Rangesum Histograms*

S. Muthukrishnan Martin Strauss

Abstract

A *rangesum query* to an array $\mathbf{A}$ is a pair (ℓ, r) of range endpoints, which should be answered by $\sum_{\ell \le i < r} \mathbf{A}[i]$. To compress $\mathbf{A}$, we consider representing an array $\mathbf{A}$ lossily by a *histogram*, a function that is constant on each of a small number of buckets. We then answer range queries from $\mathbf{H}$ instead of from $\mathbf{A}$, *i.e.*, as $\sum_{\ell \le i < r} \mathbf{H}[i]$. An optimal rangesum histogram $\mathbf{H}$ for this purpose is one whose bucket boundaries and constant heights within buckets are chosen to minimize the expected square error, $E_{\ell,r}\left[\left(\sum_{\ell \le i < r} \mathbf{A}[i] - \sum_{\ell \le i < r} \mathbf{H}[i]\right)^2\right]$, assuming each rangesum query is equally likely. Rangesum histograms find many applications in database systems.

In a degenerate variation, all rangesum queries are over ranges of size one, namely, individual points; histograms optimal for this special case are called *pointwise optimal histograms*. Pointwise optimal histogram is a classical notion in statistics and approximation theory, but rangesum optimal histogram appears to be novel in these areas. While optimal pointwise histograms can be constructed efficiently by simple dynamic programming, no efficient (even approximate) general rangesum histogram construction algorithms were previously known. In practice, all commercial database systems use heuristically built histograms for pointwise and rangesum queries.

We present the first general algorithms for approximate rangesum histograms. Given parameter B, we denote by (α, β)-approximation an algorithm to produce a (αB)-bucket histogram with error at most β times the error of the optimal B-bucket histogram. We give a $(2, 1)$-approximation with runtime $O(N^2 B)$, a $(2, 1+\epsilon)$-approximation with runtime $N + (B \log(N)/\epsilon)^{O(1)}$, and a $(1, 1+\epsilon)$-approximation with runtime $O(B^3 N^4/\epsilon^2)$. We also consider the problem of dynamic maintenance of rangesum histograms for data updated by additive changes, and we give a $(2, 1+\epsilon)$-approximation that uses space $(B \log(N)/\epsilon)^{O(1)}$ and time $(B \log(N)/\epsilon)^{O(1)}$ for update and query operations. The bounds are nearly competitive with some of the best known bounds for constructing pointwise optimal histograms modulo small additional number of buckets used; however, rangesum histograms are substantially harder to construct because of the long range dependence between subproblems.

1 Introduction

Histograms are piecewise-constant approximations of signals. More formally, a *B-bucket histogram* $\mathbf{H}$ is defined by a *partition* of $[0, N-1]$ into B intervals, I_k, $1 \le k \le B$, with integer endpoints, together with bucket heights b_k. Given an input signal $\mathbf{A}[0 \cdots N-1]$, the histogram is an approximation to the $\mathbf{A}$ in which $\mathbf{A}[i]$ is approximated by $\widetilde{\mathbf{A}[i]} = b_k$ for $i \in I_k$. Of course, B is always at most N; typically, $B \ll N$. Histograms are fundamental statistical entities that are used to summarize the data and capture trends in the signal, typically for visualization. In Computer Systems, in particular in Database Systems, they are extensively used for cost-based query optimization — estimating result sizes of intermediate tables in complex queries so that optimizers can choose efficient plans for executing complex queries. Every commerical database system uses histograms, of one sort or another. See [15].

The goal in constructing appropriate histograms is to optimize them for specific applications. Typically, histograms have been applied for approximating the signal pointwise. Then, the histogram construction problem is to determine the partition I_k's and bucket heights b_k's such that the total error, say the sum squared error, of approximating the signal at each point is minimized, *i.e.*, $\sum_i (\mathbf{A}[i] - \widetilde{\mathbf{A}[i]})^2$ is minimized; here, the assumption is that each point is equally likely to be approximated using the histogram by the application. We call this the *pointwise optimal* histogram. It is well known (and can be easily proven that) that in a pointwise optimal histogram, the bucket height b_k is the average of $\mathbf{A}[i]$'s for all $i \in I_k$. Given this observation, a simple dynamic programming solution computes the pointwise optimal histogram efficiently in $O(N^2 B)$ time [11].

In general, however, the definition of optimal histogram depends on the application. In Database Systems, histograms are used to approximate answers to queries, and the query that is widely popular is a *rangesum query* where we wish to determine $S[i,j] = \sum_{k=i}^{k=j} \mathbf{A}[k]$. Rangesum query is ubiquitous in Database Systems, both as part of user posed query transactions, datacube operations and mining, as well as in optimizing internal operations of the system. Formally, a range query is a pair of integers (ℓ, r), with $\ell < r$. The query to $\mathbf{A}$ should return $\mathbf{A}[\ell, r) = \sum_{\ell \le i < r} \mathbf{A}[i]$. Instead, let $\mathbf{H}[\ell, r)$ represent the approximation using the histogram. *Rangesum optimal histogram* is defined to be

*AT&T Labs—Research, 180 Park Avenue, Florham Park, NJ 07932 USA, {`muthu`, `mstrauss`}`@research.att.com`.

one that minimizes the total error over all range queries, where each range query is equally likely, *i.e.*, one for which $E_{\ell<r}(\mathbf{A}[\ell,r) - \mathbf{H}[\ell,r))^2$ is minimized. There are signals $\mathbf{A}$ for which the pointwise optimal and rangesum optimal histograms produce significantly different errors in approximating the underlying $\mathbf{A}$. (See [12] for an example.) Curiously, we do not know of a reference to rangesum histograms in the statistics or mathematical approximation theory literature, where pointwise histograms are classical.

Surprisingly, there are only a few results on constructing rangesum optimal histograms. Database systems use heuristic methods [15] for constructing simple histograms which are used as a substitute for both pointwise optimal histograms as well as rangesum histograms; most times they are not optimal for either pointwise or rangesum histograms. In Database research literature, several approaches have been suggested for constructing pointwise histograms, some with provable approximation guarantees; typically these suggestions get evaluated not only at pointwise queries, but also for rangesum queries for which they are not opimized [15, 11]. The only known algorithmic results make simplifying assumptions. For example, in [6], the authors fixed each bucket height b_k to be the average of $\mathbf{A}[i]$'s for all $i \in I_k$, and presented a pseudo-polynomial time algorithm, that is, one with running time polynomial in B, N and $\|\mathbf{A}\|_1 = \sum_i |\mathbf{A}[i]|$. They also gave polynomial time algorithms to construct certain variants of histograms, but left open the problem of constructing (near) optimal In [12], the authors *further* made the assumption that the set of all range queries are *hierarchical*, *i.e.*, if two ranges that are queries intersect, then one is contained in the other. In that case, the ranges can be arranged in a tree structure and a dynamic programming approach gives a polynomial time algorithm for obtaining the optimal rangesum histogram. Under the same assumptions, the authors [9] present significantly faster algorithms that provide approximate rangesum histograms while allowing a constant factor more buckets.

The underlying assumptions in [6, 12, 9] are ultimately limiting. They do not directly apply for constructing the optimal rangesum histogram because, for optimal representations, each bucket height b_k needs to be arbitrary, not fixed to be the average of $\mathbf{A}[i]$'s for all $i \in I_k$. Also, perhaps more importantly, rangesum queries need not be hierarchical in general. The difficulty of designing (near) optimal rangesum histograms comes from the *long range dependence* in the subproblems of constructing rangesum histograms for two distant parts of the signal. More specifically, dynamic programming approach that works for pointwise optimal histograms or for hierarchical rangesums rely on being able to partition the problem into two (or more) independent pieces. However, when queries are rangesum queries, we need to account for not only the error of ranges that fall into each of the subproblems fully, but also for range queries that span the subproblems. Since all range queries are possible, every pair of subproblems have dependent error, and standard approachs fail.

We present the first known algorithmic results for approximating rangesum optimal histograms. We consider two versions of the problem.

- *Offline.* We are given signal $\mathbf{A}$ and number B as input. Our goal is to construct (an approximation of) the rangesum optimal histogram.
- *Online Maintenance.* We have an underlying signal $\mathbf{A}$ which is initially all zero. We are given updates to the signal, namely, add or subtract values to signal components $\mathbf{A}[i]$'s. Each query is to construct (an approximation of) the rangesum optimal histogram on the signal that is the result of all the updates so far. Updates and queries come as a series of intermixed transactions.

Both versions of the problem are of interest in applications. Database systems for example tend to use the offline solution periodically (say every night) to build statistics on the data; however, proposals are underway to make database systems be more responsive to changes in the data they manage by modifying histograms frequently, amidst updates [1, 4]. This calls for the online maintenance of histograms.

Our main results are as follows. (Recall that, typically, $B \ll N$.) Define (α, β) approximation to the rangesum optimal histogram to be one in which we use at most αB buckets and have total error at most $\beta\delta^*$ where, δ^* is the optimal error.

1. *We present algorithms for the offline version. These include a $(2,1)$-approximation, a $(1, 1+\epsilon)$-approximation, a linear time algorithm, and a sublinear-space algorithm.*

   These are the first known approximation results for rangesum optimal histograms. For typical values of B and ϵ, the result of Theorem 3.2 runs in the time to make a handful of passes over the data and uses space just a small constant multiple of the data size; it is quite simple and implementable, and may be of use in practice. The result of Theorem 3.3 is reassuring since it shows that we do not have to use extra space to approximate the error of the optimal rangesum histogram efficiently.

2. We present an algorithm for online maintenance of rangesum histograms. Specifically, *we present*

an algorithm that takes time $(B \log N/\epsilon)^{O(1)}$, ie., polynomial in $\log N$ and B, for each update as well as each query operation outputing $(2, 1+\epsilon)$ approximation to the optimal rangesum histogram.[1]

These are again first known results for online maintenance of rangesum histograms. The algorithm maintains only sublinear space, specifically, space $(B \log N/\epsilon)^{O(1)}$.

These bounds are essentially competitive with the best known bounds for pointwise optimal histograms for offline [11, 7] as well as online setting shown recently [5], only slightly worse in the number of buckets used.

We will now provide a technical overview of our results. Our $(2, 1)$-approximation is simplest, and it is obtained by introducing an *alternating histogram* where every other bucket has size one. To the best of knowledge alternating histograms are novel. We show that using an optimal or near-optimal alternating histogram for rangesum queries to $\mathbf{A}$ is equivalent to using an optimal or near-optimal piecewise-linear pointwise approximation to the prefix array $(\Sigma\mathbf{A})$ to $\mathbf{A}$. The piecewise-linear approximation to the prefix sum of the signal can be obtained by dynamic programming quite easily. The same transformation would not work with general rangesum histograms which, intuitively, would be equivalent to *continuous* piecewise-linear approximations to the prefix sum of the signal, which we do not know how to solve optimally. Our process loses a factor of 2 in buckets going from rangesum histogram to the alternating histogram and back. On the other hand, the $(1, 1+\epsilon)$ approximation follows a different approach. Here, we take the basic dynamic programming approach for rangesum histograms, but bound the interaction between subproblems by careful analyses.

Our online maintenance algorithm depends on many technical steps. Here, we follow the alternating histogram approach and settle for $(2, 1)$-approximation as a starter. So, the problem becomes one of maintaining a piecewise-linear histogram on the prefix sum of the signal. Here, we borrow from the recent approach in [5] for maintaining a piecewise constant histogram on a signal. There are two steps: first produce what is known as the *robust histogram* and second, convert the robust histogram to a standard one. For our setting, each of these steps presents new technical challenges. We develop technical machinery to work through these challenges, and obtain the online maintenance result giving up an $1+\epsilon$ factor in the error. The machinery we have developed may find more applications in the future.

[1] We ignore factors of $\log\left(\sum_i |\mathbf{A}[i]|\right)$ in the costs of some algorithms, since we assume $\log\left(\sum_i |\mathbf{A}[i]|\right)$ is at most $O(\log(N))$.

2 Preliminary Definitions

A *signal* is an integer-valued array of length N. It will be convenient to regard the sample points as sitting on positions $\{0, \ldots, N-1\} = [0, N)$.

We will be interested in two formats for presenting $\mathbf{A}$: either in aggregate, in which the elements $\mathbf{A}[i]$ are presented in order and available in a traditional random access array; or the *dynamic maintenance model.* In this model, there are two operations, *update* and *query.* The signal is implied by updates of the form "add v to $\mathbf{A}[i]$," where v may be a positive or negative integer, and we will want to construct a histogram from the then-active signal upon a query operation.

A *range query* is a pair of integers (ℓ, r) (sometimes written $[\ell, r)$). First assume $\ell < r$. The query to $\mathbf{A}$ should return $\mathbf{A}[\ell, r) = \sum_{\ell \le i < r} \mathbf{A}[i]$. Note that there are $r - \ell$ elements of $\mathbf{A}$ that contribute to the range query (ℓ, r). It will also be convenient to define the query $\mathbf{A}[\ell, \ell)$ to be zero and $\mathbf{A}[r, \ell) = -\mathbf{A}[\ell, r)$.

If $\mathbf{H}$ is a(n approximate) representation for $\mathbf{A}$, then we will be interested in the expected square error in using $\mathbf{H}$ to answer range queries to $\mathbf{A}$. That is, we are interested in the quantity $E_{\ell,r}(\mathbf{A}[\ell, r) - \mathbf{H}[\ell, r))^2$, where (ℓ, r) is chosen uniformly from $N \times N$. Note that, because of our convention, this is also $\frac{N(N-1)}{N^2}$ times the expected error over all uniformly-chosen distinct ℓ and r, since the query $[\ell, \ell)$ is answered with zero error. The quantity $E_{\ell,r}(\mathbf{A}[\ell, r) - \mathbf{H}[\ell, r))^2$ is also $\frac{N(N-1)}{N^2}$ times the expected square error over all ℓ and r chosen conditioned on $\ell < r$, since $[\ell, r)$ and $[r, \ell)$ are answered with the same square error.

A *B-bucket histogram* $\mathbf{H}$ of length N is defined by a partition of $[0, N)$ into B intervals, I_k, $1 \le k \le B$, with integer endpoints, together with bucket heights b_k. The histogram itself is a signal, such that $\mathbf{H}[i] = b_k$ for $i \in I_k$. We also write $I_k^<$ for the left-endpoint of I_k and $I_k^>$ for the right-endpoint of I_k, and we write $k = \mathrm{buck}(i)$ if $i \in I_k$, *i.e.*, if $i \in [I_k^<, I_k^>)$. (Note that a bucket includes its left endpoint but excludes its right endpoint.) One can form range queries to $\mathbf{H}$ with answer $\mathbf{H}[\ell, r) = \sum_{\ell \le i < r} b_{\mathrm{buck}(i)} = \sum_{\ell \le i < r} \mathbf{H}[i]$. We also write $\mathbf{H}[I_j)$ for $\mathbf{H}[I_j^<, I_j^>)$.

The *prefix array* $(\Sigma\mathbf{A})$ of a signal $\mathbf{A}$ of length N is an array of length $N+1$ with data at the integer points on $[0, N]$, such that $(\Sigma\mathbf{A})[j] = \sum_{i<j} \mathbf{A}[i]$. (In particular, $(\Sigma\mathbf{A})[0]$ is always 0.) Note that, for any signal $\mathbf{A}$, the range query $\mathbf{A}[\ell, r)$ is equal to the difference $(\Sigma\mathbf{A})[r] - (\Sigma\mathbf{A})[\ell]$ of the two corresponding point queries to the prefix array.

The *delta* $\Delta(\mathbf{S})$ of an array $\mathbf{S}$ of length $N+1$ indexed on $[0, N]$ is an array of length N indexed on $[0, N)$, such that $\Delta(\mathbf{S})[i] = \mathbf{S}[i+1] - \mathbf{S}[i]$. (This is the reason for including $(\Sigma\mathbf{A})[0] = 0$ in the definition of a prefix array.)

3 Offline Algorithms

In this section, we give two offline approximation algorithms, based on two different dynamic programming techniques. We give a $(2,1)$-approximation that runs in time $O(BN^2)$ and a $(1, 1+\epsilon)$-approximation that runs in time $O(B^3N^4/\epsilon^2)$. As a corollary of the $(2,1)$-approximation, we give a $(2, 1+\epsilon)$-approximation with runtime $O(N + B^3/\epsilon^2)$ and space cost $O(N + B^2/\epsilon)$.

3.1 A $(2,1)$-Approximation In this section, we give a $(2,1)$-approximation. An *alternating histogram* is a histogram with an odd number $2B-1$ of buckets, indexed 1 to $2B-1$, such that $B-1$ alternating buckets (with even index) have size one.

A *piecewise-linear* array of B pieces is an array on $[0, N]$: a partition of $[0, N]$, and a linear function associated with each interval of the partition. Observe that, if $\mathbf{S}$ is a piecewise-linear array of B pieces, then $\Delta(\mathbf{S})$ is an alternating histogram of $2B-1$ buckets, with partitioning induced from the partition of $\mathbf{S}$. For example, suppose $N = 7$ and $\mathbf{S} = (3, 5, 7, 9, 11, 1, 4, 7)$ is an array of length $N + 1 = 8$ with $B = 2$ pieces of sizes 5 and 3. Then $\Delta(\mathbf{S}) = (2, 2, 2, 2, -10, 3, 3)$ is an alternating histogram of length $N = 7$ and $2B - 1 = 3$ buckets of sizes $4 = 5-1$, 1, and $2 = 3-1$ corresponding to the pieces of sizes 5 and 3 in $\mathbf{S}$. The prefix array of $\Delta(\mathbf{S}) = (2, 2, 2, 2, -10, 3, 3)$ is $(0, 2, 4, 6, 8, -2, 1, 4)$, which differs from $\mathbf{S}$ in each position by $\mathbf{S}[0] = 3$.

A piecewise linear array is called *unbiased* if the expected entry is zero. For example, the expected entry of $\mathbf{S} = (3, 5, 7, 9, 11, 1, 4, 7)$ is $\mu = 47/8$, so $\mathbf{S} - \mu = (3-\mu, 5-\mu, 7-\mu, 9-\mu, 11-\mu, 1-\mu, 4-\mu, 7-\mu)$ is unbiased.

As datasets, clearly a B-bucket histogram is a $(2B-1)$-bucket alternating histogram (in which each singleton bucket has the same value as a neighboring bucket), and a $(2B-1)$-bucket alternating histogram is a $(2B-1)$-bucket histogram. Thus, it suffices to find an optimal $(2B-1)$-bucket alternating histogram.

Note that, if we use $\mathbf{H}$ to answer range queries to $\mathbf{A}$, the error for range query $[\ell, r)$ is the difference in error in using $\mathbf{H}$ to answer two prefix queries. *i.e.*, the difference of errors in using $(\Sigma\mathbf{H})$ to answer point queries to $(\Sigma\mathbf{A})$. The correspondence goes further:

Lemma 3.1. *Fix a signal* $\mathbf{A}$ *and histogram* $\mathbf{H}$ *such that* $(\Sigma(\mathbf{A} - \mathbf{H}))$ *is unbiased. If we answer range queries to* $\mathbf{A}$ *from* $\mathbf{H}$*, the expected square error equals twice the expected error of using* $(\Sigma\mathbf{H})$ *to answer point queries to* $(\Sigma\mathbf{A})$.

Proof: Let $X[i] = \mathbf{A}[0, i) - \mathbf{H}[0, i)$ be the error vector. By hypothesis, $E_i[X[i]] = 0$. Let X and Y be iid according to $X[i]$ for random i. Then $E_{\ell,r}[(X[\ell] - X[r])^2] = E[(X-Y)^2] = E[X^2] - 2E[X]E[Y] + E[Y^2] = 2E[X^2]$, since $E[X] = E[Y] = 0$ and $E[Y^2] = E[X^2]$. ∎

For any candidate representation $\mathbf{H}$, let $\mu = E[\mathbf{A}[0, i) - \mathbf{H}[0, i)] = E[X]$. Then the error in using $(\Sigma\mathbf{H}) + \mu$ to approximate $\mathbf{A}$ for range queries is unchanged (since μ cancels for the left and the right prefix approximation), whereas the error in using $(\Sigma\mathbf{H}) + \mu$ for prefix queries is optimal among represenations of the type $(\Sigma\mathbf{H}) + c$:

$$E[(X + c)^2] = E[X^2] - 2c\mu + c^2,$$

a quadratic in c that is minimized at $c = \mu$.

Corollary 1. *Let* $\mathbf{A}$ *be any signal, and suppose* $(\Sigma\mathbf{H})$ *is the best (or near-best)* B*-piece piecewise-linear representation for* $(\Sigma\mathbf{A})$ *under point queries. Then* $\mathbf{H}$ *is a best (or near-best)* $(2B-1)$*-bucket alternating histogram for representing* $\mathbf{A}$ *under range queries.*

Proof of the above corollary is omitted. Now, using dynamic programming:

Lemma 3.2. *Given* $\mathbf{S}$*, one can find the optimal piecewise-linear represenation to* $\mathbf{S}$*, in time* $O(N^2B)$.

Theorem 3.1. *Given a signal* $\mathbf{A}$*, in time* $O(N^2B)$*, one can find a* $(2,1)$*-approximation to the best* B*-bucket histogram to represent* $\mathbf{A}$ *for range queries.*

Note that an alternating histogram requires roughly $3B$ words of storage—B bucket boundaries and $2B$ heights—since one does not need to store a bucket boundary for buckets of size 1. Thus an alternating histogram of $2B-1$ buckets uses roughly 1.5 times the space of a B-bucket histogram, and the result of this section can be regarded as a $(1.5, 1)$-approximation in terms of space.

A piecewise-linear array $\mathbf{S}$ of B pieces is called *continuous* if the associated even-indexed buckets in $\Delta\mathbf{S}$ (of size 1) have value equal to their right neighbor. In that case, $\Delta\mathbf{S}$ may be regarded as a histogram of B buckets. For example, if $\mathbf{S} = (0, 2, 4, 6, 8, 11, 14, 17)$ consists of two pieces of sizes 5 and 3, then $\Delta\mathbf{S} = (2, 2, 2, 2, 3, 3, 3)$. Note that we could also regard $\mathbf{S}$ as having buckets of sizes 4 and 4; *i.e.*, the value 8 agrees with the lines in both the first and second pieces. This is the basis of an alternative characterization of continuity.

Finding an optimal B-piece continuous linear array for point queries to $(\Sigma\mathbf{A})$ is equivalent to finding the best B-bucket histogram for range queries to signal. We do not know how to do this efficiently, in general. But consider a k-*alternating histogram*, in which every k'th bucket has size one, but the others are arbitrary. On the prefix domain, this is a equivalent to a piecewise-linear array in which most boundaries are continuous, but every k'th boundary may be discontinuous. By a straightforward generalization of the natural dynamic programming algorithm, we can optimize over such objects in time $O(BN^{k+1})$, thereby getting a $(1 + 1/k, 1)$-approximation.

One can get a simple linear-time $(2, 1 + \epsilon)$-approximation algorithm using a dual version of dynamic programming. First perform a linear-time preprocessing stage, to convert the signal to its prefix array and to compute prefix arrays for other linear piecewise statistics to be used later. Store these in linear space. Next, assume we know the the optimal sum square error δ^2 to the factor 2. If we do not know δ^2, we can rerun the algorithm for various powers of 2. Specifcally, we can do binary search to find the best power of 2, with factor $\log\log(N)$ by the assumptions of this paper. Quantize the error on each piece into units of $\epsilon\delta^2/B$; there are only $O(B/\epsilon)$ values to track and the accumulated quantization error is at most B times as big, namely, $\epsilon\delta^2$, as desired. For each number $k' < k \le B$ of pieces and each $\ell \le 2B/\epsilon$, assume that, inductively, we have found the maximum i such that some k'-piece piecewise-linear array covers $[0, i)$ with error at most $\ell\epsilon\delta^2/B$. To extend this table from k to $k + 1$, we try extending, by a single piece, each of our representations of $(k - 1)$ pieces (there are $O(B/\epsilon)$, one for each quantized error), and try all $O(B/\epsilon)$ possible allocations of quantized error between the first $k - 1$ pieces and the last piece. This is similar to [8]. We can use binary search to find i in time $O(\log(N))$. Thus each extension takes time $\log(N)(B/\epsilon)^2$, there are $O(B)$ extensions to make, for $1 \le k \le B$, and there are at most $\log(N)$ iterations to find a bound on the optimal error, for a total time of $O\left(N + B^3 \log^2(N)/\epsilon^2\right)$.

Theorem 3.2. *There is a dynamic programming algorithm, that, for each N, B, and ϵ, finds a $(2, 1 + \epsilon)$-approximation to the best B-bucket histogram and runs in time $O\left(N + B^3 \log^2(N)/\epsilon^2\right)$.*

3.2 A $(1, 1 + \epsilon)$-Approximation We now give a $(1, 1 + \epsilon)$-approximation for B-bucket histograms. In this section, we mostly switch from expected square error to sum square error, since this will be more convenient.

In the natural way to use dynamic programming, we want to find the best k-bucket histogram on $[0, i)$, for $k \le B$ and $i \le N$. If the optimal $(k + 1)$-bucket histogram decomposes into an optimal k-bucket histogram and an optimal 1-bucket histogram, then it is easy to construct each histogram from previously-constructed ones. The central difficulty in building optimal histograms for range queries, however, is that optimal $(k+1)$-bucket histograms *do not* decompose this way, due to long-range dependence.

Our overall approach is as follows. For a histogram $\mathbf{H}$ defined on $[0, i)$, $i < N$, with last bucket $[i', i)$, we will define below a parameter $S(\mathbf{H}[0, i))$, that depends only $S(\mathbf{H}[0, i'))$ and on values of $\mathbf{H}$ and $\mathbf{A}$ in $[i', i)$. Also, we will define a cost function C on buckets of $\mathbf{H}$ such that

- $C(I_j)$ depends only on values in the histogram or signal within bucket j and on the (approximate) value of $S(\mathbf{H}[0, I_j^<))$.
- For any histogram on $[0, N)$, $\sum_j C(I_j)$ is the total sum square error. Thus we can sensibly apply the error function C to a prefix of buckets in $\mathbf{H}$, by summing the contribution of each bucket.

Inductively, we suppose that, for each $k \le B$, each $j \le i$, and each value s of $S()$, we have found an optimal k-bucket histogram $\mathbf{H}$ on $[0, j)$ with $S(\mathbf{H}) \approx s$. We have stored the histogram itself and its error, in a table. To extend our table from i to $i + 1$, we proceed as follows. For for each k, consider all possible $i' < i + 1$ and all possible s' and s. Take the best k-bucket histogram, according to $C()$, such that $\mathbf{H}$ has $k - 1$ buckets up to i' and a k'th bucket equal to $[i', i + 1)$, $S(\mathbf{H}[0, i')) \approx s'$ and $S(\mathbf{H}[0, i)) = s$. Record just the one best histogram on $[0, i + 1)$ for each possible (approximate) value of $S(\mathbf{H}[0, i + 1))$. For each s, the best histogram on $[0, i')$ with $S(\mathbf{H}[0, i + 1)) = s$ has *some* $(k - 1)$'st boundary i' and *some* S value s'. All histograms $\mathbf{H}$ with $(k - 1)$ buckets on $[0, i')$ and $S(\mathbf{H}[0, i')) = s'$ are equivalent for the purpose of extension to a k-bucket histogram, and we have considered one such histogram. It follows that we have constructed a nearly optimal k-bucket histogram $\mathbf{H}$ on $[0, i + 1)$ with $S(\mathbf{H}[0, i + 1)) \approx s$. (The result is only nearly optimal because we only approximate S values.) The time to extend the table from i to $i+1$ is at most $O(BN|\Sigma|^2)$, where Σ is the set of possible approximate values for S. This is because we try all possible $k \le B$, all $i' < i + 1 \le N$, and all $s', s \in \Sigma$. To extend from $i = 1$ to $i = N$ then takes time $N^2B|\Sigma|^2$. Once we have extended the table to N, we can take the best B-bucket histogram on $[0, N)$, optimizing over all values for $S(\mathbf{H}[0, N))$. It remains only to define S and C and to determine $|\Sigma|$.

For fixed implicit signal $\mathbf{A}$, define $S(\mathbf{H}[0, i))$ to be $\sum_{0 \le j < i}(\mathbf{A}[j, i) - \mathbf{H}[j, i))$; *i.e.*, the sum raw error in using $\mathbf{H}$ to answer a suffix query $[j, i)$ to $\mathbf{A}[0, i)$. Note that $\sum_{0 \le j < i} \mathbf{A}[j, i) = \sum_{0 \le j < i}(j + 1)\mathbf{A}[j]$ can be computed from i in constant time after a linear time preprocessing step. Also, if i' is the penultimate boundary in $\mathbf{H}[0,i)$, then $\sum_{0\le j<i}(j+1)\mathbf{H}[j] = \sum_{0\le j<i'}(j+1)\mathbf{H}[j] + \sum_{i'\le j<i}(j+1)\mathbf{H}[j]$. Note that, in our application, we will be given $s' = S(\mathbf{H}[0,I_j^<))$ and a desired value $s = S(\mathbf{H}[0,I_j^>))$; there is exactly one bucket height that realizes this, namely, the solution h to

$$s = s' + (\mathbf{A}[I_j) - h|I_j|)I_j^< + \sum_{\ell\in I_j}\left(\mathbf{A}[\ell, I_j^>) - h(I_j^> - \ell)\right)$$

a linear equation in h whose coefficients can be found in constant time, assuming linear-time preprocessing of $\mathbf{A}$.

We now define $C(I_j)$. First, for illustration, suppose there is just a single query, $[\ell, r)$, with $\ell \in I_j$. If also $r \in I_j$, then the error $(\mathbf{A}[\ell,r) - \mathbf{H}[\ell,r))^2$ contributes cleanly to bucket j and to no other bucket. Now suppose buck$(r) = j+2$, so that bucket $j+1$ is overflown. Then the error $(\mathbf{A}[\ell,r) - \mathbf{H}[\ell,r))$ splits into three parts, that we now trace:

$$\begin{cases} X &=& \mathbf{A}[\ell, I_j^>) - \mathbf{H}[\ell, I_j^>) \\ Y &=& \mathbf{A}[I_{j+1}) - \mathbf{H}[I_{j+1}) \\ Z &=& \mathbf{A}[I_{j+2}^<, r) - \mathbf{H}[I_{j+2}^<, r). \end{cases}$$

The square error is $(X+Y+Z)^2 = X^2 + Y^2 + Z^2 + 2XY + 2YZ + 2XZ$. We attribute X^2 to bucket j. We attribute $Y^2 + 2XY$ to bucket $j+1$. Note that Y^2 and Y depend only on values in bucket $j+1$ and X is equal to $s' = S(\mathbf{H}[0,I_j^<))$, which is stored, so we can compute the desired quantity according to the requirements. Also, when processing bucket $j+1$, one can compute $s = S(\mathbf{H}[0,I_j^>) = X + Y$ from Y (which depends locally on bucket $j+1$) and X (which is stored). Finally, when we process bucket $j+2$, we can compute Z^2 and $2(X+Y)Z$, since Z is local and $X+Y$ is stored. Attribute $2(X+Y)Z + Z^2$ to bucket $j+2$.

In general, we need to consider all queries and iterate over buckets, rather than trace a single query through several buckets. Consider bucket j. We attribute to it:

- the sum square error of all queries wholly contained in bucket j.
- $(N - I_j^>)X_j^2$, where X_j is the sum suffix error $\mathbf{A}[\ell, I_j^>) - \mathbf{H}[\ell, I_j^<)$, summed over ℓ with $I_j^< \le \ell < I_j^>$. This term is involved in $(N - I_j^>)$ queries with left endpoint in bucket j and right endpoint to the right.
- $I_j^< \cdot (N - I_j^>)Y_j^2$, where $Y_j = \mathbf{A}[I_j) - \mathbf{H}[I_j)$, as above, but now with bucket j being overflown, by $I_j^< \cdot (N - I_j^>)$ queries.
- $2(N - I_j^>)S(\mathbf{H}[0,I_j^<))Y_j$. Here $S(\mathbf{H}[0,I_j^<))$ is the sum (not expectancy) of all suffix errors for queries $[\ell, I_j^<)$ with $\ell < I_j^<$; the factor $(N - I_j^>)$ counts all possible right endpoints.
- $I_j^< Z_j^2$, where Z_j is the sum prefix error $\sum_{I_j^< < r \le I_j^>}(\mathbf{A}[I_j^<, r) - \mathbf{H}[I_j^<, r))$. The factor $I_j^<$ counts all possible left endpoints.
- $2S(\mathbf{H}[0,I_j^<))Z_j$. This counts all queries with left endpoint left of $I_j^<$ and right endpoint r in the range $I_j^< < r \le I_j^>$.

This defines $C(I_j)$.

Above we ideally assumed that, for *any* value s, we stored a histogram $\mathbf{H}$ with $S(\mathbf{H}) = s$. In practice, we will only consider values of s from some finite set Σ of equally-spaced values. We will associate $\mathbf{H}$ with a value of Σ close to $S(\mathbf{H})$. Note that, when we extend a histogram from $[0,i')$ to $[0,i)$, we find the required bucket height h for $[i',i)$ exactly. Also, the error we calculate is precise for the histograms we consider. But, at each stage, when we seek the best histogram, we are actually optimizing only over all histograms with quantized S value, a smaller set. Furthermore, these errors can accumulate. We now bound the damage due to this phenomena.

Our strategy is as follows. We will show inductively that, if we quantize S values finely enough, then we maintain a table of near-optimal partial histograms, so that, for B-bucket histograms, the suboptimality is at most $\epsilon\delta^2$. Here δ^2 is the optimal sum-square error, so our goal is to find a representation with sum-square error at most $(1+\epsilon)\delta^2$. Then $|\Sigma|$ will be $2\max\Sigma$ divided by the quantization unit (the factor 2 arises because there are positive and negative elements of Σ, of the same range). It turns out that, roughly, $\sigma\pi$ enters into the suboptimality, where σ and π are in Σ. So we will use the bound on $\max\Sigma$ twice—first to bound π in determining the precision necessary for σ in $\sigma\pi \le \epsilon\delta^2$ and, second, to bound $|\Sigma|$ given the desired quantization unit.

One can bound $\max(\Sigma)$ by $(NB\sum_i \mathbf{A}[i])^{O(1)}$ and use precision $(\epsilon/(BN))^{O(1)}$ times the input precision (say, integers). This gives a pseudopolynomial runtime, based on $|\Sigma|$, of $((BN/\epsilon)\sum_i \mathbf{A}[i])^{O(1)}$, similar to what was achieved in [6], but this does not scale well as entries in $\mathbf{A}$ grow. Below we will define a Σ with size $O(BN/\epsilon)$, independent of the entries in $\mathbf{A}$, and just linear in N.

We assume that we know δ to within the factor $(1+\epsilon)$; if necessary, try all appropriate powers of $(1+\epsilon)$ as candidate approximations for δ. Recall that we are seeking a representation with sum square error at most $(1+\epsilon)\delta^2$.

First note that Σ values may be assumed to be at most $\sqrt{2N}\delta$ in absolute value, since, otherwise,

$$\sigma^2 = \left(\sum_{0\le\ell<i} \mathbf{A}[\ell,i) - \mathbf{H}[\ell,i)\right)^2 \ge 2N\delta^2 \ge 2i\delta^2,$$

and, by the Cauchy-Schwarz inequality,

$$\sigma^2 = \left(\sum_{0\le\ell<i} \mathbf{A}[\ell,i) - \mathbf{H}[\ell,i)\right)^2 \le i \sum_{0\le\ell<i} (\mathbf{A}[\ell,i) - \mathbf{H}[\ell,i))^2,$$

so the contribution to the sum square error of the suffixes represented by σ already exceeds $2\delta^2 \gg (1+\epsilon)\delta^2$.

Analogous to Σ values are values for sum *prefix* errors, which are likewise bounded by $N\sqrt{2N}\delta$. Consider two boundaries i' and i, $i' < i$, and fix some bucket height between i and i'. Let Y denote the overflight error, as above, and, as above, let Z denote the sum prefix error $\mathbf{A}[i',r) - \mathbf{H}[i',r)$, summed over r in the range $i' \le r < i$. Finally, let P denote the sum prefix error $\mathbf{A}[i,r) - \mathbf{H}[i,r)$, summed over $r \ge i$. Then Z, P, and $Z + Y(N-i) + P$ are all sum-prefix errors, for r in the range $[i',i), [i,N)$, and $[i',N)$, respectively; it follows that each is bounded in absolute value by $\sqrt{2N}\delta$. By the triangle inequality, it then follows that $Y(N-i)$ is bounded by $3\sqrt{2N}\delta \le 5\sqrt{N}\delta$.

Note that, by definition of $C()$, $C(\mathbf{H}[0,i))$ depends on $s = S(\mathbf{H}[0,i'))$ by the additive amount

$$2(N - I_j^{>})sY_j + 2sZ_j = 2s((N - I_j^{>})Y_j + Z_j).$$

From the above discussion, it follows that the coefficient for s is bounded by $5\sqrt{N}\delta$.

Suppose we quantize S values to units of η. Assume by induction that, for $(k-1)$-bucket partial histograms on $[0,i')$, we have a partial histogram with error within $((k-1)/B)\epsilon\delta^2$, additively, of optimal over all such partial histograms with the specified values for i', k, and $S(\mathbf{H}[0,i'))$.

To extend our table to k-bucket histograms on $[0,i+1)$, we consider all $(k-1)$-buckets on $[0,i')$, $i' < i+1$. We ideally want to consider any such histogram; instead, we limit ourselves to those with quantized value $s' = S(\mathbf{H}[0,i'))$. Given s' and $s = S(\mathbf{H}[0,i+1))$, the cost C_k for our constructed histogram is C_{k-1} for the $(k-1)$-bucket histogram plus as', for some a at most $5\sqrt{N}\delta$. When we look for the least cost, because of our quantization of s', we may overestimate the cost of the optimal histogram by $a\eta$ and underestimate the cost of another histogram by $a\eta$, for a total of $2a\eta$. By induction, the histogram on $[0,i')$ that we keep is suboptimal by at most $((k-1)/B)\epsilon\delta^2$. Thus C_k is suboptimal by at most

$$2 \cdot 5\sqrt{N}\delta\eta + ((k-1)/B)\epsilon\delta^2 \le (k/B)\epsilon\delta^2,$$

provided $10\sqrt{N}\delta\eta \le (\epsilon/B)\delta^2$, or $\eta \le \epsilon\delta/(10B\sqrt{N})$. Since $\max\Sigma \le \sqrt{2N}\delta$, it follows that

$$\begin{aligned} |\Sigma| &\le 2\max|\Sigma|/\eta \\ &\le 2(\sqrt{2N}\delta)(10B\sqrt{N}/(\epsilon\delta)) \\ &\le O(BN/\epsilon). \end{aligned}$$

It follows that the overall runtime of our algorithm is $O(BN^2|\Sigma|^2) = O(B^3N^4/\epsilon^2)$. In summary,

Theorem 3.3. *There is an algorithm that, given a signal $\mathbf{A}$ of length N and parameters B and ϵ, returns a $(1, 1+\epsilon)$ approximation to rangesum histogram in time $O(B^3N^4/\epsilon^2)$.*

4 Dynamic Maintenance

In this section, we consider a data structure for the dynamic maintenance version of this problem. By Corollary 1, it suffices to find a near-optimal piecewise-linear representation for $(\Sigma\mathbf{A})$. Therefore, to ease notation, we switch entirely to the prefix domain for approximations. The representations $\mathbf{S}$ and $\mathbf{P}$ should be regarded as piecewise-linear pointwise approximations to $(\Sigma\mathbf{A})$. Our algorithm will transform an update from the signal to prefix domain, and keep a small sketch of $(\Sigma\mathbf{A})$. At query time, the algorithm will find a near-optimal B-piece piecewise-linear representation $\mathbf{S}$ for $(\Sigma\mathbf{A})$, and return $\Delta(\mathbf{S})$, a histogram of $2B-1$ buckets. Thus the overall representation is a $(2, 1+\epsilon)$ approximation.

Our algorithms follow those in [5]. The overall structure is to provide a data structure that supports updates and construction of a *robust* piecewise-linear array $\mathbf{S}_r$, that we define below. We then give an algorithm for producing output from the robust piecewise-linear array. The output $\mathbf{H}$ is simply $\Delta\mathbf{S}$, where $\mathbf{S}$ is the best B-piece approximation to $\mathbf{S}_r$. We show how to construct $\mathbf{S}$ from $\mathbf{S}_r$ (both arrays of length $N+1$) in time $(B\log(N)/\epsilon)^{O(1)}$.

Definition 1. A (B_r, ϵ_r)-*robust* piecewise-linear array $\mathbf{S}_r$ for $(\Sigma\mathbf{A})$ is such that, if $\mathbf{S}$ is any piecewise-linear array with the boundaries of $\mathbf{S}_r$ and at most B_r additional boundaries, and $\mathbf{S}$ has optimal spline parameters, then the error of using $\mathbf{S}$ for point queries to $(\Sigma\mathbf{A})$ is at most $(1+\epsilon_r)$ times the error of using $\mathbf{S}_r$ for point queries to $(\Sigma\mathbf{A})$. ∎

Lemma 4.1. *Fix N, B_r, and ϵ_r. There's a data structure that handles updates to a signal of length N and* *produces a (B_r, ϵ_r)-robust approximation $\mathbf{S}_r$ for $(\Sigma\mathbf{A})$. The time to process an update, the time to produce the output, and the space of the data structure are polynomial in $(B_r \log(N)/\epsilon_r)$.*

In particular, $\mathbf{S}_r$ has at most $(B_r \log(N)/\epsilon_r)^{O(1)}$ pieces.

Proof: [sketch] The algorithm builds on [5], where an algorithm was given for a robust piecewise-constant representation instead of piecewise-linear. We now summarize that algorithm in our context, indicating the technical differences between the algorithm as presented in [5] and the algorithm we need here.

We build a variant of the array sketch data structure in [5]. Our data structure represents the prefix array $(\Sigma\mathbf{A})$ of a signal $\mathbf{A}$ as a *sketch* and supports the following fundamental operations:

- Given an update to $\mathbf{A}$, simulate the corresponding update to $(\Sigma\mathbf{A})$.
- Estimate the square norm $\sum_i ((\Sigma\mathbf{A})[i])^2$ of $(\Sigma\mathbf{A})$.
- Find dyadic intervals I such that $(\Sigma\mathbf{A})$ restricted to I has large norm.[2]
- For each dyadic interval I with high norm, estimate parameters for the best linear representation to $(\Sigma\mathbf{A})$ on I.
- Quickly sketch any piecewise-linear representation $\mathbf{A}$ of few pieces.

With this data structure, one builds a robust piecewise-linear representation for $(\Sigma\mathbf{A})$ under point queries. To do this, as in [5], start with $\mathbf{S}$ equal to the zero piecewise linear histogram. Repeatedly do the following. Seek dyadic intervals where $(\Sigma\mathbf{A}) - \mathbf{S}$ has large norm. For each such interval I, find near-best parameters a and b, where "best" means minimizing the error of $\mathbf{S}+(ai+b)\chi_I$ to $(\Sigma\mathbf{A})$, where $(ai+b)\chi_i$ takes the value $ai + b$ on interval I and zero elsewhere. For the near best interval I under the near best parameters, put $\mathbf{S} = \mathbf{S} + (ai + b)\chi_I$, and continue looping. Continue looping while there are improvements to make. One can show that if the growing representation is not already good enough, then there is some dyadic interval giving substantial improvement—roughly the factor $\left(1 - \frac{\epsilon}{B\log(N)}\right)$ in decreasing error. Thus the loop terminates in $k \leq (B \log(N) \log \|\mathbf{A}\|_1/\epsilon)^{O(1)}$ steps, since, otherwise, making $k-1$ substantial improvements would recover $(\Sigma\mathbf{A})$ exactly and there would be no improvement to make on the k'th step.

It remains to indicate how to implement the data structure. The data stucture computes $\sum_{i\in S}(\Sigma\mathbf{A})_i X_i$, where $(\Sigma\mathbf{A})[j] = \sum_{i<j} \mathbf{A}[i]$ is the j'th entry of the prefix array for $\mathbf{A}$, the X_i's are unit Gaussian-distributed random variables, and the sets S form a particular dyadic combinatorial design. The random variables are ultimately generated from a pseudo-random number generator, to save space. The only difference in *what* is computed from the data is that, in [5], the data structure represents the signal $\mathbf{A}$ (for which the updates apply directly), whereas here we represent $(\Sigma\mathbf{A})$ instead. The sets S are the same here as in [5]. The random variables are distributed the same way here as in [5], but are constructed differently.

First, consider simulating an update to $(\Sigma\mathbf{A})$ given an update to $\mathbf{A}$. If we add v to position j in $\mathbf{A}$, that results in adding v to each position $i \geq j$ in $(\Sigma\mathbf{A})$. This capability is already present in the data structure of [5].

One can estimate norms and find dyadic intervals with high norm, exactly as in [5].

For each dyadic interval I with high norm, we need to estimate parameters for the best linear representation to $(\Sigma\mathbf{A})$ on I. As in [5], the estimate for the parameters comes from optimizing a quadratic function of the parameters. In [5] there was a single parameter (the bucket height); here there are two parameters for the best line. To find these parameters, we instead optimize a bi-variate quadratic.

We need to sketch quickly any given linear representation $(\Sigma\mathbf{A})$ of few buckets. Specifically, we need to find the sketch of piecewise linear representations, *i.e.*, find $\sum_{i\in I}(ai+b)X_i$. For this, it suffices to find $\sum_{i\in I} iX_i$ and $\sum_{i\in I} X_i$ quickly. In the Naor-Reingold construction [13], this becomes the following. First sample from the joint distribution $(\sum_{i\in[0,N)} X_i, \sum_{i\in[0,N)} iX_i)$. By properties of the Gaussian, this is a two-dimensional Gaussian distribution. Conditioned on an outcome for $(\sum_{i\in[0,N)} X_i, \sum_{i\in[0,N)} iX_i)$, sample from the joint distribution $(\sum_{i\in[0,N/2)} X_i, \sum_{i\in[0,N/2)} iX_i)$. This is also a two-dimensional Gaussian distribution, whose parameters (mean and variance) are easy to compute from the first outcome and N. Continue in this way, sampling from smaller sums, to the joint distribution on $(X_i + X_{i+1}, iX_i + (i + 1)X_{i+1})$. One can write any interval I as a disjoint union of $O(\log(N))$ dyadic intervals, and the above procedure shows how to find $\sum_{i\in I}(ai + b)X_i$ quickly for any dyadic interval. ∎

We turn now to finding the best approximation $\mathbf{S}$ to $\mathbf{S}_r$.

Definition 2. Let $X \subseteq [0, N]$. A *left-ϵ-fracturing* of

[2] A dyadic interval is one whose endpoints are aligned in a particular way, based on halving. Specifically, a dyadic interval is of the form $[k2^j, (k + 1)2^j)$ for *integers* j and k.

X is a set $Y \supseteq X$ such that $0 \in Y$, and, if x_1 and x_2 are consecutive elements in X and y_2 and y_3 are consecutive elements of Y with $x_1 \le y_2 < y_3 \le x_4$, then $(y_3 - 1 - y_2) \le \epsilon(y_2 - x_1)$. Symmetrically, a *right-ϵ-fracturing* is a set $Y \supseteq X$ such that $N \in Y$ and, for x_1, y_2, y_3 and x_4 as above, $(y_3 - 1 - y_2) \le \epsilon(x_4 - y_3)$. Finally, an *$\epsilon$-fracturing* of X is the union of a left-ϵ-fracturing and a right-ϵ-fracturing of X. ∎

LEMMA 4.2. *There exists ϵ-fracturings Y of X with $|Y| \le O(|X|\log(N)/\epsilon)$.*

LEMMA 4.3. *Let $\mathbf{P}$ be a B'-piece piecewise-linear array. Then the best B-piece piecewise-linear representation to $\mathbf{P}$ restricted to an ϵ'-fracturing Y of the boundaries of $\mathbf{P}$ has error at most $(1+\epsilon)$ times that of the best unrestricted B-piece piecewise-linear array for sufficiently small $\epsilon' \ge \Omega(\epsilon)$.*

Proof: Let $\mathbf{S}$ denote an optimal B-piece piecewise-linear pointwise approximation to $\mathbf{P}$. We will show that we can move boundaries in $\mathbf{S}$ to nearby fracture points without changing the error much. Suppose we are given a representation $\mathbf{S}$ with consecutive boundaries at w, x, z, such that x is not a fracture point. We want to move x to a fracture point y with $x < y < z$. Suppose that, on $[w, x)$, $\mathbf{S}$ takes value $ai+b$ at position i and, on $[x, z)$, $\mathbf{S}$ takes value $ci+d$. After moving x to y, in the new histogram $\mathbf{S}$, the value on $[w, y)$ will be defined to be $ai+b$ and the value on $[y, z)$ to be $ci+d$. Thus only the values in $[x, y)$ change. Note that the new values may be suboptimal for the new bucketing.

First observe that we may assume there is at most one boundary of an optimal $\mathbf{S}$ in the interior of any one piece of $\mathbf{P}$, since a $\mathbf{S}$ with two or more such boundaries would remain at least as good if we moved those boundaries to the endpoints of the $\mathbf{P}$ piece.

Given a general approximation $\mathbf{S}$, we need to show that if we move a boundary x right to a nearby fracture point y, getting $\mathbf{S}'$, the error on the piece to the left of the boundary does not increase by much. (The error on the piece to the right can only decrease, under optimal spline parameters, since the piece gets smaller.)

Note that each boundary of $\mathbf{P}$ is a fracture point, so $\mathbf{P}$ is linear on $[x, y)$. It follows that the raw error of answering a point query $i \in [w, y)$ to $\mathbf{P}$ by $\mathbf{S}'$ is a linear function $ai+b$ of i. Thus we may assume that $\mathbf{P} = 0$ and $\mathbf{S} = ai+b$ and we need to show that $\sum_{x \le i < y}(ai+b)^2 \le \epsilon \sum_{w \le i < x}(ai+b)^2$, provided $(y-x) \le \epsilon'(x-w)$.

Next, we may assume that the absolute error $|ay+b|$ at y is at least the absolute error $|aw+b|$ at w, since a proof for this case immediately implies a proof for the other case, by exchanging the roles of w and y. By similar triangles, this implies that $ai+b=0$ for some $v \le \frac{w+y}{2}$, possibly $v = -\infty$.

Next, we may assume $w = v$, by modifying ϵ', if necessary. First, if $w > v$, consider the line $a'i + b'$ that goes through the points $(w, 0)$ and $(x, ax+b)$. Clearly $\sum_{w \le i < x}(ai+b)^2 \ge \sum_{w \le i < x}(a'i+b')^2$ and $\sum_{x \le i < y}(ai+b)^2 \le \sum_{x \le i < y}(a'i+b')^2$, so it suffices to show $\sum_{x \le i < y}(a'i+b')^2 \le \epsilon \sum_{w \le i < x}(a'i+b')^2$. On the other hand, if $w < v \le \frac{w+y}{2}$, then ignore the error contribution on $[w, v)$. Then $\sum_{w \le i < x}(ai+b)^2 \ge \sum_{v \le i < x}(ai+b)^2$. Also, $(x-w) \le 2(x-v)$. So it suffices to show $\sum_{x \le i < y}(ai+b)^2 \le \epsilon \sum_{v \le i < x}(ai+b)^2$ assuming $(y-x) \le 2\epsilon'(x-v)$. Thus, if we are given $(y-x) \le \epsilon'(x-w)$, it follows that $(y-x) \le 2\epsilon'(x-v)$, we'll show that this implies $\sum_{x \le i < y}(ai+b)^2 \le \epsilon \sum_{v \le i < x}(ai+b)^2$, which implies $\sum_{x \le i < y}(ai+b)^2 \le \epsilon \sum_{w \le i < x}(ai+b)^2$.

If $ai+b$ is identically zero, then the error is zero before or after the move. So assume $ai+b$ is not identically zero.

We are left with the case $aw+b=0$, $a \ne 0$, with w, x, y spaced as if under a $(2\epsilon')$-fracturing. (Note that an ϵ'-fracturing is also a $(2\epsilon')$-fracturing.) Then, by the change of variables $i = j + w$, we have

$$\begin{aligned} \sum_{w \le i < x}(ai+b)^2 &= \sum_{0 \le j < x-w}(aj)^2 \\ &= \Theta\left(\frac{a^2}{3}(x-w)^3\right), \end{aligned}$$

where the constants in $\Theta()$ are independent of a and b. It follows that

$$\sum_{w \le i < y}(ai+b)^2 = \Theta\left(\frac{(y-w)^3}{(x-w)^3}\right) \sum_{w \le i < x}(ai+b)^2.$$

Since y is a fracture point and x is not, $(x-w) \le (y-w) \le (1+2\epsilon')(x-w)$, so that $1 \le \frac{(y-w)^3}{(x-w)^3} \le (1+2\epsilon')^3$. The result follows for $\epsilon' \approx \epsilon/6$. ∎

In summary,

THEOREM 4.1. *Fix N, B, and ϵ. There exists a data structure that supports additive pointwise updates to $\mathbf{A}$ and produces, at a query, a $(2B-1)$-bucket (alternating) histogram whose error in answering range queries to $\mathbf{A}$ is at most $(1+\epsilon)$ times optimal. The algorithm takes time $(B\log(N)/\epsilon)^{O(1)}$ for updates and queries and uses space $(B\log(N)/\epsilon)^{O(1)}$.*

Proof: By Corollary 1, it suffices to find a near-best piecewise-linear array $\mathbf{S}$ for point queries to $(\Sigma\mathbf{A})$, then output $\Delta\mathbf{S}$. To do this, by an argument in [7], it suffices to find the best piecewise-linear representation $\mathbf{S}$ to $\mathbf{S}_r$, where $\mathbf{S}_r$ is a (B_r, ϵ_r)-robust approximation, for B_r and ϵ_r^{-1} each $(B\log(N)/\epsilon)^{O(1)}$. The robust representation $\mathbf{S}_r$ is found from a sketch, and the sketch is maintained under updates, by techniques similar to [5]. The natural dynamic programming algorithm runs in time polynomial in the number of potential boundaries of $\mathbf{S}$, *i.e.*, $(B\log(N)/\epsilon)^{O(1)}$. The potential boundaries for $\mathbf{S}$ consists of a $\Theta(\epsilon)$-fracturing of the $(B\log(N)/\epsilon)^{O(1)}$ boundaries in $\mathbf{S}_r$. ∎

Note that our algorithm can be used in an offline setting as well, by considering $\mathbf{A}$ to be a collection of N updates to the zero signal. This gives

THEOREM 4.2. *There is a dynamic programming algorithm, that, for each N, B, and ϵ, finds a $(2, 1+\epsilon)$-approximation to the best B-bucket histogram on integer signals of length N and runs in time $N(B\log(N)/\epsilon)^{O(1)}$. Space used is $(B\log(N)/\epsilon)^{O(1)}$.*

5 Conclusions

We have given the first general algorithms for approximating histograms for rangesum queries. We have given a variety of results, with varying quality of approximation, suitable for different models.

The main challenge in giving optimal histograms for rangesum queries is the long range dependence in subproblems. We give two techniques for addressing this. First, long range dependence does not persist across size-1 buckets in optimal representations. By considering alternating histograms, long range dependence disappears, at the cost of a size-1 bucket between every pair of "real" buckets. Alternatively, this technique could be viewed as a transformation from the signal domain to the prefix array domain, which maps range/prefix queries to point queries. Our other method for addressing long range dependence is to analyze and bound it carefully.

The quartic runtime of our $(1, 1+\epsilon)$-approximation may not be best possible. Note that long range dependence contributes twice—first, it contributes the factor N^2, ultimately due to the Cauchy-Schwarz inequality, in Section 3.2 from the size of Σ. Also, it seems to prevent a lemma analagous to Lemma 4.3. Thus we used full $O(BN^2)$-time dynamic programming for the top-level algorithm instead of a $(B\log(N)/\epsilon)^{O(1)}$-time algorithm possible in this context for point queries.

Throughout we have assumed that each rangesum query is equally likely. This is certainly reasonable in the applications. However, database systems may be forced to profile the rangesum queries over a period of time and determine a suitable probability distribution on the different ranges involved. Hence, we may consider the problem of designing rangesum histograms optimal for a given probability distribution over the ranges. Our results extend to the case when such probability distributions have a product nature (that is, the left and right endpoints (ℓ, r) of a query are drawn independently from the same distribution $\mathcal{D}$: $(\ell, r) \sim \mathcal{D} \times \mathcal{D}$; equivalently, $(\ell, r) \sim \mathcal{D} \times \mathcal{D}$ conditioned on $\ell < r$.) It is open to construct (near) optimal rangesum histograms when the probability distribution over the rangesum queries is arbitrary.

References

[1] A. Aboulnaga, S. Chaudhuri. Self-tuning Histograms: Building Histograms Without Looking at Data. In *Proceedings of the ACM SIGMOD Conference*, 1999, 181–192.

[2] N. Alon, Y. Matias, M. Szegedy. The Space Complexity of Approximating the Frequency Moments. JCSS 58(1): 137–147 (1999).

[3] P. Gibbons and Y. Matias. Synopsis Data Structures for Massive Data Sets SODA 1999, 909–910

[4] P. Gibbons, Y. Matias, and V. Poosala. Fast Incremental Maintenance of Approximate Histograms. VLDB 1997, 466–475.

[5] A. Gilbert, S. Guha, P. Indyk, Y. Kotidis, S. Muthukrishnan, and M. Strauss. Fast, Small-Space Algorithms for Approximate Histogram Maintenance. STOC, 2002.

[6] A. Gilbert, Y. Kotidis, S. Muthukrishnan, M. Strauss. Optimal and Approximate Computation of Summary Statistics for Range Aggregates. PODS 2001, 227–236.

[7] S. Guha, P. Indyk, S. Muthukrishnan, and M. Strauss. Histogramming Data Streams with Fast Per-Item Processing. ICALP 2002.

[8] S. Guha, N. Koudas, and K. Shim. Data-streams and histograms. STOC 2001, 471–475.

[9] S. Guha, N. Koudas and D. Srivastava. Fast algorithms for hierarchical range histogram construction. In Proceedings of the ACM Symposium on Principles of Database Systems (PODS), 2002.

[10] P. Indyk. Stable Distributions, Pseudorandom Generators, Embeddings and Data Stream Computation. FOCS 2000, 189–197.

[11] H. V. Jagadish, N. Koudas, S. Muthukrishnan, V. Poosala, K. Sevcik, and T. Suel. Optimal Histograms with Quality Guarantees. VLDB 1998, 275–286.

[12] N. Koudas, S. Muthukrishnan and D. Srivastava. Optimal histograms for hierarchical range queries. In Proceedings of the ACM Symposium on Principles of Database Systems (PODS), 196-204, 2000.

[13] M. Naor and O. Reingold. Private communication, March, 1999.

[14] N. Nisan Pseudorandom Generators for Space-Bounded Computation. STOC 1990, 204–212.

[15] V. Poosala. *Histograms for selecitivty estimation.* PhD Thesis, U. Wisconsin, Madison. 1997.