However, the area method may sometimes still fail to reduce the errors, caused by a value distribution approximation, in approximate aggregation. Below are the examples to show the limitation of the existing data partitioning methods.

In the example as depicted in Fig. 6(a), a data distribution consists of eight distinct values $\{1,2,3,5,405,409,411,412\}$. The frequencies of 1, 2, 411, and 412 are 1000, respectively; and the frequencies of 3, 5, 405 and 409 are 1010, respectively. In the example, a partitioning method based on either Maxdiff [26] or V-optimal [20] (including the linear-spline model in [21]) will always produce the following partition, as shown in Fig. 6(a), if only three buckets are given:

Partition 1A: Bucket 1: $\{1,2\}$. Bucket 2: $\{3,5,405,409\}$. Bucket 3: $\{411,412\}$.

Suppose that we have three range aggregates to answer: (1) $1 < value < 3$, (2) $10 < value < 390$, and (3) $409 < value < 412$. Clearly, a histogram (conventional or linear-spline) built based on Partition 1A is able to give an exact answer to the first query and the third query but the error will be at least *2020* for the second query depends on the query processing method: *2020* for the uniform-spread assumption, and 7371 for the continuous model. In this example, consider the following partition:

Partition 1B: Bucket 1: $\{1,2,3\}$. Bucket 2: $\{5,405\}$. Bucket 3: $\{409,411,412\}$.

A histogram (conventional or linear-spline based) built against Partition 1B will produce the exact solution for the second query but approximate solutions for the first query and the third query within an error b ($b < 10$) respectively. Clearly, if these three queries are equally important and we want to have the minimum average error, we would prefer to have Partition 1B.

The variations of V-optimal and Maxdiff, respectively, may fix the problem in the example in Fig. 6(a) by using the *area* ($f_i * s_i$) instead of f_i. They both produced following data partition with a similar performance (a little bit worse than) to that of Partition 1B.

Partition 1C: Bucket 1: $\{1,2,3\}$. Bucket 2: $\{5\}$. Bucket A: $\{405,409,411,412\}$.

However, with respect to the example in Fig. 6(b) both the above variations for V-optimal and Maxdiff perform poorly; and they produced the data partition as illustrated in Fig. 6(b). With respect to the same set of transactions above, the following data partition is the best for a similar reason as stated above.

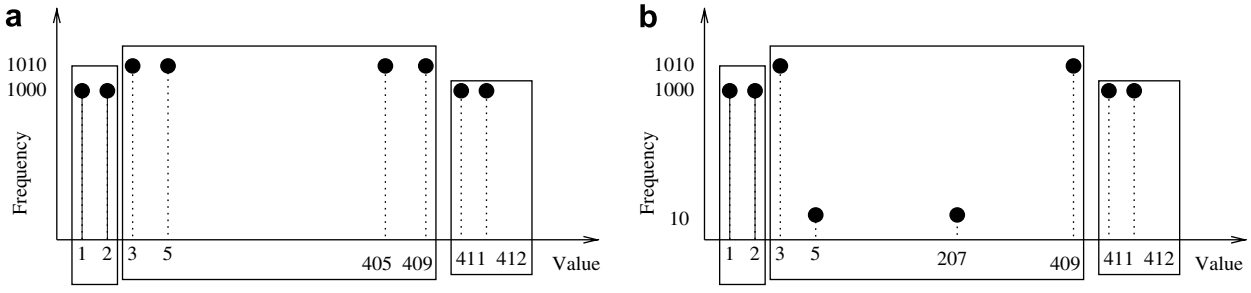


Fig. 6. Two data distributions.

Partition 2B: Bucket 1: {1,2,3}. Bucket 2: {5,207}. Bucket 3: {409,411,412}.

Note that the data partitioning models in [9] will also lead to Partition 1A for the example in Fig. 6(a). It should also mention that the histogram construction techniques in [21] cannot handle well the example in Fig. 6(a), even by the variation where the area parameter is used.

Clearly, in the above two examples we want a data partitioning algorithm to generate Partition 1B for the example in Fig. 6(a) and Partition 2B for the example in Fig. 6(b); this can be done through minimizing the average error. In fact our algorithm can guarantee Partition 1B and Partition 2B, respectively, for the two examples.

3. New models of linear-spline histogram

Suppose that a data distribution $T = \{(v_1,f_1),(v_2,f_2),\ldots,(v_d,f_d)\}$ is given, which is partitioned into β buckets $\{B_k : 1 \leqslant k \leqslant \beta\}$. In each bucket B_k, a linear spline, $l_k(x) = q_k * x + c_k$, is used to approximate the corresponding data distribution. The total variances of the approximation by linear splines are

$$\sum_{k=1}^{\beta} \sum_{v_i \in B_k} (l_k(v_i) - f_i)^2$$

Below, we show two new ways to determine q_k and c_k corresponding to each bucket B_k.

3.1. LSCG: linear-spline histogram with count guaranteed

Suppose that a bucket B_k is given. A linear function $l'_k(x) = q'_k * x + c'_k$ is used to approximate the data distribution $\{(v_{i_k}, f_{i_k}), (v_{i_k+1}, f_{i_k+1}), \ldots, (v_{i_k+j_k}, f_{i_k+j_k})\}$ in B_k. In this model, we will first enforce that the total approximate frequency of B_k, computed from the linear function $l'_k(x)$, equals the original one; that is property P1 in Section 2.2.1 is hold. Then, the least-square method is used. That is, we first enforce the following equation:

$$\sum_{m=0}^{j_k} (q'_k * v'_{i_k+m} + c'_k) = \sum_{m=0}^{j_k} f_{i_k+m} \tag{1}$$

Here, v'_{i_k+m} represents the $(m+1)$th smallest distinct value in B_k with respect to a histogram. According to the uniform-spread assumption, v'_{i_k+m} should be calculated in (2). Note that this v'_{i_k+m} will be used in approximate query processing against the histogram instead of v_{i_k+m} (if $m \neq 0$ or $m \neq j_k$) in the original distribution.

$$v'_{i_k+m} = v_{i_k} + m * \frac{v_{i_k+j_k} - v_{i_k}}{j_k} \tag{2}$$

From (1) and (2), we can derive

$$c'_k = \overline{f_k} - \frac{v_{i_k} + v_{i_k+j_k}}{2} * q'_k \tag{3}$$

Here, $\overline{f_k}$ is the average frequency in B_k. Next we use the least-square method to determine a'_k. That is

$$\frac{\mathrm{d}(\sum_{m=0}^{j_k} (q'_k * v_{i_k+m} + \overline{f_k} - \frac{v_{i_k}+v_{j_k}}{2} * q'_k - f_{i_k+m})^2)}{\mathrm{d}(q'_k)} = 0 \tag{4}$$

Note that in (4), we use (3) to replace c'_k first.

Thus we derive

$$q'_k = \frac{12 * \sum_{m=0}^{j_k} (f_{i_k+m} * (m+1)) - 6 * (j_k+2) * \sum_{m=0}^{j_k} f_{i_k+m}}{(v_{i_k+j_k} - v_{i_k}) * (j_k+1)(j_k+2)} \tag{5}$$

Note that in (5), the denominator equals zero if and only if $v_{i_k} = v_{i_k+j_k}$ (i.e., B_k contains only one distinct value v_{i_k}). Consequently, we assign that $q'_k = 0$ and $c'_k = f_{i_k}$ if B_k contains only one distinct value.

3.2. LSCSG: linear-spline histogram with count and sum guaranteed

As with *LSCG*, a linear function $l''_k(x) = q''_k * x + c''_k$ is used to approximate the data distribution in bucket B_k. However, q''_k and c''_k will be chosen to make the total frequency and total summation approximately calculated over B_k in the histogram are the same as those from the original data set, simultaneously; that is, enforce P1 and P2 in Section 2.2.1. This requires that q''_k and c''_k satisfy the following linear equations:

$$\begin{cases} \sum_{m=0}^{j_k}(q''_k * v'_{i_k+m} + c''_k) = \sum_{m=0}^{j_k} f_{i_k+m} & \text{(a)} \\ \sum_{m=0}^{j_k}(v'_{i_k+m} * (q''_k * v'_{i_k+m} + c''_k)) = \sum_{m=0}^{j_k}(f_{i_k+m} * v_{i_k+m}) & \text{(b)} \end{cases} \tag{6}$$

Similarly, here

$$v'_{i_k+m} = v_{i_k} + m * \frac{v_{i_k+j_k} - v_{i_k}}{j_k}$$

Solving Eq. (a) and (b) of (6), we get a''_k and b''_k as follows.

$$\begin{cases} q''_k = \frac{\sum_{m=0}^{j_k}(f_{i_k+m} * v_{i_k+m}) * (j_k+1) - \sum_{m=0}^{j} f_{i_k+m} * \sum_{m=0}^{j_k} v'_{i_k+m}}{(m+1) * \sum_{m=0}^{j_k}(v'_{i_k+m})^2 - \sum_{m=0}^{j_k} v'_{i_k+m} * \sum_{m=0}^{j_k} v'_{i_k+m}} & \text{(a)} \\ c''_k = \frac{\sum_{m=0}^{j_k} f_{i_k+m} * \sum_{m=0}^{j_k}(v'_{i_k+m})^2 - \sum_{m=0}^{j_k}(f_{i_k+m} * v_{i_k+m}) * \sum_{m=0}^{j_k} v'_{i_k+m}}{(j_k+1) * \sum_{m=0}^{j_k}(v'_{i_k+m})^2 - \sum_{m=0}^{j_k} v'_{i_k+m} * \sum_{m=0}^{j_k} v'_{i_k+m}} & \text{(b)} \end{cases} \tag{7}$$

It can be immediately verified that the denominators in (7) equals zero if and only if $v_{i_k} = v_{i_k+j_k}$ (i.e., B_k contains only one distinct value). Consequently, we assign that $a''_k = 0$ and $b''_k = f_{i_k}$ if B_k contains only one distinct value.

It is worth to note that under uniform-spread assumption in the value space, our two new histogram models, LSCG and LSCSG, will degrade to LSLS model.

Theorem 1. *Suppose that a value space follows the uniform-spread assumption. Then, LSCSG, LSCG and LSLS are equivalent to each other.*

Proof. Suppose that a bucket B_k is given. The data distribution in B_k can be represented as $\{(v_{i_k}, f_{i_k}), (v_{i_k+1}, f_{i_k+1}), \ldots, (v_{i_k+j_k}, f_{i_k+j_k})\}$. A linear function $l_k(x) = q_k * x + c_k$ is used to approximate this data distribution. Let $\overline{f_k}$ represents the average frequency in B_k and $\overline{v_k}$, the average value. As discussed earlier, the two variables in $l_k(x)$ are chosen as follows against the three different models LSLS, LSCG and LSCSG, respectively.

In LSLS:

$$\begin{aligned} q_k &= \frac{\sum_{m=0}^{j_k}(v_{i_k+m} - \overline{v_k}) * (f_{i_k+m} - \overline{f_k})}{\sum_{m=0}^{j_k}(v_{i_k+m} - \overline{v_k})^2} \\ c_k &= \overline{f_k} - q_k * \overline{v_k} \end{aligned} \tag{8}$$

In LSCG:

$$\begin{aligned} q'_k &= \frac{12 * \sum_{m=0}^{j_k}(f_{i_k+m} * (m+1)) - 6 * (j_k+2) * \sum_{m=0}^{j_k} f_{i_k+m}}{(v_z i_k + j_k - v_{i_k}) * (j_k+1)(j_k+2)} \\ c'_k &= \overline{f_k} - \frac{v_{i_k} + v_{i_k+j_k}}{2} * a'_k \end{aligned} \tag{9}$$

In LSCSG:

$$q''_k = \frac{\sum_{m=0}^{j_k}(f_{i_k+m} * v_{i_k+m}) * (j_k+1) - \sum_{m=0}^{j} f_{i_k+m} * \sum_{m=0}^{j_k} v'_{i_k+m}}{(m+1) * \sum_{m=0}^{j_k}(v'_{i_k+m})^2 - \sum_{m=0}^{j_k} v'_{i_k+m} * \sum_{m=0}^{j_k} v'_{i_k+m}}$$
$$c''_k = \frac{\sum_{m=0}^{j_k} f_{i_k+m} * \sum_{m=0}^{j_k}(v'_{i_k+m})^2 - \sum_{m=0}^{j_k}(f_{i_k+m} * v_{i_k+m}) * \sum_{m=0}^{j_k} v'_{i_k+m}}{(j_k+1) * \sum_{m=0}^{j_k}(v'_{i_k+m})^2 - \sum_{m=0}^{j_k} v'_{i_k+m} * \sum_{m=0}^{j_k} v'_{i_k+m}} \quad (10)$$

Since the given value space follows the uniform-spread assumption, we have:

$$v_{i_k+m} = v'_{i_k+m} \quad (0 \leqslant m \leqslant j_k) \quad \text{(a)}$$
$$v'_{i_k+m} = v_{i_k} + \tfrac{v_{j_k} - v_{i_k}}{j_k+1} * m \quad (0 \leqslant m \leqslant j_k) \quad \text{(b)} \quad (11)$$

From (a) and (b) of (11), we can derive:

$$\sum_{m=0}^{j_k} v_{i_k+m} = (j_k+1) * v_{i_k} + \frac{v_{j_k} - v_{i_k}}{j_k+1} * \frac{j_k * (j_k+1)}{2}$$
$$\sum_{m=0}^{j_k} v^2_{i_k+m} = (j_k+1) * v^2_{i_k} + v_{i_k} * \frac{v_{j_k} - v_{i_k}}{j_k+1} * j_k(j_k+1) + \frac{j_k * (j_k+1) * (2 * j_k+1)}{6} * \left(\frac{v_{j_k} - v_{i_k}}{j_k+1}\right)^2 \quad (12)$$

Replacing corresponding items of Eqs. (8)–(10) by (a) of (11) and (12), we can easily get

$$q_k = q'_k = q''_k$$
$$c_k = c'_k = c''_k \quad \square \quad (13)$$

4. Minimization of average aggregate errors

In this section, we will present a new histogram partition technique. This technique is based on the minimization of average errors.

Suppose that a data set (data distribution) $T = \{(v_1,f_1),(v_2,f_2),\ldots,(v_n,f_n)\}$ is given, which is partitioned into β disjoint buckets $\{B_i : 1 \leqslant i \leqslant \beta\}$, where the partition is denoted by B. Against a histogram built on the data partition B, an aggregate over a range $[x,y]$ will be processed by the following two steps:

- Allocate x and y in the partition B.
- Using the corresponding buckets to answer the query.

Note that while processing aggregates over a range $[x,y]$ against a data set T, we need to find all the distinct values in $[x,y]$ to answer the queries. However, it is impossible to find exact distinct values in $[x,y]$ in a histogram of T due to an approximate representation of the value distribution in each bucket unless values are evenly distributed. In fact, in a bucket of a histogram the ith distinct value may be approximately obtained by a uniform-spread model [18] or a continuous model [7,4]. Note that the continuous model in a histogram can support only one aggregate where the histogram is built based on the aggregate. Unless we will build a histogram for each aggregate, the uniform-spread model should be used in order to approximately process all range aggregates. In this paper, we will use the uniform-spread model/assumption.

Given a partition B, a histogram H on B, and a range $[x,y]$, the error generated by an approximate query over the range against H is denoted by $e^H_B(x,y)$. Note that in this paper, we will always choose the specific linear-spline form LSCSG; this implies that once a data partition is given, H is unique. Consequently, $e^H_{T,B}(x,y)$ may be abbreviated to $e_{T,B}(x,y)$. There are several ways to specify an error [18]. Without loss of generality, in this paper we use an absolute error function; that is,

$$e_{T,B}(x,y) = |A_T(x,y) - A'_{T,B}(x,y)| \quad (14)$$

Here, $A_T(x,y)$ represent the exact answer to a range aggregate over $[x,y]$, and $A'_{T,B}(x,y)$ is an approximate answer. Note that there are many types of aggregates. In this paper, we will use COUNT aggregate in our study for generating a data partition.

Suppose that the probability distribution for ranges $[x,y]$ is a continuous distribution over a data set T, which can be represented by a *probability density function* $P(x,y)$ [29] where $\int_{v_1}^{v_n}\int_{x}^{v_n} P(x,y)\mathrm{d}x\,\mathrm{d}y = 1$. The average error for range aggregations (COUNT) over T against a data partition B is

$$E_{T,B} = \int_{v_1}^{v_n}\left(\int_{x}^{v_n} P(x,y) * e_{T,B}(x,y)\mathrm{d}y\right)\mathrm{d}x \tag{15}$$

Our goal is to find a data partition B such that (15) will be minimized. In this paper we will investigate the minimization of (15) with respect to a uniform query distribution; that is $P(x,y) = \frac{2}{(v_n-v_1)^2}$. Therefore, (15) can be represented as follows:

$$E_{T,B} = \frac{2}{(v_n - v_1)^2}\int_{v_1}^{v_n}\left(\int_{x}^{v_n} e_{T,B}(x,y)\mathrm{d}y\right)\mathrm{d}x \tag{16}$$

The problem of finding a data partition B to minimize the goal function tends to be computation intractable. Below we first present a reasonably tight upper-bound on the goal function in (16); this will be done based on a mathematic analysis of the goal function. Then we will adopt the dynamic programming technique developed in [3,20] to generate an optimal data partition against the upper-bound.

The rest of the section will be organized as follows. We first describe the linear-spline model used in our study. Then we will provide an upper-bound for (16). This will be followed by an efficient optimal algorithm against the upper-bound.

4.1. A linear-spline model

As stated in Section 3, we proposed to use a new linear-spline $qv + c$ to represent the frequency in a bucket. We also proposed two ways to fix q and c. Our performance evaluation showed that LSCSG model is slightly better than LSCG. Thus, we propose to use LSCSG model to fix q and c in our data partitioning techniques presented in this section – Section 4:

> q and c should be chosen to make the COUNT and SUM over the bucket by $qv + c$ are, respectively, equal to those over the bucket by using the original data set.

Note that in a bucket $[a_i,b_i]$ with k distinct values, the k distinct values in a histogram restricted to the bucket $[a_i,b_i]$ may be approximately retrieved by the following formula (17) in the uniform-spread assumption.

$$v'_l = a_i + l * \frac{b_i - a_i}{k-1} \quad (\text{for } 0 \leqslant i \leqslant k-1) \tag{17}$$

4.2. Upper-bounds

In this subsection, we work out good upper-bounds for (16). The deduction is quite mathematics involved.

Suppose that $B = \{B_i = [a_i,b_i] : 1 \leqslant i \leqslant \beta\}$ is a disjoint partition, with β buckets, of the distinct values of T, where $a_1 = v_1$ and $b_\beta = v_n$. Suppose that C_{B_i} denotes the COUNT value over B_i against T. Clearly, the COUNT value over B_i against the histogram using the linear-spline model LSCSG is also C_{B_i}. By this property and an immediate math deduction, below are the important properties for $e_{T,B}(x,y)$ regarding the five possibilities for x and y for a given range $[x,y]$.

Case 1: x and y fall into different bucket; that is, $x \in [a_i, b_i]$ and $y \in [a_j,b_j]$ $(i < j)$. In this case,

$$e_{T,B}(x,y) \leqslant e_{T,B}(x,b_i) + e_{T,B}(a_j,y) \tag{18}$$

Case 2: x falls between buckets, y falls into a bucket; that is, $x \in (b_i,a_{i+1})$ and $y \in [a_j,b_j]$ $(i < j)$. In this case,

$$e_{T,B}(x,y) = e_{T,B}(a_j,y) \tag{19}$$

Case 3: x falls into a bucket, y falls between buckets; that is, $x \in [a_i,b_i]$ and $y \in (b_j,a_{j+1})$ $(i \leqslant j)$. In this case,

$$e_{T,B}(x,y) = e_{T,B}(x,b_i) \tag{20}$$

Case 4: x and y both fall between buckets; that is, $x \in (b_i, a_{i+1})$ and $y \in (b_j, a_{j+1})$ $(i \leqslant j)$. In this case,

$$e_{T,B}(x,y) = 0 \tag{21}$$

Case 5: x and y fall into one bucket B_i; that is, $x, y \in [a_i, b_i]$. In this case,

$$e_{T,B}(x,y) \leqslant e_{T,B}(a_i,x) + e_{T,B}(x,x) + e_{T,B}(y,y) + e_{T,B}(y,b_i) \tag{22}$$

Since the original data distribution is partitioned into β buckets, the query $[x,y]$ can have the above five forms. By the four properties (18)–(21), we can derive the following upper-bound for (16).

Theorem 2.

$$E_{T,B} \leqslant \frac{2}{(v_n - v_1)^2} \sum_{i=1}^{\beta} \left(\int_{a_i}^{b_i} \int_{x}^{b_i} e_{T,B}(x,y)\mathrm{d}y\,\mathrm{d}x + (v_n - b_i) \int_{a_i}^{b_i} e_{T,B}(x,b_i)\mathrm{d}x + (a_i - v_1) \int_{a_i}^{b_i} e_{T,B}(a_i,x)\mathrm{d}x \right) \tag{23}$$

(*Note that* $a_1 = v_1$ *and* $b_\beta = v_n$.)

Proof. By properties (19)–(21), the following formula can be immediately obtained.

$$\begin{aligned} E_{T,B} = \frac{2}{(v_n - v_1)^2} \Bigg(& \sum_{i=1}^{\beta} \left(\int_{a_i}^{b_i} \int_{x}^{b_i} e_{T,B}(x,y)\mathrm{d}y\,\mathrm{d}x \right) + \sum_{i=1}^{\beta-1} \sum_{j=i+1}^{\beta} \left(\int_{a_i}^{b_i} \int_{a_j}^{b_j} e_{T,B}(x,y)\mathrm{d}y\,\mathrm{d}x \right) \\ & + \sum_{i=1}^{\beta-1} \sum_{j=i+1}^{\beta} \left(\int_{b_i}^{a_{i+1}} \int_{a_j}^{b_j} e_{T,B}(a_j,y)\mathrm{d}y\,\mathrm{d}x \right) + \sum_{i=1}^{\beta-1} \sum_{j=i}^{\beta} \left(\int_{a_i}^{b_i} \int_{b_j}^{a_{j+1}} e_{T,B}(x,b_i)\mathrm{d}x\,\mathrm{d}y \right) \Bigg) \end{aligned} \tag{24}$$

Now, by the property (18) the following inequality immediately follows from (24).

$$\begin{aligned} E_{T,B} \leqslant \frac{2}{(v_n - v_1)^2} \Bigg(& \sum_{i=1}^{\beta} \left(\int_{a_i}^{b_i} \int_{x}^{b_i} e_{T,B}(x,y)\mathrm{d}y\,\mathrm{d}x \right) + \sum_{i=1}^{\beta-1} \sum_{j=i+1}^{\beta} \left(\int_{a_i}^{b_i} \int_{a_j}^{b_j} (e_{T,B}(x,b_i) + e_{T,B}(a_j,y))\mathrm{d}y\,\mathrm{d}x \right) \\ & + \sum_{i=1}^{\beta-1} \sum_{j=i+1}^{\beta} \left(\int_{b_i}^{a_{i+1}} \int_{a_j}^{b_j} e_{T,B}(a_j,y)\mathrm{d}y\,\mathrm{d}x \right) + \sum_{i=1}^{\beta-1} \sum_{j=i}^{\beta} \left(\int_{a_i}^{b_i} \int_{b_j}^{a_{j+1}} e_{T,B}(x,b_i)\mathrm{d}x\,\mathrm{d}y \right) \Bigg) \end{aligned} \tag{25}$$

The theorem immediately follows from (25) by a simple calculation; that is, we can verify that the right side of (23) is equal to the right side of (25). □

In the upper-bound given by the right side of (23), there are three parts. Below we show that the second and third parts may be represented by a similar form.

Theorem 3. $\forall B_i \in B$, $\int_{a_i}^{b_i} e_{T,B}(x,b_i)\mathrm{d}x = \int_{a_i}^{b_i} e_{T,B}(a_i,x)\mathrm{d}x$.

Proof. Note that $\forall B_i$, the COUNT value over B_i against our histogram construction is always the same as that in T. Thus,

$$e_{T,B}(a_i,x) = |A_T(a_i,x) - A'_{T,B}(a_i,x)| = |A_T(x,b_i) - A'_{T,B}(x,b_i) - (A_T(x,x) - A'_{T,B}(x,x))| \tag{26}$$

Note that $A_T(x,x) - A'_{T,B}(x,x)$ takes at most $2k_i$ non-zero values in B_i where k_i is the number of distinct values in B_i. This implies,

$$\int_{a_i}^{b_i} |A_T(x,b_i) - A'_{T,B}(x,b_i) - (A_T(x,x) - A'_{T,B}(x,x))| = \int_{a_i}^{b_i} |A_T(x,b_i) - A'_{T,B}(x,b_i)| = \int_{a_i}^{b_i} e_{T,B}(x,b_i). \qquad \square$$

From Theorem 3, it follows that the right side of (23) may be re-written as

$$\frac{2}{(v_n - v_1)^2} \sum_{i=1}^{\beta} \left(\int_{a_i}^{b_i} \int_{x}^{b_i} e_{T,B}(x,y)\mathrm{d}y\,\mathrm{d}x + (v_n - v_1 - (b_i - a_i)) \int_{a_i}^{b_i} e_{T,B}(a_i,x)\mathrm{d}x \right) \tag{27}$$

The calculation of the first part in (27) may have to be carried out in a quadratic time. By the property (22), below we show that the first part is usually not a dominant factor.

Theorem 4. $\forall B_i \in B$, $\int_{a_i}^{b_i}\int_{x}^{b_i} e_{T,B}(x,y)\mathrm{d}y\,\mathrm{d}x \leqslant (b_i - a_i)\int_{a_i}^{b_i} e_{T,B}(a_i,x)\mathrm{d}x$.

Proof. By property (22), the following is immediate.

$$\int_{a_i}^{b_i}\int_{x}^{b_i} e_{T,B}(x,y)\mathrm{d}y\,\mathrm{d}x \leqslant \int_{a_i}^{b_i}\int_{x}^{b_i} (e_{T,B}(a_i,x) + e_{T,B}(x,x) + e_{T,B}(y,y) + e_{T,B}(y,b_i))\mathrm{d}y\,\mathrm{d}x \tag{28}$$

By a similar argument as that in the proof of Theorem 3, the right side of (28) is equal to

$$\int_{a_i}^{b_i}\int_{x}^{b_i} (e_{T,B}(a_i,x) + e_{T,B}(y,b_i))\mathrm{d}y\,\mathrm{d}x \tag{29}$$

By a simple integration transformation and calculation, together with Theorem 3, (29) is equal to

$$(b_i - a_i)\int_{a_i}^{b_i} e_{T,B}(a_i,x)\mathrm{d}x \qquad \square$$

From Theorem 2, (27), Theorems 3 and 4, the following is immediate:

$$E_{T,B} \leqslant \frac{2}{v_n - v_1}\sum_{i=1}^{\beta}\left(\int_{a_i}^{b_i} e_{T,B}(a_i,x)\mathrm{d}x\right) \tag{30}$$

It should be clear, $v_n - v_1 - (b_i - a_i)$ is usually larger than $b_i - a_i$ (for $1 \leqslant i \leqslant n$) unless there is a huge bucket. In fact, in our experiment we are always able to obtain the same data partitioning result for the goal functions in (30) and (27) respectively. However, the upper-bound in (30) leads to a much faster algorithm than the upper-bound in (27) does. Below we present our algorithm to produce a data partition B with β buckets to minimize (30).

4.3. Data partition algorithm

In this subsection, we will present an efficient algorithm to partition a data set $T = \{(v_1,f_1), (v_2,f_2),\ldots,(v_n,f_n)\}$ such that the upper-bound in (30) is minimized. Since v_1 and v_n are two constants for a given T, we need only to find a data partition B with β buckets such that the following goal function is minimized.

$$\sum_{i=1}^{\beta}\left(\int_{a_i}^{b_i} e_{T,B}(a_i,x)\mathrm{d}x\right) \tag{31}$$

We first show a linear algorithm to calculate an integration $\int_{a_i}^{b_i} e_{T,B}(a_i,x)\mathrm{d}x$.

4.3.1. Integration calculation

Our algorithm for calculating the integration $\int_{a_i}^{b_i} e_{T,B}(a_i,x)\mathrm{d}x$ for a bucket B_i is based on a *merge* paradigm in external merge sort algorithm [27]. It runs in linear time with respect to the number of distinct values in $[a_i,b_i]$.

Suppose that $B_i = [a_i,b_i]$ contains k distinct values in T, that is, B_i contains $\{(v_{j_i},f_{j_i}),(v_{j_i+1},f_{j_i+1}),\ldots,(v_{j_i+k-1},f_{j_i+k-1})\}$, and $v_{j_i} = a_i$ and $v_{j_i+k-1} = b_i$. In a histogram representation of B, the k distinct values are approximately represented, by a uniform-spread assumption, as the following:

$$v'_{j_i+l} = a_i + l * \frac{b_i - a_i}{k-1} \quad (0 \leqslant l \leqslant k-1)$$

The k corresponding frequencies are represented as

$$f'_{j_i+l} = q * v'_{j_i+l} + c$$

In general, the relationship between $V_i = \{v_{j_i+l} : 0 \leqslant l \leqslant k-1\}$ and $V'_i = \{v'_{j_i+l} : 0 \leqslant l \leqslant k-1\}$ may be arbitrary. This implies that the combination of V_i and V'_i may divide $[a_i, b_i]$ into at most $2k-3$ sub-intervals (noting $a_i = v_{j_i} = v'_{j_i}$ and $b_i = v_{j_i+k-1} = v'_{j_i+k-1}$). However, in each sub-interval, $e_{T,B}(a_i, x)$ is a constant, and can be determined by the values in V_i and the values in V'_i before the sub-interval. More specifically, given an x, $e_{T,B}(a_i, x)$ is equal to

$$e_{T,B}(a_i, x) = \left| \sum_{l=0}^{p} f_{j_i+l} - \sum_{l=0}^{m} f'_{j_i+l} \right| \quad (v_{j_i+p} \leqslant x < v_{i_{j_i+p+1}}, v'_{j_i+m} \leqslant x < v'_{j_i+m+1}) \tag{32}$$

Clearly, once V_i and V'_i are merged together to form a partition on $[a_i, b_i]$, we need only to run a linear scan over the partition to calculate the integration. For instance, we have a bucket which holds three value-frequency points $\{(0,4),(1,5),(4,9)\}$. In the histogram, we approximately reconstruct the bucket by $\{(0,4.75),(2,6),(4,7.25)\}$. The combination of them form three sub-intervals: $[0,1)$, $[1,2)$, and $[2,4)$. Then:

$$\int_0^4 e(0,x)\mathrm{d}x = |4.75-4| * (1-0) + |4.75-4-5| * (2-1) + |6+4.75-4-5| * (4-2) = 8.5$$

To calculate the integration, the critical part is to merge V_i and V'_i together according to their ordering. However, as V_i and V'_i is already ordered, this can be easily done by the merge paradigm in the external merge sort [27], and runs in a linear time with respect to k. The algorithm for calculating the integration consists of the following two steps.

Step 1: Sort merge V_i and V'_i together to form a disjoint partition on B_i.
Step 2: Scan the partition once to calculate the integration by taking the advantage that $e_{T,B}(a_i, x)$ is a constant in each partitioned sub-interval.

As described above, both steps run in $O(k)$ time.

4.3.2. Dynamic programming based data partition algorithm

We now present an efficient algorithm to partition a give data set $T = \{(v_1,f_1),(v_2,f_2),\ldots,(v_n,f_n)\}$ with at most β buckets such that (31) is minimized. The algorithm follows the dynamic programming technique developed in [3,20]. We briefly described it below.

Let $M^*(X, Y)$ represent the optimal result for using at most Y buckets to partition the first X values of T. Let $M[a,b]$ denote the bucket containing the consecutive $\{(v_a,f_a),\ldots,(v_b,f_b)\}$ in T. Below is the crucial formula.

$$M^*(n,\beta) = \min_{1 \leqslant j \leqslant n-1} \{M^*(j,\beta-1) + M[j+1,n]\}$$

Thus, in order to calculate $M^*(n,\beta)$, we must calculate $M^*(i,k)$ for $1 \leqslant i \leqslant n$ and $1 \leqslant k < \beta$. Applying the dynamic programming based algorithm from [20] to our problem, there will be $O(\beta n^2)$ iterations to do the computation of $M^*(i,k) + M[i+1,l]$ where $l \leqslant n$; and a computation of $M^*(i,k) + M[i+1,l]$ may be done in constant time (i.e., the computation of $M[i+1,l]$) in [20] if a linear pre-process is performed.

Note that in our problem, to make the computation of each $M*(i,k) + M[i+1,l]$ be constant time we may have to pre-compute $M[a,b]$ for every possible sub-interval in T. Further, as shown in the last Section 4.3.1, the computation of $M[a,b]$ takes linear time. Therefore, the pre-process runs in $O(n^3)$ time. This implies that the data partitioning algorithm will run in $O(n^3 + \beta n^2) = O(n^3)$ time. It occupies $O(n^2)$ space mainly due to the pre-process. On the other hand, if each $M[a,b]$ is not pre-computed and stored, then the algorithm may be implemented in $O(\beta n^3)$ with $O(\beta n)$ space requirement. So there is always a trade-off between space and efficiency.

It is worth to note that if we use the goal function in (27) to replace the goal function in (31), then the dynamic programming paradigm above also works. However, the complexities will be increased to $O(n^4)$ time with $O(n^2)$ space and $O(\beta n^4)$ with $O(\beta n)$ space, respectively. This increment is due to the calculation of each $\int_{a_i}^{b_i}\int_{x}^{b_i} e_{T,B}(x,y)\mathrm{d}y\,\mathrm{d}x$, which can be done in a similar way to the paradigm in the last subsection but runs a quadratic time.

5. Experiment results

The data sets used in our experiments are synthesized *zipf* [30] data, which is the most popular benchmark to evaluate the histogram techniques, and normal distributed data. Three databases have been generated; each database consists of 10 data sets.

- The database 1 is used to evaluate the performance of histogram techniques in the environment where all distinct values are evenly distributed over a data set with 100,000 records. Each data set in the database 1 uses the integers in [1] as the value set.
- In the database 2, a data set has all together 10,000 records with 1001 different values from the domain $[0, 10000]$.
- A data set in the database 3 contains 100,000 records with 1001 different values from the domain $[0, 100000]$.

Note that a data set from the database 2 or the database 3 has its values normally distributed. The difference between the database 2 and the database 3 is that the frequency distribution in the data sets of the database 2 is smoother than that of the database 3. The generation of each data set follows three steps below.

- *Generating frequencies:* Different frequencies are generated according to *zipf law* and the *zipf* parameter $z = 1.0$. This means a medium skew frequency distribution.
- *Generating values:* The values of database 1 follow a uniform distribution. The values of database 2 and database 3 follow the standard normal distribution.
- *Generating data distribution:* Frequencies are randomly assigned to different values.

Note that the number of buckets to be used in a histogram controls a data reduction "degree". If the number of buckets equals the number of distinct values, then there is no data reduction, and consequently all histograms will produce the same result – the exact result.

In our experiments, we compared the performance of our data partition algorithm – *MINHERR*, as presented in Section 3 for minimizing (31), with other seven algorithms.

- V-optimal (area) [26,20]: We use average area instead of average frequency as a parameter for optimization, because this gives the best performance [26] for V-optimal model.
- BSW: The linear wavelet technique [23] based on biorthogonal spline wavelet.
- LSLS (area) [21]: A linear spline of representation of a bucket, in combining with the least-square method to fix the two parameters in a line. Note that the data partitioning algorithm for LSLS follows a similar objective to that in V-optimal method; the difference is that in LSLS, line is used instead of average frequency. Further, in our experiments we also use the area as parameter for LSLS; this is because it gives a better performance [21].
- LSCS (area): The linear-spline histogram technique presented in Section 3.1. It uses the same data partitioning method and goal function as those in LSLS (area) except that the two parameters of the line are fixed by the discussions in Section 3.1.
- LSCSG (area): Another linear-spline histogram technique presented in Section 3.2. It uses the same data partitioning method and goal function as those in LSLS (area) except that the two parameters of the line are fixed by the discussions in Section 3.2.
- SAP0: The error minimization algorithm from [9].
- SAP1: Another error minimization algorithm from [9]. The difference between SAP0 and SAP1 is that SAP0 uses average frequency summation in a bucket, while SAP1 uses two lines in a bucket.
- MINHERR: Our data partition algorithm presented in this paper in combining with LSCSG line representation technique in Section 3.2.

In our experiments, we evaluated only the accuracies of these algorithms for constructing histograms for approximate aggregation. Note that these algorithms all run in a similar lower order polynomial time. In

the applications where updates are not frequent, a histogram construction algorithm does not have to run very often; and thus, the accuracy of these algorithm will be the key for approximate range queries.

We use *MATLAB's bior2.4* to construct BSW wavelet histogram. This model uses two different spline functions (Fig. 7) to do wavelet decomposition and reconstruction. In the wavelet approximation, we chose the important coefficients [23] to store.

Note that in our experiments, we have enforced that every data reduction technique occupies same storage space, to be fair. For the database 1, we do not need to store information about the number of distinct values for the histograms and need only to store the minimum value for each bucket since we know that the values are evenly distributed. Therefore, we need only record two fields for each bucket in V-optimal, while respectively in LSLS (area), LSCG (area), LSCSG (area), and MINHERR, three fields need to be recorded (see Fig. 3(a) and (b) for reference). In BSW, for each remaining coefficient we need to record their position as well; and thus two fields are needed for storing one coefficient. Consequently, the number of wavelet coefficients stored in BSW is equal to the number of buckets in V-Optimal (area), while the numbers of buckets in LSCG (area), LSCSG (area), LSLS (area) and MINHERR, respectively, are about 67% of that in V-optimal (area). Note that the numbers of buckets in SAP1 and SAP0, respectively, have been taken the same as that in V-optimal according to the suggestions from [9].

Note that for the database 2 and the database 3, it may not necessary to store the number of distinct values in a bucket if we calculate the number of records using a continuous model. However, as mentioned earlier this information is necessary for some other aggregations; for instance, the total summation of the record values in a range. Therefore, we keep this information in our histograms; see Fig. 3(a) and (b) for storage requirements. For similar reasons to those in the last paragraph, the number of wavelet coefficients, now, should be twice of the number of buckets in V-optimal (area), while the numbers of buckets in LSCG (area), LSCSG (area), LSLS (area) and MINHERR, respectively, are 80% of that in V-optimal (area). Again, the numbers of buckets in SAP1 and SAP0, respectively, are the same as that in V-optimal.

In our experiments, we use the bucket numbers 20, 25, 30, 35, 40, 45, 50 in V-optimal (area) as the reference values, while the numbers in the other algorithms are adjusted according to the above ratios.

In our experiments, we targeted the three most popular aggregates – COUNT, SUM, and AVG. Since AVG is derived from a division between SUM and COUNT, we focused only on two types of range queries, COUNT and SUM. For each data set and each type of range aggregates, 1000 queries are randomly generated with the form:

$$\{x \leqslant values \leqslant y | x < y\}$$

Here x and y are randomly selected from the value domains.

Let A_i denotes the actual result of a query q_i and A'_i denotes the approximately calculated result. The error metrics used to evaluate our histograms are:

- *absolute error:* $e_i^{\text{abs}} = |A_i - A'_i|$
- *average absolute error:* $e_N^{\text{abs}} = \frac{\sum_{i=1}^{N} e_i^{\text{abs}}}{N}$, where N represents the number of queries.
- *relative error:*

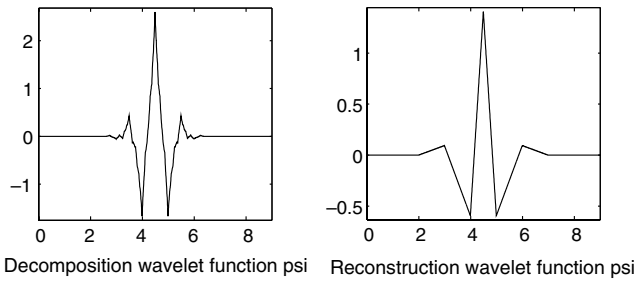


Fig. 7. Two wavelet basis functions of bior2.4.

$$e_i^{\text{rel}} = \begin{cases} \frac{|A_i - A_i'|}{A_i} & \text{if } A_i \neq 0 \\ A_i' & \text{otherwise} \end{cases}$$

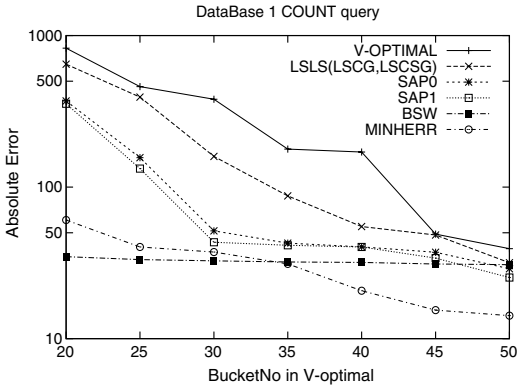
- *average relative error:* $\overline{e_N^{\text{rel}}} = \frac{\sum_{i=1}^{N} e_i^{\text{rel}}}{N}$, where N represents the number of queries.

All the generated data sets and histograms are stored in an *Oracle* DBMS and our experiments are done on a Pentium III 700 MHz CPU, 256 MB memory computer with Linux 2.4.7.

Since the results do not vary significantly on different data sets within a database, we only show some typical results here; that is, one data set per database.

Figs. 8–10 show our experiment results for the database 1. Since the value space follows the uniform-spread assumption, LSLS, LSCG, and LSCSG are equivalent according to Theorem 1. In our experiments, we focused on the average relative errors since this performance index is more explanatorily and the space is limited. However, for the database 1 we also recorded the average absolute errors for COUNT since this performance index was evaluated in [9]. It is interested to notice that unlike the other methods, adding more coefficients in BSW does not improve very much the accuracy. Further, MINHERR outperforms SAP0 and SAP1 although SAP0 and SAP1 are the two optimal algorithms to achieve the minimum average errors. This is because SAP0 and SAP1 employed different bucket representations than that in MINHERR; and the

a

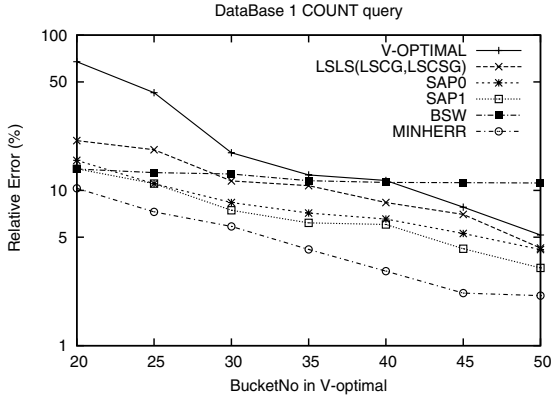


b

Bucket Number	Various Histograms' Absolute Errors of COUNT (%)					
	V-optimal	LSLS LSCG LSCSG	SAP0	SAP1	BSW	MINHEER
20	825.74	647.24	370.25	354.76	**34.76**	60.65
25	460.32	392.5	156.31	132.47	**33.25**	40.28
30	380.16	159.26	51.4	43.24	**32.59**	37.25
35	178.2	87.46	42.68	41.38	32.07	**31.16**
40	170.47	54.9	40.32	40.25	31.82	**20.7**
45	48.92	48.25	37.18	34	31.06	**15.42**
50	39.26	31.66	28.95	25.3	30.75	**14.18**

Fig. 8. Approximate COUNT query on databases 1. (a) Absolute error and (b) Table 1: Absolute error.

a

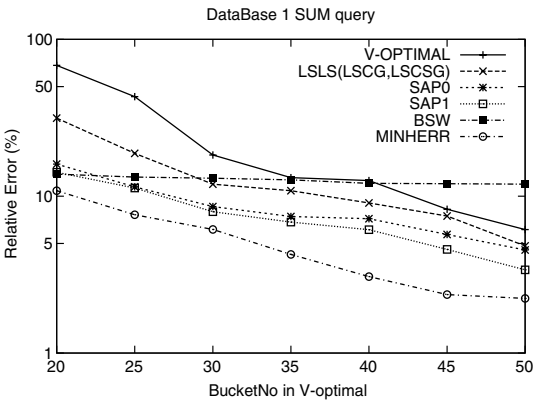


b

Bucket Number	Various Histograms' Relative Errors of COUNT (%)					
	V-optimal	LSLS LSCG LSCSG	SAP0	SAP1	BSW	MINHEER
20	67.47	20.93	15.62	13.78	13.74	**10.32**
25	42.58	18.27	11.04	11.06	13.01	**7.27**
30	17.47	11.5	8.36	7.45	12.76	**5.86**
35	12.58	10.72	7.15	6.18	11.54	**4.17**
40	11.59	8.36	6.54	6.03	11.25	**3.02**
45	7.8	7	5.28	4.21	11.21	**2.18**
50	5.16	4.25	4.16	3.17	11.16	**2.1**

Fig. 9. Approximate COUNT query on databases 1. (a) Relative error and (b) Table 2: Relative error.

a



b

Bucket Number	Various Histograms' Relative Errors of SUM (%)					
	V-optimal	LSLS LSCG LSCSG	SAP0	SAP1	BSW	MINHEER
20	68.21	31.45	16.03	14.22	13.82	**10.84**
25	43.15	18.76	11.46	11.27	13.25	**7.62**
30	18.32	11.94	8.58	7.96	13.01	**6.14**
35	13.14	10.83	7.42	6.84	12.71	**4.25**
40	12.58	9.05	7.18	6.12	12.07	**3.07**
45	8.25	7.47	5.7	4.58	12.01	**2.36**
50	6.13	4.82	4.54	3.39	11.94	**2.23**

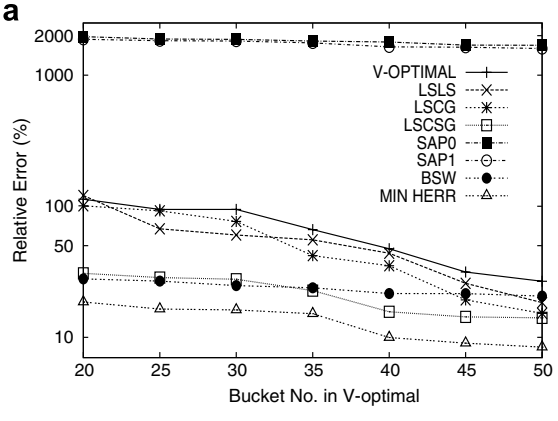
Fig. 10. Approximate SUM query on databases 1. (a) Relative error and (b) Table 3: Relative error.

algorithmic paradigms cannot be applied to our representation to provide the optimal solutions. Moreover, for the average absolute errors, BSW outperforms our MINHERR when the number of buckets is small. The MINHERR outperforms all the other techniques significantly for the average relative errors; the minimum improvement is about 50%.

Figs. 11 and 12 show the experiment results for the database 2 where Fig. 11 records the average relative error for approximate COUNT and Fig. 12 records the average relative error for approximate SUM. Figs. 13 and 14 show the experiment results for the database 3. Clearly, the experiment results for the database 2 and the database 3 suggested that the performances of the eight algorithms (including ours) follow a similar pattern to that for the database 1, except that SAP0 and SAP1 are less competitive. This is because that SAP0 and SAP1 are specially designed for an evenly distributed value space.

In summary, our experiment results clearly suggested that our algorithm MINHERR leads to the best performance among these eight algorithms, while LSCSG and LSCG representation models are significantly better than the existing histogram representation models. The minimum reduction for the average relative error, by using our MINHERR, is about 50% comparing with the other techniques; however, this excludes the combination of LSCSG combining with V-optimal for the database 3. For the database 3, MINHERR is just slightly better than LSCSG on average. This is because in the database 3, the frequency distribution varies sharply and takes the dominant role in data partitioning algorithm; and thus the advantage of MINHERR over LSCSG fades away.

a

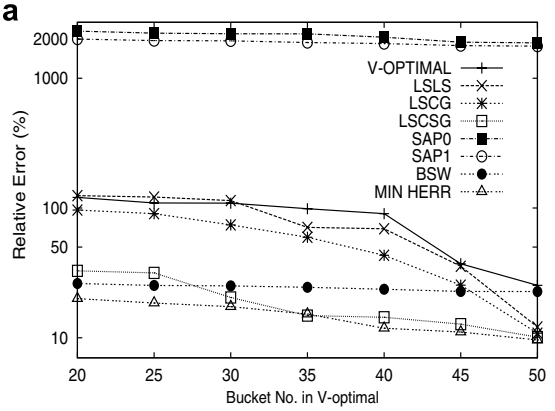


b

Bucket Number	Various Histogram's Relative Errors of COUNT (%)							
	V-optimal	LSLS	LSCG	LSCSG	SAP0	SAP1	BSW	MIN HERR
20	113.1	120.65	100.8	30.77	1965.18	1877.67	27.9	**18.61**
25	94.54	67.21	92.22	28.58	1886.18	1831.29	26.86	**16.44**
30	94.51	60.34	76.57	27.72	1874.45	1820.21	24.81	**16.18**
35	66.26	55.45	41.98	22.71	1819.26	1761.02	23.86	**15.14**
40	47.38	43.77	35.12	15.67	1788.12	1643.94	21.57	**9.97**
45	31.49	25.84	19.3	14.32	1695.83	1633.32	21.53	**9.02**
50	26.76	18.41	15.23	14.07	1688.56	1595.8	20.63	**8.42**

Fig. 11. Approximate COUNT query on databases 2. (a) Relative error and (b) Table 4: Relative error.

a

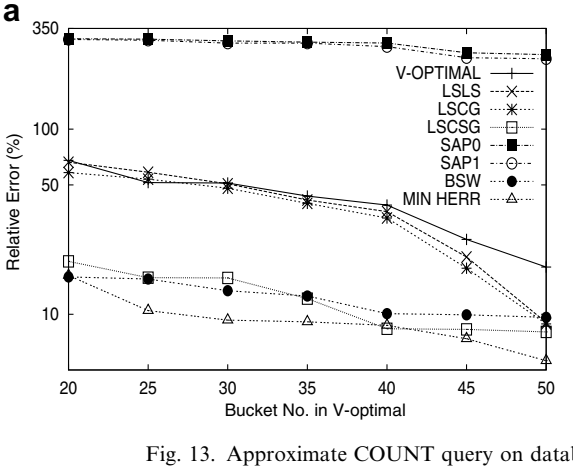


b

Bucket Number	Various Histogram's Relative Errors of SUM (%)							
	V-optimal	LSLS	LSCG	LSCSG	SAP0	SAP1	BSW	MIN HERR
20	120.74	124.5	96.42	32.77	2308.65	2004.97	26.17	**20 .04**
25	109.31	121.36	90.42	31.59	2225.13	1951.5	25.28	**18.52**
30	108.94	114.1	74.06	20.44	2201.2	1940.76	25.1	**17.36**
35	98.77	70.94	59.51	14.81	2197.71	1885.79	24.46	**15.22**
40	90.46	69.14	43.2	14.39	2071.24	1850.72	23.63	**11.84**
45	37.13	35.54	25.42	12.72	1898.17	1784.5	22.75	**11.06**
50	25.32	12.1	10.84	10.11	1873.3	1766.32	22.71	**9.61**

Fig. 12. Approximate SUM query on databases 2. (a) Relative error and (b) Table 5: Relative error.

a

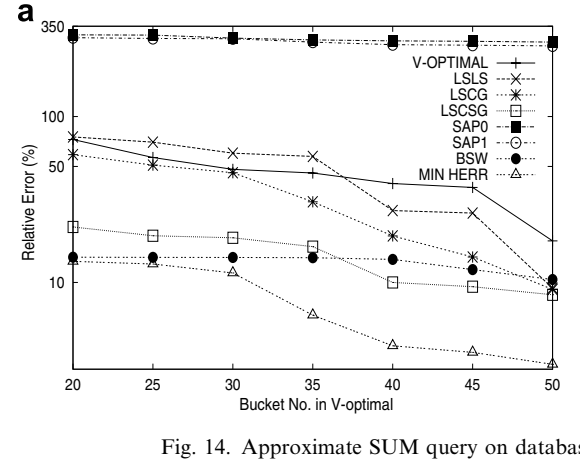


b

Bucket Number	Various Histogram's Relative Errors of COUNT (%)							
	V-optimal	LSLS	LSCG	LSCSG	SAP0	SAP1	BSW	MIN HERR
20	67.81	66.3	58.27	19.3	308.23	305.44	**15.9**	16.22
25	59.45	58.55	53.67	15.77	306.43	302.11	15.51	**10.46**
30	51.18	50.63	47.94	15.71	299.88	291.71	13.4	**9.3**
35	43.49	41.41	39.75	12.14	295.1	291.64	12.55	**9.1**
40	38.89	35.85	33.12	8.35	291.98	279.22	10.08	**8.71**
45	25.35	20.4	17.73	8.28	258.71	243.28	9.93	**7.37**
50	17.98	9.02	8.87	8.03	252.71	240.06	9.62	**5.61**

Fig. 13. Approximate COUNT query on databases 3. (a) Relative error and (b) Table 6: Relative error.

a



b

Bucket Number	Various Histogram's Relative Errors of SUM (%)							
	V-optimal	LSLS	LSCG	LSCSG	SAP0	SAP1	BSW	MIN HERR
20	72.69	75.34	58.9	21.6	310.71	298.64	14.2	**13.4**
25	56.64	70.05	50.92	19.13	309.6	295.23	14.16	**12.92**
30	48.02	60.18	45.82	18.58	296.67	294.35	14.15	**11.41**
35	45.69	57.45	30.63	16.44	289.6	280.8	14.10	**6.35**
40	39.44	27.1	19.07	10	285.74	271.08	13.77	**4.15**
45	37.38	26.18	14.23	9.43	283.95	269.03	11.95	**3.79**
50	17.8	9.23	9.07	8.43	280.72	266.54	10.41	**3.21**

Fig. 14. Approximate SUM query on databases 3. (a) Relative error and (b) Table 7: Relative error.

6. Conclusion and remarks

In this paper, we presented a new linear-spline model and a novel framework for minimizing the average approximate errors on aggregates. The optimization problem from our framework tends to be computationally intractable even restricted to a uniform query pattern. We then concentrated on a mathematic analysis to obtain an upper-bound for the average error, which is composed of the dominant parts in the average error. Consequently, we developed a near optimal efficient algorithm to solve the problem. According to our results, we will be able to choose the number of buckets to control an approximation accuracy in contrast to the existing results for general data distribution. The experiment results showed that our algorithm significantly outperforms the existing techniques.

Note that we may replace the absolute value based error function by other error functions (for instance, variance based error function) without the need of a change of techniques developed in this paper. Very recently, Guha et al. [13] extended the V-optimal techniques for minimizing the relative errors. As a future study, we will investigate the average error problem based on relative errors. We will also investigate the situation when query patterns are not necessarily uniform, as well as a data stream environment.

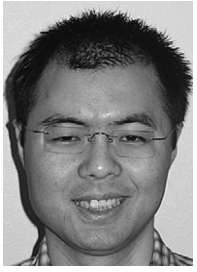
References

[1] S. Acharya, P.B. Gibbons, V. Poosala, Congressional samples for approximate answering of group-by queries, in: SIGMOD Conference, 2000, pp. 487–498.
[2] B.B. Hubbard, The world According to wavelets, A.K.Peters, 1996.
[3] R.E. Bellman, The theory of dynamic programming, Bull. Am. Math Soc. (1954).
[4] F. Buccafurri, L. Pontieri, D. Rosaci, D. Sacca, Improving range query estimation on histograms, in: ICDE 2002, 2002.
[5] A. Dobra, M. Garofalakis, J. Gehrke, R. Rastogi, Processing complex aggregate queries over data streams, in: SIGMOD 2002, 2002.
[6] T. Fukuda, Y. Morimoto, S. Morishita, T. Tokuyama, Data mining using two-dimensional optimized association rules: scheme, algorithms, and visualization, in: SIGMOD 1996, 1996.
[7] M. Garofalakis, P.B. Gibons, Approximate query processing: taming the terabytes, in: VLDB 2001 , VLDB, 2001.
[8] P.B. Gibbons, Y. Matias, New sampling-based summary statistics for improving approximate query answers, in: SIGMOD 1998, 1998, pp. 331–342.
[9] A.C. Gilbert, Y. Kotidis, S. Muthukrishnan, M. Strauss, Surfing wavelets on streams: one-pass summaries for approximate aggregate queries, The VLDB Journal (2001) 79–88.
[10] A. Graps, An introduction to wavelets, IEEE Computational Science and Engineering 2 (Summer) (1995) 50–61.
[11] M. Greenwald, S. Khanna. Space-efficient online computation of quantile summaries, in: Proceedings of the 2001 ACM SIGMOD International Conference on Management of Data, 2001, pp. 58–66.
[12] S. Guha, N. Koudas, K. Shim, Data-streams and histograms. in: ACM Symposium on Theory of Computing, 2001, pp. 471–475.
[13] S. Guha, K. Shim, J. Woo, Rehist: relative error histogram construction algorithms, in: Proceedings of 30th VLDB, 2004.
[14] J. Han, M. Kamber, Data Mining: Concepts and Techniques , Morgan Kaufman, 2001.
[15] Y. Ioannidis, Universality of serial histograms, in: Proceedings of the 19th Conference on Very Large Databases, Morgan Kaufman pubs., Los Altos CA, Dublin, 1993.
[16] Y.E. Ioannidis, S. Christodoulakis, Optimal histograms for limiting worst-case error propagation in the size of the join results, ACM Transactions on Database Systems 18 (4) (1993) 709–748.
[17] Y.E. Ioannidis, V. Poosala. Balancing histogram optimality and practicality for query result size estimation, in: SIGMOD 1995, 1995, pp. 233–244.
[18] Y.E. Ioannidis, V. Poosala, Histogram-based approximation of set-valued query-answers, The VLDB Journal (1999) 174–185.
[19] H.V. Jagadish, H. Jin, B.C. Ooi, K.-L. Tan, Global optimization of histograms, in: SIGMOD Conference, 2001.
[20] H.V. Jagadish, N. Koudas, S. Muthukrishnan, V. Poosala, K.C. Sevcik, T. Suel, Optimal histograms with quality guarantees, in: VLDB'98, Proceedings of 24rd International Conference on Very Large Data Bases, August 24–27, 1998, New York City, New York, USA, 1998.
[21] A.C. König, G. Weikum, Combining histograms and parametric curve fitting for feedback-driven query result-size estimation, in: VLDB'99, Proceedings of 25th International Conference on Very Large Data Bases, September 7–10, 1999, Edinburgh, Scotland, UK, 1999.
[22] R.P. Kooi, The Optimization of Queries in Relational Database, Ph.D. Thesis, Case Western Reserver University, 1980.
[23] Y. Matias, J.S. Vitter, M. Wang. Wavelet-based histograms for selectivity estimation, in: SIGMOD 1998, 1998, pp. 448–459.
[24] G. Piatetsky-Shapiro, C. Connell, Accurate estimation of the number of tuples satisfying a condition, in: SIGMOD'84, Proceedings of the 1984 ACM SIGMOD International Conference on Management of Data, 1984, pp. 256–276.
[25] V. Poosala, Histogram-Based Estimation Techniques in Database Systems, Ph.D. Thesis, University of Wisconsin-Madison, 1997.
[26] V. Poosala, P.J. Haas, Y.E. Ioannidis, E.J. Shekita, Improved histograms for selectivity estimation of range predicates, in: SIGMOD 1996, 1996, pp. 294–305.

[27] R. Ramakrishnan, J. Gehrke, Database Management Systems, McGraw Hill, 2000.
[28] J.S. Vitter, Random sampling with a reservoir, ACM Transactions on Mathematical Software 11 (1) (1985) 37–57.
[29] D.D. Wackerly, W. Mendenhall, R.L. Scheaffer, Mathematical Statistics with Application, Duxbury Press, 1995.
[30] G.K. Zipf, Human Behaviour and the Principle of Least Effort, Addison-Wesley, Reading, 1949.

Xuemin Lin is an Associate Professor in the School of Computer Science and Engineering, the University of New South Wales. He has been the head of database research group at UNSW since 2002. Before joining UNSW, Xuemin held various academic positions at the University of Queensland and the University of Western Australia. He got his Ph.D. in Computer Science from the University of Queensland in 1992 and his B.Sc. in Applied Math from Fudan University in 1984. During 1984–1988, he studied for Ph.D. in Applied Math at Fudan University. His current research interests lie in data streams, approximate query processing, spatial data analysis, and graph visualization.

Qing Zhang received his B.S. degree in applied physics from Tsinghua University, China, and the Ph.D. degree in computer science from the University of new south Wales, Australia, in 2005. Presently, he is a post-doctoral fellow at e-Health research centre/CSIRO ICT centre, Australia. His research interests are approximate query processing, data stream, and query processing in multidatabase.



Yidong Yuan received his B.S. and M.S. degree in Computer Science from Shanghai Jiao Tong University, PR China, in 1998 and 2001. Currently, he is a Ph.D. Candidate in the School of Computer Science and Engineering, the University of New South Wales. His research interests include efficient query processing and query optimization for spatio-temporal database and data stream systems.

Qing Liu received her Ph.D. degree in the School of Computer Science and Engineering, the University of New South Wales, Australia in 2006. Currently she works as a post-doctoral research fellow in the University of Queensland. Her research interests include approximate query processing and query optimization for spatial and spatio-temporal database systems.