Optimal and Approximate Computation of Summary Statistics for Range Aggregates

Anna C. Gilbert* Yannis Kotidis* S. Muthukrishnan* Martin J. Strauss*

Abstract

Fast estimates for aggregate queries are useful in database query optimization, approximate query answering and online query processing. Hence, there has been a lot of focus on "selectivity estimation", that is, computing summary statistics on the underlying data and using that to answer aggregate queries fast and to a reasonable approximation. We present two sets of results for range aggregate queries, which are amongst the most common queries.

First, we focus on a histogram as summary statistics and present algorithms for constructing histograms that are provably optimal (or provably approximate) for range queries; these algorithms take (pseudo-) polynomial time. These are the first known optimality or approximation results for arbitrary range queries; previously known results were optimal only for restricted range queries (such as equality queries, hierarchical or prefix range queries).

Second, we focus on wavelet-based representations as summary statistics and present fast algorithms for picking wavelet statistics that are provably optimal for range queries. No previously-known wavelet-based methods have this property.

We perform an experimental study of the various summary representations show the benefits of our algorithms over the known methods.

*AT&T Labs—Research, Florham Park, NJ. {agilbert, kotidis, muthu, mstrauss}@research.att.com.

1 Introduction

Databases have traditionally relied on selectivity estimation methods to generate fast estimates for (intermediate) result sizes. This is typically used in cost-based query optimization [13, 10, 8, 3, 4]. More recently, selectivity estimation is used where an approximate answer to a query suffices; this is appropriate where an approximation can be tolerated provided the query execution is rapid. Scenarios like that are common in the exploratory data analysis which has become a significant part of database processing with the growth in data warehousing applications. Database engines are now being designed to include approximate query processing; an example is AQUA [1]. Finally, an emerging trend in database research explores *online query processing* wherein fast estimates are provided and they get refined over time at rates controlled by the user [7]. Selectivity estimation is an inherent part of such database systems.

Here we address a basic selectivity estimation problem. We are given the attribute value distribution of a numerical attribute, that is, for each attribute[1] value i, the number of records of the database that has value i for the numerical attribute under consideration. While the joint attribute values distribution of two or more numerical attributes is also relevant, we focus only on the single attribute (i.e., the one dimensional) case in this paper.[2] Selectivity estimation proceeds by storing some summary information about the attribute value distribution which is used for providing estimates for queries [1, 11].

We consider the range-sum or simply the *range query* where the goal is to return the number of records in the database with attribute values in a given attribute range. A special case is when all ranges have coinciding endpoints, that is, each query is an *equality query*.

In this paper, we study the foundational issues in estimating

[1] Throughout, we assume the attribute value is integral; this does not affect the generality of our results since non-integral attribute can be aggregated into ranges indexed by integral values without any difficulty.

[2] Straightforward extension of our results to higher dimensions are possible, but more extensive investigation will be needed.

range queries with limited space to store summary values. Despite extensive research on selectivity estimation, there is little understanding of how to provide the "best" range query estimates for a given summary space bound. Our work is the first to address algorithmic issues behind this problem and prove guaranteed results. In particular, our technical contributions are as follows:

1. **[Histogram-Based Methods]**

   Histograms are commonly used for selectivity estimation. The goal is to build a *range-optimal* histogram which, informally, is a histogram that has the smallest total error in estimating range queries, for a given space bound for storing the histogram.

   Our main result is a pseudo-polynomial time algorithm that finds the range-optimal histogram. (The algorithm takes pseudo-polynomial time means it is polynomial in the database size and not in the number of attribute values. More formal definition is in Section 2.) No previous optimal results were known for this problem save for rather special cases equality queries [6] or hierarchically-limited range queries [9].

   We add to our main result by presenting a faster algorithm with a provable approximation guarantee that gets arbitrarily close to the range-optimal histogram, and polynomial time algorithms for *alternate* versions of the histogram that are optimal for these versions. Note that optimality of histogram depends on its representation and changing representation (e.g., the number of values stored per bucket), changes the histogram (e.g., how to estimate a range query from it) and its optimality for the given representation.

2. **[Wavelet-Based Methods]**

   Wavelet techniques work by transforming the data into a form that concentrates most of the information into a small number of coefficients that can be easily summarized. We present a fast (in fact, near-linear time) algorithm to construct a provably optimal wavelet summary for range queries; no such algorithm was previously known.

3. **[Experimental Results]**

   We perform an experimental study of the suggested histogram-based algorithms, wavelet methods as well as some heuristic variations thereof. Here we use the pseudopolynomial time optimal histogram algorithm as a benchmark to compare the performance of the other algorithms.

   We show not only that selectivity estimation methods that do not optimize for range-queries perform significantly poorly compared to the ones that do, but also that our heuristics that are no more expensive to compute than the ones known in literature (for example, those that optimize equality queries) perform very well for range queries.

Map. We discuss the histogram definition and our algorithmic results for range-optimal histogram construction in Section 2. We discuss the wavelet approach briefly in Section 3. Further heuristics and experimental results can be found in Section 4. Concluding remarks in Section 5 discuss other results and future directions. Due to space constraints, we have only stated or sketched the results; proofs will be found in the final version of this paper.

2 Histogram approach

In this Section, we consider histogram methods. That is, we partition the indices $1, \ldots, n$ of a given array A into B contiguous subsequences, (called buckets), store $O(1)$ summary statistics for each bucket, and answer queries based on the summary statistics. In each of the methods below, we will optimize the selection of bucket boundaries and fix the procedure used to answer queries (described further below). Each such representation will result in possibly different "optimal" histogram for given datasets, and we will present algorithms for computing such optimal histograms or approximations thereof. In the experimental section, we will compare the effectiveness of the different representations and our algorithms.

A brief technical overview of our result follows. Histogram construction algorithms mostly rely on dynamic programming wherein an optimal result for a (sub)problem is obtained by dividing it into two or more subproblems and combining the optimal solutions to those subproblems. For this strategy to work, the constituent subproblems should have no interaction so that it suffices to consider their optimal solutions independently. This framework applies for equality queries and the resulting dynamic programming algorithm is in [6]. This does not, however, work for range queries where the ranges induce long range dependence between the subproblems.[3] Our technical results in this category consist of many observations that lets us precisely compute the long range interaction within the solution of various subproblems, or approximate this long range dependence, or show certain histogram representations wherein the long range dependence can be made to disappear. These observations give us the variety of algorithms we describe in this section.

[3]This dependence persists even if one thinks of a range query on the original distribution as a collection of two equality queries on its prefix sums distribution comprising the two endpoints of the range.

We first consider the OPT-A histogram, in which there is a single summary statistic for each bucket, namely, the average value of $A[i]$'s in that bucket. This is the classical histogram considered in the literature.

In our histogram discussion, we will use the following notation. Define $s[a,b] = \sum_{a \le i \le b} A[i]$; all ranges below include both endpoints. We will denote by $\overline{s[a,b]}$ the approximation to $s[a,b]$ given by whatever method is under consideration, so the goal is to minimize the sum-squared error (summed over all possible range queries)[4]

$$SSE = \sum_{a \le b} \left(s[a,b] - \overline{s[a,b]} \right)^2 .$$

For an index $a = 1, \ldots, n$, let $B_a^<$ and $B_a^>$ denote the leftmost and rightmost elements in the bucket to which a belongs. We also write $B_i^<$ and $B_i^>$ for the left- and rightmost elements of the i'th bucket, $i = 1, \ldots, B$; no confusion should ensue. Let $\mathrm{buck}(a)$ denote the bucket index $1, \ldots, B$ of the bucket containing $a = 1, \ldots, n$, and let $\mathrm{avg}(i)$ denote the average of the i'th bucket.

2.1 OPT-A and Pseudopolynomial Time Algorithms

The OPT-A approximation is given by equation 1 in the box. The square brackets indicate rounding their argument to a nearby integer in an arbitrary way. The argument to the square brackets represents the outcome of breaking the query (a,b) into three pieces—the piece contained in $\mathrm{buck}(a)$, the piece contained in $\mathrm{buck}(b)$, and the piece that spans all buckets (if any) strictly between $\mathrm{buck}(a)$ and $\mathrm{buck}(b)$. Note that, since

$$(B_i^> - B_i^< + 1)\mathrm{avg}(i) = \sum_{B_i^< \le j \le B_i^>} A[j],$$

the approximation

$$\overline{s[B_a^> + 1, B_b^< - 1]}$$

of the middle piece is exact—the error in $s[a,b]$ is due to the end pieces only. Similarly, given a query (a,b) with $\mathrm{buck}(a) = \mathrm{buck}(b)$, answer with

$$\overline{s[a,b]} = [(b - a + 1)\mathrm{avg}(\mathrm{buck}(a))] .$$

[4]One may more generally consider workload distribution in which there is a probability associated with each range query. We will address the problem of extending our solution to such general workload case in the final version of this paper.

2.1.1 A Warm-up Exact Algorithm for OPT-A

We now give and analyze a pseudopolynomial time algorithm for OPT-A. In the next Section, we will give a more efficient algorithm that builds on the algorithm of this Section.

Define $\delta_{ab} = s[a,b] - \overline{s[a,b]}$. Next, we will define

$$E(i, k, \Lambda_2, \Lambda)$$

for any given histogram solution with at most k buckets for $[1,i]$ such that $\sum_{l \le i} \delta_{lB_l^>} = \Lambda$ and $\sum_{l \le i} \delta_{lB_l^>}^2 = \Lambda_2$. In these cases, define $E(i,k,\Lambda_2,\Lambda)$ by $\sum_{[l,r] \subseteq [1,i]} \delta_{lr}^2$. Here we stress that $\overline{s[a,b]}$, δ_{ab}, and $E(i,k,\Lambda_2,\Lambda)$ all depend on a (partial) bucketing of the array A. The implied bucketing will be clear from context.

For example, suppose $n = 6$, the array A is given by

$$\langle 1, 3, 5, 11, 12, 13 \rangle,$$

$i = 4$, and $k = 2$. Consider the bucketing of the first $i = 4$ values into two equal buckets, $\langle 1, 3 \rangle$ and $\langle 5, 11 \rangle$. The averages of the buckets are 2 and 8. Then $\sum_{l \le i} \delta_{lB_l^>} = (1+3) - 2 \cdot 2 + 3 - 2 + (5+11) - 2 \cdot 8 + 11 - 8 = 4$ and $\sum_{l \le i} \delta_{lB_l^>}^2 = (1+3-2\cdot 2)^2 + (3-2)^2 + (5+11-2\cdot 8)^2 + (11-8)^2 = 10$. Thus $E(4,2,4,10)$ is defined for this histogram and its value is

$$\begin{aligned} & E(4,2,4,10) \\ &= (1-2)^2 + (1+3-2\cdot 2)^2 + (1+3+5-2\cdot 2-8)^2 \\ &\quad +(1+3+5+11-2\cdot 2-2\cdot 8)^2 \\ &\quad +(3-2)^2 + (3+5-2-8)^2 + (3+5+11-2-2\cdot 8)^2 \\ &\quad +(5-8)^2 + (5+11-2\cdot 8)^2 \\ &\quad +(11-8)^2 \\ &= 36. \end{aligned}$$

Define $E^*(i,k,\Lambda_2,\Lambda)$ to be minimum value of

$$E(i, k, \Lambda_2, \Lambda)$$

for any histogram that satisfies the conditions above. In order to solve our problem, clearly it suffices to compute

$$E^*(n, B, \Lambda_2, \Lambda)$$

for all possible values of Λ and Λ_2 and to take the minimum possible. Our strategy is to compute the best bucketization of size k by trying all possible values j for the right boundary of the leftmost $k-1$ buckets, and recursively solve the $(k-1)$-bucket problem. We have

$$E^*(i,k,\Lambda_2,\Lambda) = \min_{\substack{j,\lambda,\lambda_2 \\ j<i \\ \lambda + \sum_{j+1 \le l \le i} \delta_{li} = \Lambda \\ \lambda_2 + \sum_{j+1 \le l \le i} \delta_{li}^2 = \Lambda_2}} E^*(j, k-1, \lambda_2, \lambda) +$$

$$\overline{s[a,b]} = \left[(B_a^> - a + 1)\mathrm{avg}(\mathrm{buck}(a)) + \left(\sum_{\mathrm{buck}(a)<i<\mathrm{buck}(b)} (B_i^> - B_i^< + 1)\mathrm{avg}(i) \right) + (b - B_b^< + 1)\mathrm{avg}(\mathrm{buck}(b)) \right]. \quad (1)$$

$$\sum_{[l,r]\subseteq[j+1,i]} \delta_{lr}^2 + \sum_{\substack{l,r \\ 1\le l\le j \\ j+1\le r\le i}} \delta_{lr}^2.$$

Expanding the third term on the right, we have

$$\begin{aligned}
&\sum_{1\le l\le j<r\le i} \delta_{lr}^2 \\
&= \sum_{1\le l\le j<r\le i} \Big(s[l, B_l^>] + \cdots + s[j+1, r] \\
&\qquad - \overline{s[l, B_l^>]} - \cdots - \overline{s[j+1,r]} \Big)^2 \\
&= \sum_{1\le l\le j<r\le i} (\delta_{lB_l^>} + \delta_{(j+1)r})^2 \\
&= \left(\sum_{l\le j} \delta_{lB_l^>}^2 \right) (i-j) + \sum_{j+1\le r\le i} \delta_{(j+1)r}^2 (j) \\
&\quad +2 \left(\sum_{j+1\le r\le i} \delta_{(j+1)r} \right) \left(\sum_{l\le j} \delta_{lB_l^>} \right) \\
&= \lambda_2 (i-j) + \sum_{j+1\le r\le i} \delta_{(j+1)r}^2 (j) + 2 \sum_{j+1\le r\le i} \delta_{(j+1)r} \lambda
\end{aligned}$$

Hence,

$$\begin{aligned}
&E^*(i, k, \Lambda_2, \Lambda) \\
&= \min_{\substack{j,\lambda,\lambda_2 \\ j<i \\ \lambda + \sum_{j+1\le l\le i} \delta_{li} = \Lambda \\ \lambda_2 + \sum_{j+1\le l\le i} \delta_{li}^2 = \Lambda_2}} E^*(j, k-1, \lambda_2, \lambda) \\
&\quad + \sum_{[l,r]\in[j+1,i]} \delta_{lr}^2 + (i-j)\lambda_2 + j \sum_{j+1\le r\le i} \delta_{(j+1)r}^2 \\
&\quad +2\lambda \sum_{j+1\le r\le i} \delta_{(j+1)r}.
\end{aligned}$$

Theorem 1 *There exists a pseudopolynomial time algorithm for constructing the OPT-A histogram with optimal bucket boundaries.*

Proof. The algorithm involves computing $E^*(n, B, \Lambda_2, \Lambda)$, for all possible values of Λ and Λ_2, using the recurrence above. Let Λ^* be the largest value of $|\Lambda|$ such that Λ needs to be explored; we now bound Λ^* from above. First, we show that $|\delta_{lB_l^>}| \le s[1..n]$. We have assumed that all values of A are non-negative whence it follows that the average in each bucket is non-negative, so (using crude bounds)

$$\begin{aligned}
\delta_{lB_l^>} &= \sum_{l\le i\le B_l^>} (A[i] - \mathrm{avg}(\mathrm{buck}(l))) \\
|\delta_{lB_l^>}| &\le \sum_{1\le i\le n} (A[i] + \mathrm{avg}(\mathrm{buck}(l))) \\
&\le 2s[1,n].
\end{aligned}$$

Thus

$$|\Lambda| = \left| \sum_l \delta_{lB_l^>} \right| \le \sum_l |\delta_{lB_l^>}| \le ns[1,n].$$

Similarly, an upper bound, Λ_2^*, for Λ_2 is given by $\Lambda_2 \le \sum_l \delta_{lB_l^>}^2 \le n(s[1,n])^2$. Furthermore, Λ and Λ_2 are always integral. Hence, the number of different E^* values we will compute is

$$O(nB(ns[1,n])n(s[1,n]])^2) = O(n^3 B(s[1,n])^3).$$

Each computation involves checking at most n choices for j, and each such choice can be evaluated in $O(1)$ time after a $O(n)$ preprocessing. (This step is nontrivial, but uses standard techniques as can be found in e.g., [9].) Hence, the entire algorithm takes time $O(n^4 Bs[1,n]^3)$. This is polynomial in $s[1,n]$ and hence, the entire algorithm is pseudopolynomial. We note that Λ^* and Λ_2^* are likely to be much less than the upper bound above we have used to prove that the algorithm works in time pseudopolynomial in the input size. For example, one can show that each of $|\Lambda^*|$ and $|\Lambda_2^*|$ is at most OPT, where OPT is the optimal error. ∎

2.1.2 Improved Exact Algorithm

In this section, we present a faster pseudopolynomial time algorithm for constructing OPT-A histograms.

We define $F(i, k, \Lambda)$ for any given histogram solution with at most k buckets for $[1, i]$ such that $\sum_{l\le i} \delta_{lB_l^>} = \Lambda$. The quantity $F(i, k, \Lambda)$ is defined to be

$$F(i, k, \Lambda) = \sum_{[l,r]\subseteq[1,i]} \delta_{lr}^2 + \sum_{l\le i} \delta_{lB_l^>}^2 (n-i)$$

Define $F^*(i,k,\Lambda)$ to be the minimum value of $F(i,k,\Lambda)$ for any histogram that satisfies the conditions above. As in the previous Section, we can try all possible values $j+1$ for the left endpoint of the rightmost bucket. That is,

$$F^*(i,k,\Lambda) = \min_{\substack{j<i \\ \lambda+\sum_{j+1\le l\le i}\delta_{li}=\Lambda}} F^*(j,k-1,\lambda) - \sum_{l\le j}\delta^2_{lB_l^>}(n-j)$$
$$+\sum_{l\le i}\delta^2_{lB_l^>}(n-i) + \sum_{[l,r]\subseteq[j+1,i]}\delta^2_{lr} + \sum_{1\le l\le j<r\le i}\delta^2_{lr}.$$

We expand the last term:

$$\sum_{1\le l\le j<r\le i}\delta^2_{lr} = \sum_{j+1\le r\le i}\delta^2_{(j+1)r}(j) + 2\sum_{j+1\le r\le i}\delta_{(j+1)r}\lambda$$
$$+\sum_{l\le i}\delta^2_{lB_l^>}(n-i),$$

and get

$$F^*(i,k,\Lambda) = \min_{\substack{j<i \\ \lambda+\sum_{j+1\le l\le i}\delta_{li}=\Lambda}} F^*(j,k-1,\lambda)+$$
$$+\sum_{[l,r]\subseteq[j+1,i]}\delta^2_{lr} + \sum_{j+1\le r\le i}\delta^2_{(j+1)r}(j)+$$
$$2\sum_{j+1\le r\le i}\delta_{(j+1)r}\lambda + \sum_{j+1\le l\le i}\delta^2_{li}(n-i).$$

Clearly, it suffices to compute $F^*(n,B,\Lambda)$ for all possible Λ values in order to compute the range-optimal histogram. As before, we bound Λ by Λ^*, and, since Λ values are integral, we need only consider Λ^* values of Λ.

Theorem 2 *There exists an $O(n^2B\Lambda^*)$ time algorithm to compute the OPT-A histogram with optimal bucket boundaries.*

Note that $\Lambda^* = O(\min\{OPT, ns[1,n]\})$, where OPT is the optimal error of the range-optimal histogram.

2.1.3 OPT-A-ROUNDED—A Faster Approximate Method

In this Section, we consider maintaining intermediate results only to a nearby multiple of an integer, x, to be determined later. The result is that the runtime improves by a factor of x (since we need only consider Λ values that are multiples of x), while the histogram quality degrades by a bounded amount.

Specifically, define the OPT-A-ROUNDED histograms (with parameter x, to be optimized later) as follows:

Definition 3 *Given an array A, round the entries (up or down, arbitrarily) to nearby multiples of x. Divide the result through by x. Compute the OPT-A histograms on the result, and multiply through by x.*

Theorem 4 *Fix any $\epsilon > 0$.*

1. *The histogram OPT-A-ROUNDED with parameter ϵ gives error within the factor $(1+\epsilon)$ of the error of OPT-A.*
2. *Our algorithms for OPT-A and OPT-A-ROUNDED require the same storage, $2B$ values per bucket.*
3. *There is an algorithm for OPT-A-ROUNDED that requires time $O(n^8B/\epsilon^2 + n^5B\sqrt{\Lambda^*}/\epsilon)$.*

The dominant expression in the runtime is $\sqrt{\Lambda^*}$. This should be compared with Λ^* in the runtime of our best exact OPT-A algorithm.

Some additional savings is possible by using unbiased randomized rounding instead of arbitrary rounding. In particular, we can prove the theorem above with runtime
$O(n^4B/\epsilon + n^4B\sqrt{\Lambda^*/\epsilon})$.

2.2 Histogram Variants and Polynomial Time Methods

In this section, we define *variants* of the classical histogram where we store $O(1)$ summary values for each bucket (recall that in the classical histogram, we just store the average value per bucket). Informally, the main theme in this section is that we choose appropriate summary value representations for the buckets such that the cross-terms in the earlier sections (that is ones that depend on Λ and/or Λ_2) will vanish. As a result, the algorithms are considerably faster (in fact, they run in time polynomial in n and B) and produce optimal histogram variants. Later, in the experimental section, we compare how these histogram variants compare with the classical ones discussed in the previous section. For now, we focus on computing the histogram variants optimally and efficiently.

The results of this Section follow from a main Lemma that says, intuitively, that, for a natural answering procedure that we specify below, the cross terms in the sum-squared error vanish. We first describe one variant of this algorithm, SAP0, then indicate a modest generalization to SAP1.

2.2.1 The SAP0 Histogram

The SAP0 and SAP1 methods are histogram approximations. Given a histogram approximation H for range queries to an array A, recall that $s[l,r]$ denotes $\sum_{l\le i\le r} A[i]$, $\overline{s[l,r]}$ denotes the approximation to $s[l,r]$, and δ_{lr} denotes $s[l,r]-\overline{s[l,r]}$. Associated with each bucket i, in SAP0 there are three values: an suffix value $\mathrm{suff}(i)$, the average value $\mathrm{avg}(i)$, and a prefix value $\mathrm{pref}(i)$. To answer an inter-bucket query (a,b), return

$$\begin{aligned}
&\overline{s[a,b]}\\
&\quad= \mathrm{suff}(\mathrm{buck}(a))\\
&\qquad+\left(\sum_{\mathrm{buck}(a)<i<\mathrm{buck}(b)} (B_i^> - B_i^< + 1)\mathrm{avg}(i)\right)\\
&\qquad+\mathrm{pref}(\mathrm{buck}(b)).
\end{aligned}$$

Note that the response to an inter-bucket query depends only on $\mathrm{buck}(a)$ and $\mathrm{buck}(b)$, but not otherwise on a and b. An intra-bucket query (a,b) is answered by $\mathrm{avg}(\mathrm{buck}(a))(b-a+1)$. Also note that, in contrast with OPT-A, the above value is not necessarily an integer.

The answering procedure specifies that the stored average value is the actual bucket average, but, *a priori*, the suffix and prefix values can be any values.

Each of the optimal values has a closed-form expression—they will be the averages of the bucket suffix sums and bucket prefix sums, respectively. Furthermore, the special properties of the suffix and prefix averages make the final, cross terms in

$$\delta_{lr}^2 = \delta_{lB_l^>}^2 + \delta_{rB_r^<}^2 + 2\delta_{lB_l^>}\delta_{rB_r^<}$$

vanish, leaving only error contributions computable from local information. It follows that one can compute an SAP0 histogram in polynomial time, using the dynamic programming technique from [6] and from earlier in this Section. We defer details until after presenting the following Lemma, which is needed to show correctness and to motivate the algorithm.

Lemma 5 (Decomposition Lemma)

1. *For each r in the rightmost bucket of a partial bucketing, with prefix summary values equal to average bucket prefix sums and suffix summary values equal to average bucket suffix sums,*

$$\sum_{l<B_r^<} \delta_{lr}^2 = \left(\sum_{l<B_r^<} \delta_{lB_l^>}^2\right) + \delta_{B_r^< r}^2.$$

2. *For this fixed query answering procedure and any partition, the suffix values and prefix values that optimize $\sum_{l,r}\delta_{lr}^2$ are the averages of bucket suffix sums and bucket prefix sums,*

$$\mathrm{suff}(\mathrm{buck}(i)) = \frac{1}{B_i^> - B_i^< + 1}\sum_{B_i^<\le j\le B_i^>} s[j,B_j^>]$$

and

$$\mathrm{pref}(\mathrm{buck}(i)) = \frac{1}{B_i^> - B_i^< + 1}\sum_{B_i^<\le j\le B_i^>} s[B_j^<,j].$$

Proof: First, part 1. By definition

$$\begin{aligned}
\delta_{lr}^2 \;=\;& \Big((s[l,B_l^>] + \cdots + s[B_r^<,r])\\
&\quad - \left(\overline{s[l,B_l^>]} + \cdots + \overline{s[B_r^<,r]}\right)\Big)^2.
\end{aligned}$$

Note, however, that, for any interval $[B_i^<, B_i^>]$, we have

$$\overline{s[B_i^<,B_i^>]} = s[B_i^<,B_i^>],$$

since $(B_i^> - B_i^< + 1)\mathrm{avg}(i) = \sum_{\mathrm{buck}(j)=i} A[j]$. Hence, for each $l < B_r^<$,

$$\delta_{lr}^2 = \left((s[l,B_l^>] + s[B_r^<,r]) - \left(\overline{s[l,B_l^>]} + \overline{s[B_r^<,r]}\right)\right)^2.$$

That is,

$$\delta_{lr}^2 = \delta_{lB_l^>}^2 + \delta_{B_r^< r}^2 - 2\delta_{lB_l^>}\delta_{B_r^< r}. \tag{2}$$

By our choice of approximation, $\mathrm{suff}()$, of each bucket's suffix query, it follows that, for each bucket $B < \mathrm{buck}(r)$, we have $\sum_{l\in B}\delta_{lB_l^>} = 0$, whence, considering all l in buckets left of r, we have $\sum_{l<B_r^<}\delta_{lB_l^>} = 0$. This, with (2), gives part 1 of the Lemma.

As for part 2, consider queries (l,r) with l in the leftmost bucket and r in an arbitrary other bucket. Note, that, for this answering procedure, we could add a constant, c, to each suffix approximation and subtract c from each prefix approximation. This yields an equivalent approximation in the sense that it gives the same responses to queries; we consider only equivalence classes of approximations. Thus, we may assume, without loss of generality, that the optimal suffix value in the leftmost bucket is the average of bucket suffix sums. We need only show that the average of prefix sums is the optimal prefix value.

But, for any random variable X, the expected value $E[(X - b)^2]$ is minimized for $b = E[X]$. The optimality of prefix values follows. Similarly, if the rightmost prefix value is the bucket prefix average, then all the optimal suffix values are bucket suffix averages. The Lemma follows. ∎

Intuitively, part 1 of the Lemma says that the total inter-bucket error due to all range queries with right end point r is the sum of two contributions, one that depends only on local information between $B_r^<$ and r and the other that depends only on local information between l and $B_l^<$ (independent of r). In other words, the contribution of a bucket B to the total error due to all inter-bucket ranges that have left endpoints in B is independent of the buckets into which the ranges' right endpoints fall. This greatly helps in finding the optimal histogram as we will see next.

First, note that the Decomposition Lemma implies that if we find the optimal bucket boundaries under the specified prefix and suffix values and specified answering procedure, we will in fact have the optimum over all bucket boundaries and *all* prefix and suffix values.[5] To see this (despite the apparent circularity of the Decomposition Lemma), consider the optimum histogram H over all bucket boundaries and summary values. By part 2, the summary values are the average prefix and suffix sums. Now, consider finding the histogram H' with the best bucket boundaries for the answering procedure that specifies using average prefix and suffix sums; H' can be found quickly by dynamic programming, since, by part 1, the cross-terms of the sum-squared error vanishes. By optimality of H' and H (over different sets of histograms), $H = H'$.

We now proceed to the top-level algorithm to construct histograms. We use dynamic programming and just focus on determining the error of the optimal histogram. Define the quantity $E(i, k)$ to be minimum error of a histogram of the prefix $A[1, i]$ using at most k buckets in which we consider all range queries totally contained in $[1, i]$ as well as the total left contribution of these buckets to all the ranges whose left endpoint is in $[1, i]$ and whose right endpoints is in $[i + 1, n]$.

[5]Recall that the fixed answering procedure specifies using the average values for buckets in the interior of ranges. Although this results in zero error contribution due to buckets in the interior of queries, using the bucket averages are not necessarily optimum for lowering the overall sum-squared-error.

We then have the simple recurrence in which to compute $E(i, k)$, we consider all possible choices of $j < i$ that use $k-1$ buckets for $A[1, j]$ and have $[j+1, i]$ as the k'th bucket. The error contributed by the bucket $[j + 1, i]$ can be determined in $O(1)$ time as claimed earlier, and the rest follows from dynamic programming:

Theorem 6 *The SAP0 histogram can be computed in time $O(n^2 B)$.*

One can save space by recovering the bucket averages from the bucket prefixes and suffixes. Thus bucket boundaries, prefixes, and suffixes need to be stored, plausibly requiring one computer word each. We then have:

Theorem 7 *The SAP0 histogram requires storage for $3B$ numbers.*

2.2.2 SAP1—Higher-Order Approximations

Recall that the SAP0 algorithm of the previous Section kept constants $\mathrm{suff}()$ and $\mathrm{pref}()$ and approximated $s[l, B_l^>]$ by $\mathrm{suff}(\mathrm{buck}(l))$ and $s[B_r^<, r]$ by $\mathrm{pref}(\mathrm{buck}(r))$. The approximation is not sensitive to l or r given $\mathrm{buck}(l)$ and $\mathrm{buck}(r)$. More generally, we can also store other values, $\mathrm{suff}'(i)$ and $\mathrm{pref}'(i)$, and approximate $s[l, B_l^>]$ by

$$(B_l^> - l + 1)\mathrm{suff}'(\mathrm{buck}(l)) + \mathrm{suff}(\mathrm{buck}(l))$$

and approximate $s[B_r^<, r]$ by

$$(r - B_r^< + 1)\mathrm{pref}'(\mathrm{buck}(r)) + \mathrm{pref}(\mathrm{buck}(r)).$$

We note without proof that the techniques of the previous Section all work for this answering procedure. The optimal values of $\mathrm{suff}'(i)$ and $\mathrm{suff}(i)$ are the coefficients of the best vertical-offset sum-squared-error linear regression fit to the set $\{(l, s[l, B_l^>]) : \mathrm{buck}(l) = i\}$. As in the case of the SAP0 histogram, we can simultaneously optimize the bucket boundaries and the $\mathrm{pref}, \mathrm{pref}', \mathrm{suff}$, and suff' summary values. Also, as in the case of the SAP0 algorithm, it is not necessary to store the bucket averages, since these can be recovered from the $\mathrm{suff}()$ and $\mathrm{pref}()$ values (which are the same as in the SAP0 algorithm).

Theorem 8 *The range-optimal SAP1 histogram with optimal bucket boundaries and summary statistics can be computed in time $O(n^2 B)$. The SAP1 histogram requires storage for $5B$ numbers.*

Observe that, functionally, OPT-A stores the average values per bucket, use them also for the suff' and pref' values, and sets $\mathrm{suff}()$ and $\mathrm{pref}()$ to zero. Since SAP1 optimizes the four suffix and prefix values, it produces a B-bucket histogram with error no worse (and, generally, better) than B-bucket OPT-A histograms, while using 2.5 times as much space per bucket. The SAP0 B-bucket histograms are provably incomparable with the OPT-A B-bucket histograms, and the SAP0 B-bucket histograms use 50% more space. We compare various histograms experimentally in Section 4.

3 Wavelet-based Representations

Histograms are one common flexible method for calculating summary statistics; wavelet-based techniques are another. Several recent works (see [11, 17], for example) focus on wavelet-based summary statistics for database query optimization, approximate query answering, and dynamic maintenance of such statistics. This work uses only one orthonormal basis (the Haar basis) and gives a variety of heuristic estimation methods for point and range queries with respect to this fixed basis. We will also use the Haar basis in this paper. See [5] and the references therein for more information on wavelets.

We show how to compute the wavelet coefficients that are optimal for rangesum queries. Our overall approach is general. For a given array A, we consider array AA where $AA[i, j]$ is the sum of $A[i]$ to $A[j]$, that is, the rangesum. Array AA is "virtual" in that we never materialize it. Now we compute two dimensional point wise optimal wavelets on AA. Standard two dimensional wavelet computation on a two dimensional $N \times N$ array would take $\Omega(N^2)$ time, but using the special structure in AA (it does not have $O(N^2)$ independent entries, but rather only $O(N)$ ones), we are able to show the following:

Theorem 9 *There is an algorithm to compute the optimal B-coefficient wavelet representation, for range queries in an array of length N, that runs in time $\tilde{O}(N(B \log N)^{O(1)})$.*

We have omitted several details involved in obtaining the result above due to space constraints. Although the overall approach we outlined above works for histogram based approximations as well, the key here is that we are able to get a near linear (in N) time algorithm for the wavelet representation by using the special structure in them, which is significantly faster than the ones we know for histogram representations.

4 Experiments and benchmarks

In this section we perform an experimental study of the various histogram and wavelet representations discussed in previous Sections, as well as heuristics and local search improvements to them. We used a dataset containing 127 integer keys created after doing random rounding, (up or down with probability 1/2) of floats that are Zipf distributed [18] with tail exponent $\alpha = 1.8$.

Due to space constraints, we decided to focus only representation "quality" results, that is, show the error in different representations, not the runtimes. Although our wavelet algorithms are quicker than methods for histograms, our preliminary experiments with wavelet-based representations yield results that are qualitatively worse than histogram-methods. We include one wavelet-based experiment, denoted TOPBB in Figure 1, but don't discuss this further.

In Figure 1, we plot the sum-squared error (SSE) for all histogram and wavelet approximations that we tested. Notice that y-axis is logarithmic. NAIVE is a simple summary representation that uses the average value of all $A[i]$'s to answer the queries. It is included only to provide a reasonable upper bound for SSE. The x-axis denotes the storage requirements of each representation, assuming bucket boundaries and summary values each require a single computer word.

The A0 histogram is a variation of the SAP0 histogram, in which only the average value $\mathrm{avg}(i)$ is stored in bucket i, allowing more buckets for a given target space. Query (a, b) with $\mathrm{buck}(a) \neq \mathrm{buck}(b)$ is answered using (1). We employ the same dynamic programming set-up that we used for computing SAP0, but the error, δ_{rl}, introduced for a range query is given by (2). For this answering procedure the third, cross term in (2) does not vanish, so the resulting histogram, constructed by a dynamic program that ignores the cross term, is not optimal. Since this histogram is a variant of SAP0, in which only the average value of the bucket is considered, we refer to it as A0.

Theorem 10 *The A0 histogram requires storage for $2B$ numbers.*

Based on the reported results we conclude the following for the histogram techniques:

- Histograms that are optimized for point queries are inadequate for answering range-sum queries. We implemented the V-Optimal histogram [6] optimized to answer point queries on $A[i]$ using an $O(n^2B)$ dynamic

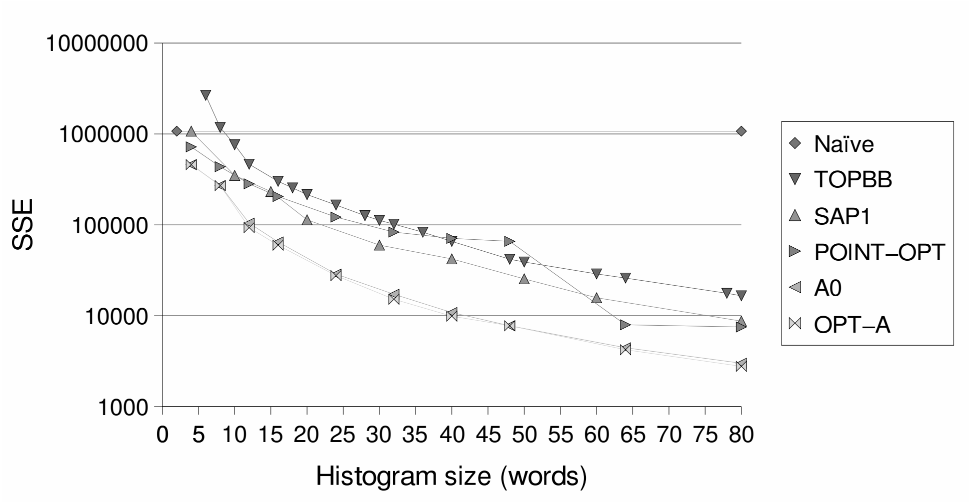

Figure 1: Measurements for Zipf distribution

programming algorithm. We adjusted the probabilities for each point $A[i]$ to reflect the probability that $A[i]$ is part of a random range-query. Performance of the V-Optimal histogram is denoted as POINT-OPT in all graphs. For our datasets the point optimal histogram is up to 8 times worst than OPT-A with respect to SSE and, on average, OPT-A is more than three times better. POINT-OPT is inferior to all histograms for range queries that we present in the graph.

- OPT-A requires pseudopolynomial time construction, which will be infeasible for realistic datasets. SAP0 and SAP1 provide two polynomial time alternatives. Construction of both histograms is faster because of the decomposition lemma that allows strictly polynomial running-time, but this comes at the expense of using more values per bucket. The SAP0 approximation is insensitive to the left or right query point in a bucket and was inferior (in terms of SSE per unit storage) to all other histograms that we tested.[6] SAP1 is provably better than OPT-A for the same number of buckets, however is requires 2.5 times more space. In our tests OPT-A is 2-4 times better than SAP1, with respect to SSE for a given space bound. This shows that using more buckets is better that incorporating more complex statistics within each bucket (and therefore realizing fewer of them).

[6]For readability we did not include measurements for SAP0 in the Figure.

5 Conclusions

We have studied the selectivity estimation problem for range queries, a fundamental problem. We have present the first-known optimal and approximate algorithms with provable guarantees for constructing histogram-based summary values; the algorithms take (pseudo) polynomial time. We have also considered the wavelet-based approach for constructing summary values and presented a near-linear time algorithm for optimally choosing the summary values for range queries. Again, no such results were known previously. We have also presented experimental results with our algorithms and other heuristics. Our work lays the foundation for understanding the true complexity of constructing "optimal" summary representations for range queries. Most results herein are only sketched, full proofs will be in final version.

We describe one final idea, namely, a general approach to improve the quality of histogram representations.

Once we have determined the bucket boundaries for any histogram, we can fix them and further optimize the approximation by changing the values stored in each bucket. For a fixed bucketing scheme, we substitute $\mathrm{avg}(i)$ in formula 1 with a value $x(i)$ to be optimized. Let $\vec{x} = \{x(1), \ldots, x(B)\}$ be the set of values stored in the histogram. The overall sum-squared error is

$$\sum_{l,r} {\delta'}_{lr}^2 = \sum_{a \le b} \left(s[a,b] - \sum_{a \le i \le b} x(\mathrm{buck}(i)) \right)^2 ,$$

a polynomial of degree 2 in the coordinates of $\vec{x}$. That is,

$$\sum_{l,r} {\delta'}_{lr}^2 = \vec{x}Q\vec{x}^T + \vec{g}\vec{x}^T + c, \qquad (3)$$

where Q is a $B \times B$ matrix depending on the bucket boundaries only, $\vec{g}$ is a vector of size B, and c is a constant. The matrix Q and vector $\vec{g}$ can be found in time $O(N + B^2)$. System (3) has a single local minimum, that can be found by solving $2\vec{x}Q + \vec{g} = \vec{0}$ in the time it takes to invert a $B \times B$ matrix, at most $O(B^3)$. The resulting approximation is denoted as A-reopt for any original histogram algorithm $\mathcal{A}$. The reopt-ing operation takes time $O(N + B^{O(1)})$.

This reopt-ing strategy may help certain histogram optimizations where the summary values stored in buckets is not optimized in the definition of the histogram (such as the classical definition where we store just the average), but will not help in other cases (e.g., in SAP0 or SAP1, which already optimize over summary values.) We did a preliminary experiment with A0-reopt on our dataset and it was superior and up to 41% better than OPT-A, with respect to the SSE. However more extensive experiments have to be performed (e.g., does OPT-A-reopt significantly outperform OPT-A?). This result is encouraging because the methodology above is general.

References

[1] S. Acharya, P. B. Gibbons, V. Poosala and S. Ramaswamy. The Aqua Approximate Query Answering System. *Proc. of SIGMOD*, 1999

[2] E. W. Cheney. Introduction to approximation theory. Chelsea, 1982.

[3] S. Chaudhuri, R. Motwani and V. R. Narasayya. Random Sampling for Histogram Construction: How much is enough? *Proc. of SIGMOD*, 1998.

[4] M. Charikar, S. Chaudhuri, R. Motwani and V. R. Narasayya Towards Estimation Error Guarantees for Distinct Values. *Proc. of PODS*, 2000.

[5] I. Daubechies, Ten lectures on wavelets. SIAM, Philadelphia, PA, 1992.

[6] H. Jagadish, N. Koudas, S. Muthukrishnan, V. Poosala, K. Sevcik and T. Suel. Optimal histograms with quality guarantees. *Proc. Very Large Databases* (VLDB), 1998.

[7] J. M. Hellerstein, P. J. Haas and H. Wang Online Aggregation. *Proc. of SIGMOD*,1998.

[8] Y. Ioannidis. Universality of Serial Histograms. *Proc. Very Large Databases* (VLDB), 1993.

[9] N. Koudas, S. Muthukrishnan and D. Srivastava. Optimal Histograms for Hierarchical Range Queries. *Proc. of PODS*,2000

[10] M. V. Mannino, P. Chu and T. Sager. Statistical Profile Estimation in Database Systems. *ACM Computing Surveys 20(3): 191-221*, Seot 1988.

[11] Y. Matias, J. S. Vitter, and M. Wang. Wavelet-based histograms for selectivity estimation. *Proc. of SIGMOD'98*, Seattle, WA, 1998.

[12] V. Poosala. Histograms in database query optimization. *PhD Thesis*, U. Wisconsin, Madison, 1997.

[13] G. P. Shapiro and C. Connell. Accurate Estimation of the Number of Tuples Satisfying a Condition. *Proc. of SIGMOD*, 1984.

[14] E. Tufte. The visual display of quantitative information. Graphics Press, 1992.

[15] J. W. Tukey. Exploratory data analysis. Reading, Mass., Addison-Wesley, 1977.

[16] M. V. Wickerhauser. Adapated wavelet analysis from theory to software. A.K.Peters, Wellesley, MA, 1992.

[17] J. S. Vitter and M. Wang Approximate Computation of Multidimensional Aggregates of Sparse Data Using Wavelets. *Proc. of SIGMOD*, 1999.

[18] G. K. Zipf. Human Behaviour and the Principle of Least Effort: an Introduction to Human Ecology. *Addison-Wesley*, 1949