Structure-Aware Sampling on Data Streams

Edith Cohen, Graham Cormode, Nick Duffield
AT&T Labs–Research
180 Park Avenue
Florham Park, NJ 07932, USA
edith,graham,duffield@research.att.com

ABSTRACT

The massive data streams observed in network monitoring, data processing and scientific studies are typically too large to store. For many applications over such data, we must obtain compact summaries of the stream. These summaries should allow accurate answering of post hoc queries with estimates which approximate the true answers over the original stream. The data often has an underlying structure which makes certain subset queries, in particular *range queries*, more relevant than arbitrary subsets. Applications such as access control, change detection, and heavy hitters typically involve subsets that are ranges or unions thereof.

Random sampling is a natural summarization tool, being easy to implement and flexible to use. Known sampling methods are good for arbitrary queries but fail to optimize for the common case of range queries. Meanwhile, specialized summarization algorithms have been proposed for range-sum queries and related problems. These can outperform sampling giving fixed space resources, but lack its flexibility and simplicity. Particularly, their accuracy degrades when queries span multiple ranges.

We define new stream sampling algorithms with a smooth and tunable trade-off between accuracy on range-sum queries and arbitrary subset-sum queries. The technical key is to relax requirements on the variance over all subsets to enable better performance on the ranges of interest. This boosts the accuracy on range queries while retaining the prime benefits of sampling, in particular flexibility and accuracy, with tail bounds guarantees. Our experimental study indicates that structure-aware summaries can drastically improve range-sum accuracy with respect to state-of-the-art stream sampling algorithms and outperform deterministic methods on range-sum queries and hierarchical heavy hitter queries.

Categories and Subject Descriptors: G.3 [**Probability and Statistics**] : Statistical Computing

General Terms: Algorithms, Measurement, Performance

Keywords: Structure-aware sampling, VarOpt, Data Streams, Approximate query processing

1. INTRODUCTION

Many applications, such as high speed networks, transaction processing, and scientific experiments, generate large quantities of data in the form of high-volume streams. These streams of ephemeral observations are too large to store in their entirety, so instead we must create a compact summary that captures the core properties of the data. These summaries must enable a variety of post hoc analyses of the data, to identify structure, patterns and anomalies.

As a motivating example, consider the massive transient data that arises in IP networks. Each network element observes a huge number of events, in the form of packets traveling across the network. Certainly, no router keeps a copy of the packets that it sees, but modern devices can maintain aggregate records of the traffic they have routed, in the form of NetFlow logs. These describe flows, in terms of the source and destination addresses, ports, duration and size. For a high speed link, these logs themselves represent high-volume streams, and across a large network add up to a vast amount of data. The network operator needs to collect summaries of these logs to enable analysis of the network health and operation, and to detect anomalies, misconfigurations or attacks in near-real time. These high-level analyses rely on being able to accurately estimate volumes of traffic in various projections of the data: from particular sources, to collections of subnetworks, between particular port combinations and so on. Importantly, the summaries should give high accuracy in small space, so that the cost of collecting, transporting, storing and analyzing them is minimized.

The state of the art for these tasks are techniques based on random sampling. The problem is abstracted as receiving a stream of multi-dimensional keys (the identifying addresses, ports, timestamps) each associated with a weight (the size of the flow). The sample enables estimating the total weight associated with an arbitrary subset of keys. This primitive is the basis of higher-lever analysis: extracting order statistics (quantiles), heavy hitters, patterns and trends. The estimates produced from the samples are unbiased, have low variance, and are subject to well-understood exponential tail bounds (Chernoff bounds). This ensures that they have good properties for the composition of several samples and queries, and so the relative estimation error decreases for queries that span multiple samples or larger subsets.

There are many other advantages to working with sampled data: the sampled keys are drawn from the original data (so, for example, within a subnetwork identified as generating an unusual volume of traffic, full details of typical flows from this subnet are captured), and these keys can be further examined or analyzed as needed. The sample itself resembles a smaller (reweighted) snapshot of the original data, and so can be directly manipulated with existing tools.

But a limitation of existing techniques is their failure to use the inherent *structure* that is present in the space of the keys. In our network data example there is significant structure in several forms. There is *order* across the timestamps, durations, and other attributes which have a natural total order. There are multiple *hierarchies*: on geographic locations, on network addresses, and on time values. Lastly, the keys are formed as the *product* of multiple attributes, each of which has certain (one-dimensional) structure. This structure fundamentally determines the kind of queries which are most important to the users: the class of *range queries* are those which are structure respecting, where the ranges correspond to contiguous keys in the ordered case, nodes in the hierarchy, or products of ranges in the multi-dimensional case (i.e. axis-parallel boxes, or hyper-rectangles).

Almost all queries posed to data are range queries, or collections of ranges. But existing stream sampling techniques are completely *oblivious* to this structure! They treat each key independently, and so consider any query as a *subset-sum* query: an arbitrary collection of keys. The guarantees for these samples assume that any subset is equally likely. But, on one-dimensional data, there are exponentially many possible subsets, while at most quadratically many of these are the more important ranges. By assuming uniformity over this larger space of queries, existing methods are missing the opportunity to optimize for the more common case of range queries.

The prior work on sampling has shown how to draw a sample from a stream of keys to build unbiased estimators with low variance. For particular classes of queries, such as subset-sum queries, the goal is *variance optimality*: achieving variance over the queries that is provably the smallest possible for any sample of that size. Classic sample-based summaries are based on Poisson sampling, where keys are sampled independently. Choosing inclusion probabilities which are proportional to the weight of a key (IPPS) [12] and using Horvitz-Thompson estimators [13] is known to minimize the sum of per-key variances. "VarOpt" summaries [2, 18, 5] improve on Poisson sampling by having a fixed sample size and better accuracy on subset-sum queries. SVarOpt [5] efficiently computes VarOpt summaries over a data stream and is a weighted generalization of reservoir sampling [19].

Classic sample-based summaries are optimized for the uniform workload of arbitrary subset-sum queries, and so do not adapt to the structure in data. Instead, for specific applications, such as (hierarchical) heavy hitters and quantiles, tailored solutions have been proposed [1, 10, 20]. For example, the popular Q-digest gives deterministic guarantees for range queries, with error bounded by a constant fraction of the total weight of all keys [15]. Such summaries are less flexible than samples, since they target a particular goal. Since they are not unbiased, combining the information for multiple subranges (as happens for HHHs) quickly loses accuracy. They cannot support non-range queries, and do not provide any "representative" keys from the input. Lastly, adapting these summaries to multiple dimensions is complex, and the size tends to grow exponentially with the dimensionality.

In the context of the IP flow example, deterministic summaries taken with respect to source IP addresses work well for capturing the volume under all sufficiently large prefixes. But if we are interested in a union of prefixes (say, associated with a certain geographic location or type of customer) or in gleaning from daily summaries the total monthly volume of a certain prefix, errors add up and the relative accuracy deteriorates. In contrast, under unbiased sample-based summaries, the relative error can decrease. For similar reasons, in estimating the total amount of traffic of many low-volume addresses, deterministic summaries can perform poorly with no useful guarantee, while the unbiased sample promises a good estimate if the total volume is large enough.

In recent work, we introduced offline structure-aware VarOpt sampling, as an improvement to structure-oblivious VarOpt that boosts accuracy on ranges. Our structure-aware summaries are built on a choice of VarOpt sample distributions in the offline setting to construct VarOpt samples with better accuracy on ranges [4]. Because the samples are VarOpt, they retain the desirable qualities of traditional sample-based summaries: unbiasedness, tail bounds on arbitrary subset-sums, and support for representative samples.

Our aim in this paper is to considerably extend this line of work,

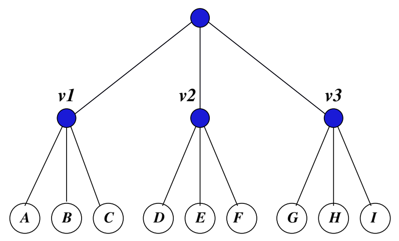


Figure 1: Hierarchy structure example.

to produce methods for structure-aware stream sampling. This requires substantially different techniques to any that have been proposed before. Stream sampling algorithms are constrained in that the sample must include at most k keys from the prefix of the stream seen so far. When a new key is added to the sample, one key must be discarded. We refer to this (randomized) action by the stream sampling algorithm as a *pivot* step. SVarOpt turns out to be inherently inflexible—there is a unique pivot at each step that results in a VarOpt sample [2, 5]. Thus in contrast to the offline setting, it is not possible to design stream sampling algorithm that compute VarOpt and structure-aware samples. Instead, we describe how to relax our requirements and find samples that are approximately VarOpt.

The next example illustrates the limitations of existing sampling methods for range queries, and the potential for a better solution.

Example 1. *Consider the hierarchy structure depicted in Figure 1. There are 9 leaf nodes (keys) with unit weights. In a hierarchy, the ranges correspond to the leaf descendants of any internal node: here, they are the sets $v_1 = \{A, B, C\}$, $v_2 = \{D, E, F\}$, and $v_3 = \{G, H, I\}$, as well the individual leaves themselves. We may also be interested in querying other subsets, such as $\{A, E, H\}$, based on some selection criteria that is not captured by the structure, but such queries are considered less likely.*

When the desired sample size is 3, since keys have uniform weights, we include each key with probability $1/3$. The summary S includes the sampled keys and associates an adjusted weight *of 3 with each key – to estimate the weight of a query set Q, we add up the adjusted weights of keys in $S \cap Q$. In our example, the estimate is $3|Q \cap S|$. The expected number of samples from each of v_1, v_2, v_3 is 1. When there is exactly one sample, the estimate is exact.*

Poisson sampling (which for unit weights is Bernoulli sampling) picks each leaf independently. The sample size $|S|$ is Binomial, with mean 3, and clearly, has no relation to the structure. The probability that S will include exactly one key from v_1, yielding an exact value of 3 for the estimate on the weight of v_1, is $4/9$.

Reservoir sampling [19] (which in our case of uniform weights is the same as SVarOpt*), uniformly selects a subset of size 3. The probability that v_1 contains exactly one sample is the same as for any other subset of size 3, i.e. $15/28$.*

A deterministic summary such as Q-digest associates values with both leaf and internal nodes which allow to obtain estimates with given absolute error. When ranges are combined, errors add up.

There is an optimal structure-aware VarOpt *sample distribution for any hierarchy [4]—the number of samples at each internal node is between the floor and ceiling of its expectation. Since the sample is* VarOpt*, estimates on arbitrary queries are at least as good as with Poisson sampling in terms of variance and tail bounds. In our example, one key is selected uniformly from each of v_1, v_2, and v_3, yielding a sample that is optimal for ranges – exact estimates of 3 on the weights of v_1, v_2, and v_3. As explained above, however, this sample can not be realized under stream constraints.*

1.1 Our Contributions

We present a class of stream sampling schemes with a parametrized tradeoff between accuracy on ranges and tightness of tail bounds on arbitrary subsets. VarOpt summaries form one end of this tradeoff, being optimal for arbitrary subset-sums but structure-oblivious.

The first component of our solution is the introduction of *weight-bounded summaries*, which generalize VarOpt summaries via a tunable *tightness parameter*. This generalization facilitates a choice of pivots in stream summarization. The tightness parameter controls a tradeoff: the choice of pivots increases with tightness parameter while accuracy on arbitrary subset queries degrades. Specifically, the degradation is tightly controlled: we establish analytically that variance and tail bounds of weight-bounded summaries are at least as good as those of VarOpt summaries of a smaller sample size. The size of that smaller sample is determined by the value of the tightness parameter. *In the context of Example 1, a weight-bounded summary, just like* SVarOpt *summary, contains* 3 *keys and the expected value of each adjusted weight is* 1*. Adjusted weights values, however, may vary, but may not exceed a value determined by the tightness parameter.*

The second component of our solution is methods and heuristics for *structure aware* choice of pivots. We propose various local optimality criteria for this selection. For applications where each new update must be processed quickly, we present fast heuristics with a more limited search space for the best structure-aware step.

Practicality. We perform an experimental study of the tradeoffs between performance on ranges and on arbitrary subsets of our structure-aware summaries and various heuristics. We find that our new structure-aware summaries can be dramatically more accurate than SVarOpt or tailored deterministic summaries (Q-digest) on range queries. This makes the new approach highly suited for summarizing large streaming data. On arbitrary subset queries, which are poorly supported by deterministic summaries, the accuracy is similar to that of VarOpt and exceeds theoretical guarantees.

Outline. Section 3 introduces *weight-bounded summaries*, and Section 3.2 presents an iterative template algorithm for computing weight-bounded summaries, where in each step a subset $X \subset S$ of the keys is selected and we sample out one key out of X. We refer to X as the *pivot set*. Each pivot choice is associated with some adjusted weight value, and we are restricted to pivots where this value is below some weight bound.

In Section 4 we define a *range-cost* measure of each pivot choice X, which captures average variance over ranges resulting from pivoting on X. Since finding a pivot with minimum range cost can be computationally intensive, we propose in Section 5 various efficient heuristics for pivot selection.

Finally, Section 6 contains an experimental evaluation, which shows the effectiveness of range-aware stream sampling, both for the basic problem of range queries, and in its application to more complex data mining.

2. PRELIMINARIES

We review some central concepts from the sampling literature. Given a set of keys $[n]$, where each key $i \in [n]$ has weight w_i, a *sample-based summary* is a random subset of keys $S \subset [n]$, together with an *adjusted weight* a_i assigned to each sampled key $i \in S$ (we define $a_i \equiv 0$ for $i \notin S$). Using the summary, the estimate of the weight $w_J = \sum_{i \in J} w_i$ of a subset J of keys is $a_J = \sum_{i \in J} a_i = \sum_{i \in S \cap J} a_i$, i.e., the sum of the adjusted weights of sampled keys that are members of J. In particular, a_i is an estimate of w_i. We use adjusted weights which are unbiased estimators of the original weights: for all i, $\mathsf{E}[a_i] = w_i$. Hence, from linearity of expectation, for all subsets J, $\mathsf{E}[a_J] = w_J$. Unbiasedness is important when estimates are combined (such as aggregating across time periods or measurement points). If the combined estimates are independent or non-positively correlated, the relative error on the sum estimate decreases.

Stream Summarization: The input is an (unordered) stream of distinct[1] keys with weight values $(w_1, w_2, \ldots)$ (key labels are identified with their position in the stream). The algorithm maintains a summary of $(w_1, \ldots, w_i)$ containing at most k keys from $[i]$: when processing key i, the algorithm computes a size-k summary of $(w_1, \ldots, w_i)$ using w_i and the summary of $(w_1, \ldots, w_{i-1})$.

Inclusion Probability Proportional to Size (IPPS) [12]: A weighted sampling method where the inclusion probability of each key in the sample is proportional to its weight, but truncated as not to exceed 1. Formally, when defined with respect to a parameter $\tau > 0$, the inclusion probability of i is $p_i = \min\{1, w_i/\tau\}$. The (expected) size of the sample is $k = \sum_i p_i$. The relation between k and $\tau \equiv \tau_k$ is expressed by the equality

$$\textstyle\sum_i \min\{1, w_i/\tau_k\} = k \ . \tag{1}$$

The value of τ_k on the observed part of the stream can be maintained by an algorithm which uses $O(k)$ storage: The algorithm adjusts τ_k upwards on the go, maintaining all observed keys with $w_i \geq \tau_k$ in a heap (there are at most k such keys), and tracking the total weight of all other keys. From this information, we can also compute the value $\tau_{k'}$ for any $k' < k$.

The Horvitz-Thompson (HT) **estimator** [13] assigns the adjusted weight $a_i = w_i/p_i$ to a key $i \in S$ with inclusion probability p_i. HT adjusted weights are optimal for the particular inclusion probability p_i in that they minimize the variance $\mathsf{Var}[a_i]$ over all assignments of adjusted weights.

Under IPPS, the HT adjusted weight of $i \in S$ is τ if $w_i \leq \tau$ and w_i otherwise. The variance is $\mathsf{Var}[a_i] = w_i^2(1/p_i - 1) \equiv w_i(\tau - w_i)$ if $w_i \leq \tau$ and 0 otherwise. HT adjusted weights with IPPS inclusion probabilities are optimal in that they minimize the sum $\sum_i \mathsf{Var}[a_i]$ of per-key variances for a given expected sample size, over all sample-based summaries.

Poisson sampling is such that inclusions of different keys are independent. With HT adjusted weights we have $\mathsf{Var}[a_J] = \sum_{i \in J} \mathsf{Var}[a_i]$ for any subset J. A Poisson IPPS sample of a expected size k can be computed by a simple stream algorithm. A drawback of Poisson sampling is that the sample size varies, whereas we would like to keep it fixed to make the best use of the space available.

VarOpt **summaries** [2, 18, 5] use IPPS inclusion probabilities (i.e. $p_i = \min\{1, w_i/\tau\}$) and HT adjusted weights (i.e. for $i \in S$, $a_i = \max\{w_i, \tau\}$). The underlying sample distribution meets the VarOpt criteria:

(a) The sample size is *exactly* $k = \sum_{i \in [n]} p_i$.

(b) *High-order inclusion and exclusion probabilities are bounded by products of first-order probabilities*: for any $J \subseteq [n]$,

$$\text{(I):} \qquad \mathsf{E}[\prod_{i \in J} X_i] \quad \leq \quad \prod_{i \in J} p_i$$

$$\text{(E):} \qquad \mathsf{E}[\prod_{i \in J}(1 - X_i)] \quad \leq \quad \prod_{i \in J}(1 - p_i)$$

where X_i is the indicator variable for i being included in the sample: $X_i = 1$ if $i \in S$ and $X_i = 0$ otherwise. The product $\prod_{i \in J} X_i$

[1]Our analysis and evaluation assume that keys are distinct but this requirement can be waved in our algorithms.

is the joint inclusion probability of all keys $i \in J$ and symmetrically, $\prod_{i\in J}(1 - X_i)$ is the joint exclusion probability.

Poisson sampling satisfies (b) (with equalities) but does not satisfy (a). VarOpt improves over Poisson IPPS by fixing the size of the summary. When τ_k is fixed, the variance of any subset-sum estimate of a VarOpt summary is at most that of a Poisson IPPS summary. This is because for a VarOpt summary, for all subsets J, $\mathsf{Var}[a_J] \le \sum_{i\in J} \mathsf{Var}[a_i]$, which is implied by the joint inclusion property for subsets of size 2 (equivalently, covariances are non-positive) whereas with Poisson IPPS, $\mathsf{Var}[a_J] = \sum_{i\in J} \mathsf{Var}[a_i]$.

VarOpt summaries are optimal for arbitrary subset-sum queries in that for *any* subset size, they minimize the expected variance of the estimates [17, 5]. This follows from VarOpt minimizing both $\sum_{i\in[n]} \mathsf{Var}[a_i]$ (by using IPPS) and $\mathsf{Var}[\sum_{i\in[n]} a_i] = 0$, and from [17] which established that the expected variance depends only on these two quantities. The notation VarOpt_k denotes a VarOpt summary of size k.

Stream Sampling. The SVarOpt method [5] efficiently computes a VarOpt_k summary of a data stream of weighted keys. The algorithm maintains a VarOpt_k summary S of the processed prefix of the stream, which we can view as a vector a with entry $a_i > 0$ if and only if $i \in S$. For each new key n, its adjusted weight is initialized to its weight value $a_n \leftarrow w_n$. The vector a now has $k + 1$ positive entries, for keys $S \cup \{n\}$. We apply $a \leftarrow \text{VarOpt}_k(a)$ to obtain a summary of size k. In sampling terms, one of the $k + 1$ keys in $S \cup \{n\}$ is ejected from the sample and weights of remaining keys are adjusted. We call the operation of VarOpt sampling out of a single key a *pivot*. The pivot performed by SVarOpt is $\text{pivot}(a, S \cup \{n\})$ (Algorithm 1).

While VarOpt_k is a family of sample distributions satisfying some constraints, SVarOpt is the unique way to maintain a VarOpt_k sample of the prefix of the stream – the pivot $\text{pivot}(a, S \cup \{n\})$ is the only way to obtain a VarOpt_k summary given a new key and a VarOpt_k summary of previously-seen keys. When keys have uniform weights, SVarOpt is isomorphic to reservoir sampling [19] and the sample distribution is a uniform selection of a k-tuple.

Chernoff bounds. Given a (query) subset J and a Poisson or VarOpt sample S the size of $J \cap S$, the number of samples from J satisfies exponential tail bounds [3, 14, 16, 11, 7]. Let $X_J = \sum_{i\in J} X_i$, where X_i is the indicator variable for $i \in S$ as before. When inclusion probabilities are IPPS and we use HT adjusted weights, Chernoff bounds imply bounds on a_J: Observe that it suffices to consider J such that $\forall i \in J, p_i < 1$ (as we have the exact weight of keys with $p_i = 1$). For such a J, the estimate is $a_J = \tau X_J = \tau|J \cap S|$. We bound a_J by substituting $X_J = a_J/\tau$ into the basic form of Chernoff bounds, and obtain

$$\Pr[a_J \le v],\ \Pr[a_J \ge v] \le e^{(v-w_J)/\tau}(w_J/v)^{v/\tau}\ . \qquad (2)$$

3. WEIGHT-BOUNDED SUMMARIES

3.1 WB Summary Definition and Properties

We introduce the concept of weight-bounded (WB) summaries which facilitates our streaming algorithms. Weight-bounded summaries generalize VarOpt summaries by replacing the function τ_k with a (larger) value M. This allows us to bound the accuracy relative to a smaller VarOpt summary of size k^* while giving more freedom to choose which elements to retain at each step.

Definition 1 (Weight-bounded summary). *A random weight vector $a \ge 0$ is an M-*bounded summary *of weight vector $w = (w_1 \ldots w_n) \ge 0$ if:*

Algorithm 1 $\text{pivot}(a, X)$

```
Require: $|X| \ge 2$
1: $A \leftarrow \text{Cand}(a, X)$
2: $M \leftarrow M(a, X) := \frac{\sum_{i\in A} a_i}{|A|-1}$
3: Select $i \in A$ with probability $q_i = 1 - \frac{a_i}{M}$
                                   ▷ IPPS probability for excluding $i$
4: $a_i \leftarrow 0$;
5: for all $j \in A \setminus \{i\}$ do
6:     $a_j \leftarrow M$
7: end for
8: return $a$
```

1. $\forall i \in [n],\ \mathsf{E}[a_i] = w_i$.
2. $\sum_{i\in[n]} a_i = \sum_{i\in[n]} w_i$
3. $\forall i \in [n]$, if $w_i \ge M$ then $a_i \equiv w_i$, otherwise $a_i \le M$.
4. For any $J \subseteq [n]$ and an upper bound $\overline{M} \ge \max_{i\in J} a_i$ (over all possible outcomes).

$$\text{(I):}\qquad \mathsf{E}[\prod_{i\in J} a_i] \le \prod_{i\in J} w_i \qquad (3)$$

$$\text{(E):}\qquad \mathsf{E}[\prod_{i\in J}(\overline{M} - a_i)] \le \prod_{i\in J}(\overline{M} - w_i) \qquad (4)$$

The size *of a weight-bounded summary is said to be k if there are k non-zero weights in it, i.e. a set of keys S with $|S| = k$.*

A VarOpt_k summary is a τ_k-bounded summary where a has exactly k positive entries, so all positive entries in a with $w_i < \tau_k$ have $a_i = \tau_k$. From Section 2, τ_k for a given set of input keys and weights is the threshold such that the corresponding IPPS probabilities sum to k.

The property of being a weight-bounded summary is transitive:

Lemma 2. *If $a^{(1)}$ is an $M^{(1)}$-bounded summary of $a^{(0)}$ and $a^{(2)}$ is an $M^{(2)}$-bounded summary of $a^{(1)}$, then $a^{(2)}$ is an M^*-bounded summary of $a^{(0)}$, where $M^{(*)}$ is the maximum value of $M^{(1)}$ and $M^{(2)}$ over all outcomes.*

Let $k^*(M)$ be the maximum k' such that $\tau_{k'} \ge M$. In other words, M is approximately the threshold that would give an VarOpt sample of size $k^*(M)$. We show that the quality of an M-bounded summary of size $k > k^*(M)$, in terms of tail bounds and average variance measures, is at least that of a $\text{VarOpt}_{k^*(M)}$ summary. Average-case variance is determined by $V\Sigma = \mathsf{Var}[\sum_i a_i]$ and $\Sigma V = \sum_i \mathsf{Var}[a_i]$. Like VarOpt summaries, weight-bounded summaries have optimal $V\Sigma = 0$, and therefore it suffices to bound ΣV. We state the results below; full proofs are in the Appendix.

Theorem 3. *For any key i, the variance in the adjusted weight of i under an M-bounded summary is at most the variance under $\text{VarOpt}_{k^*(M)}$.*

Theorem 4. *For any subset J of keys, the estimate a_J of w_J under an M-bounded summary satisfies the tail bounds obtained for $\text{VarOpt}_{k^*(M)}$.*

In the evaluations we find that in practice we get much smaller errors than suggested by these worst-case bounds. When $k^*(M) = k/2$, the performance is closer to VarOpt_k than to $\text{VarOpt}_{k^*(M)}$. This is because the adjusted weight of most included keys in an M-bounded summary is usually much smaller than M.

3.2 Computing WB summaries

We first define an (offline) iterative randomized process that manipulates a weight vector a, initialized with the original weight vector w. The number of non-zero entries decreases by one each iteration—after $(n-k)$ iterations we have a summary of size k. In each iteration we select a subset $X \subset [n]$ of the positive entries, and choose one of these to remove. Essentially, we treat X as a set of weighted keys and apply the IPPS procedure to X to compute a VarOpt summary of size $|X|-1$. We later discuss criteria for choosing X based on structural considerations.

Candidate Set and Pivot Threshold. Recall from Section 2 that for any set of weighted keys there is a unique threshold τ_k which produces a sample of size k. Applying IPPS to the set of keys X with associated weights a, the threshold is $\tau_{|X|-1}(M)$ for X, which we call the *pivot threshold*, $M(a,X)$. This can be computed efficiently as described in Section 2. We then define the *candidate set*, $\textsc{Cand}(a,X)$ as the (unique) subset of X which are candidates for ejection (under IPPS with threshold $M(a,X)$: $\textsc{Cand}(a,X)$ is the subset of keys with weight $a_i < M(a,X)$, that is, they have retention probability < 1. The remaining keys in X have sufficient weight to ensure that they will not be removed in this pivot step. Observe that $\textsc{Cand}(a,X)$ is of size at least 2 when $|X| \geq 2$ since the two smallest weight keys are always candidates for ejection.

PIVOT **Algorithm.** To perform the pivot, we use the pivot threshold to sample an element to eject from the candidate set. This is implemented in Algorithm 1: $\textsc{pivot}(a,X)$. The algorithm computes a VarOpt summary of X of size $|X|-1$. One can verify that the exclusion probabilities in line 3 sum to 1. After the pivot operation, one of the candidate keys has adjusted weight 0 and the others have their new adjusted weight set to $M(a,X)$, which is equal to $\frac{\sum_{i\in \textsc{Cand}(a,X)} a_i}{|\textsc{Cand}(a,X)|-1}$.

LEMMA 5. *The output of* $\textsc{pivot}(a,X)$ *is an* $M(a,X)$*-bounded summary of* a.

Consider such a process with input weight vector $a^{(0)} = w$. For $\ell \geq 1$, the ℓth iteration has input $(a^{(\ell-1)}, X^{(\ell-1)})$ —a weight vector and a pivot set—and outputs weight vector $a^{(\ell)}$. Let $M^{(\ell)} = M(a^{(\ell-1)}, X^{(\ell-1)})$. Define $\overline{M^{(\ell)}}$ to be the maximum of $\max_{h\leq \ell} M^{(h)}$ in all possible outcomes of the first ℓ iterations. From Lemma 2 and Lemma 5, we obtain:

THEOREM 6. $\forall \ell$, $a^{(\ell)}$ *is an* $\overline{M}^{(\ell)}$*-bounded summary of* w.

3.3 Stream WB summarization

In the streaming setting, we must ensure that the total number of stored keys at any time is small enough to fit in the available memory. The keys that must be stored when the ℓth element is processed are those amongst the first ℓ elements (the elements seen so far) that have positive adjusted weights.

Therefore for stream weight-bounded summarization, we apply a pivot operation after each arrival following the kth stream element. The ℓth pivoting iteration is performed after the $k+\ell$th stream arrival and the stored keys are $S^{(\ell)} = \{i | a_i^{(\ell)} > 0 \wedge i \leq \ell + k\}$. As each iteration sets the adjusted weight of one key to zero, the total number of stored keys is kept to k.

Algorithm 2 takes as input a data stream with weights $w_1, w_2, \ldots$ and a *tightness parameter* $c \geq 1$. This parameter controls the flexibility with which the algorithm can deviate from strict VarOpt and is described in more detail below. The algorithm maintains a set S of (at most k) keys and adjusted weights a_S. It has the property that at any point, a is a weight-bounded summary of the processed prefix of the data stream:

Algorithm 2 $\textsc{StreamWBSummary}(c, w_1, w_2, \ldots)$

```
1: $S \leftarrow \emptyset$                                        ▷ Initialize
2: for $i = 1, \ldots$ do                                          ▷ Process key $i$
3:     $S \leftarrow S \cup \{i\}$
4:     $a_i \leftarrow w_i$
5:     if $i > k$ then
6:         $a \leftarrow \textsc{pivot}(a, X)$, for some $X \subset S$ such that:
           • $X$ satisfies some structure-aware selection criteria
           • $M(a,X) \leq \tau_{k/c}(S)$;
7:         $S \leftarrow \{i : a_i > 0\}$
8:     end if
9: end for
```

LEMMA 7. *Defining* $a_i \equiv 0$ *for* $i \notin S$*, vector* a *is a* $\tau_{k/c}(w_1,\ldots,w_\ell)$*-bounded summary of* $(w_1,\ldots,w_\ell)$

The smaller c is, the less flexibility we have in choosing the pivot set X. Since τ_k is decreasing in k, as c decreases, $\tau_{k/c}$ decreases, providing less scope for choosing X. At the extreme, if $c = 1$, every X with $M(a,X) \leq \tau_k$ must include all of $\textsc{Cand}(a,S)$ and the algorithm becomes identical to SVarOpt [5]. SVarOpt also minimizes the "local" variance cost, which is the variance of $a^{(\ell)}$ with respect to $a^{(\ell-1)}$:

$$\Sigma V(X) \equiv \sum_{i\in X} \mathsf{Var}[a_i^{(\ell)}] = \sum_{i\in X} a_i^{(\ell-1)} \left(\frac{\sum_{i\in X} a_i^{(\ell-1)}}{|X|-1} - a_i^{(\ell-1)} \right). \quad (5)$$

When we set $c > 1$ (e.g. to a small constant such as 2), we have greater freedom in choosing the pivot set X meeting the constraints on $M(a,X)$. We will use this flexibility to make a "structure-aware" selection (we formalize this notion in the next section). That is, we can choose a subset of keys on which to pivot such that the keys are "close" under the assumed structure (e.g. close in the hierarchy). This way, by keeping the necessary shifting of weight due to pivoting localized, crossing fewer range boundaries, we are more likely to end up with a summary such that the adjusted weight of ranges is closer to their actual weight. Figure 2 compares the actions of SVarOpt (which is structure oblivious) and StreamWBSummary (with an intuitive structure-aware pivot selection) on a stream of keys from the example hierarchy structure of Figure 1.

An alternative to constraining pivot selection is to compute the *effective* tightness for a chosen pivot, i.e., the value of c that yields equality in line 6 of Algorithm 2. Tail bounds for estimation are then governed by the maximum effective tightness over the stream.

4. RANGE COST

In this section, we formalize "structure-awareness" by associating a "range cost" with each possible choice, and striving to select a pivot with low range cost. Our *range cost* $\rho(X)$ of a pivot X measures the variance local to the iteration, averaged over "range boundaries." By local we mean that we consider the variance of the adjusted weights a' at the end of the iteration with respect to the adjusted weights a that are the input to the iteration. This local view follows from restrictions imposed by the streaming context and is inspired by SVarOpt, where the selected pivot locally minimizes the "structure oblivious" variance ΣV (5).

For a range $R \in \mathcal{R}$, the change in its estimated weight following a pivot step is the random variable $\Delta_R = |a'_R - a_R|$. Because both sets of weights are unbiased, $\mathsf{E}[\Delta_R] = 0$ and the variance of this change is $\mathsf{E}[\Delta_R^2]$. Generally, there is no one pivot which simultaneously minimizes variance for all ranges in $\mathcal{R}$. Next we

SVarOpt:

	A	D	E	G	B	C	H	I	F	τ
3	1	1	1							1
4	$\frac{4}{3}$	0	$\frac{4}{3}$	$\frac{4}{3}$						$\frac{4}{3}$
5	0	0	$\frac{5}{3}$	$\frac{5}{3}$	$\frac{5}{3}$					$\frac{5}{3}$
6	0	0	2	2	2	0				2
7	0	0	$\frac{7}{3}$	$\frac{7}{3}$	0	0	$\frac{7}{3}$			$\frac{7}{3}$
8	0	0	$\frac{8}{3}$	$\frac{8}{3}$	0	0	$\frac{8}{3}$	0		$\frac{8}{3}$
9	0	0	3	3	0	0	3	0	0	3

StreamWBSummary:

	X	A	D	E	G	B	C	H	I	F	τ
3		1	1	1							1
4	$\{D, E\}$	1	0	2	1						$\frac{4}{3}$
5	$\{A, B\}$	0	0	2	1	2					$\frac{5}{3}$
6	$\{B, C\}$	0	0	2	1	3	0				2
7	$\{G, H\}$	0	0	2	0	3	0	2			$\frac{7}{3}$
8	$\{H, I\}$	0	0	2	0	3	0	3	0		$\frac{8}{3}$
9	$\{E, F\}$	0	0	3	0	3	0	3	0	0	3

Figure 2: Selecting a sample of size 3 from a stream of 9 unit weight keys. Each row corresponds to a new arrival (starting with the 4th arrival). The table entries are adjusted weights at the end of the iteration. The rightmost column shows the IPPS threshold τ_3 on the prefix of the stream. The tables show possible execution of the randomized algorithms. SVarOpt (top): The pivot set is always of size 4, containing the 3 participants in the current summary S and the new arrival. The final summary contains the keys $S = \{E, G, H\}$, all with adjusted weights 3. StreamWBSummary with $c = 3/2$ (bottom): According to the structure (Figure 1), we prioritize pivot sets with lower LCA. This pivot selection rule is intuitively structure aware as it tends to preserve the weight under internal nodes. In this particular case, all outcomes constitute an optimal structure-aware sample, containing exactly one sample from each of v_1, v_2, and v_3.

propose ways to measure the overall impact of a pivot in different types of structure, and to pick pivots with minimum impact.

4.1 Partition

When the structure is a partition of keys, the ranges are parts in this partition and are disjoint. We define range cost as the sum of variances over individual ranges:

$$\rho(X) = \sum_{R\in\mathcal{R}} \mathsf{E}[\Delta_R^2] = \mathsf{E}[\sum_{R\in\mathcal{R}} \Delta_R^2] \ .$$

Defining $L_R = \sum_{i\in \textsc{Cand}(X)\cap R} a_i - M(a,X)\lfloor \frac{\sum_{i\in \textsc{Cand}(X)\cap R} a_i}{M(a,X)} \rfloor$, the portion of the weight in range R that will be rounded up to $M(a,X)$ or down to 0 when pivoting on X, we get

$$\rho(X) = \sum_{R\in\mathcal{R}} L_R(M(a,X) - L_R) \ .$$

In the degenerate case when each range contains a single key, $\rho(X) \equiv \Sigma V(X)$ (see (5)) and therefore minimizing the range cost is equivalent to VarOpt sampling.

For the general case, $\rho(X) = 0$ if and only if $X \subset R$ for some range R (i.e. X is fully contained in some range R). Between pivot choices with $\rho(X) = 0$, we propose to use a pivot with minimum $M(a,X)$, which must be of the form $S \cap R$ for some range R. In summary, we propose to pick $X = S \cap R$ with minimum $M(a,X)$ if $M(a,X) \leq \tau_{k/c}$. Else, pick the whole sample S as the pivot X.

4.2 Order

When there is a natural total order over the keys in the data, we define the range cost to be a weighted average of the variance over prefixes (a.k.a. 1-d halfspaces). Apropriate weighting of prefixes prevents regions with many small weights from dominating those with few larger weights. Considering the linearly many prefixes rather than the quadratically many intervals simplifies analysis and prevents smaller intervals from dominating fewer larger ones. As the weight of an interval (range of contiguous keys) is the difference of weights of two prefixes, the (additive) estimation error on an interval is at most the sum of estimation errors on two prefixes.

To analyze the range cost, we study how a pivot changes the distribution of weights in the sample. Formally, for $0 \leq q \leq 1$, the *q-quantile point* of the sample S is the prefix of ordered points that contains a q fraction of the total adjusted weight, a_S. Key i from the ordered universe has adjusted weight a_i, and if $i \in S$, this key covers a range of quantile points. We write i_q to denote the index of the key which includes the q-quantile, i.e. the index i that satisfies

$$q \in \left(\frac{\sum_{j<i} a_j}{a_S}, \frac{a_i + \sum_{j<i} a_j}{a_S}\right] \ .$$

For a subset X and q, let Δ_q be the random variable corresponding to the (absolute value) of the weight that "moves" across the half-space induced by i_q as a result of $a' \leftarrow \textsc{pivot}(a, X)$.

If $i_q \notin X$, then this is just the difference in weight below the key before and after the pivot:

$$\Delta_q = \left| \sum_{i\in S|i<i_q} a_i - \sum_{i\in S|i<i_q} a'_i \right| \ .$$

If $i_q \in X$, we treat the distribution as a continuous one with the weight of each key spread across an interval (the cost is invariant to the size of the interval). The fraction of the weight a_{i_q} of i_q that lies "below" the quantile point is

$$\delta_q = \frac{1}{a_{i_q}}(q a_S - \sum_{i\in S|i<i_q} a_i) \ ,$$

and so the change in weight across q is

$$\Delta_q = \left| \left(\delta_q a_{i_q} + \sum_{i\in S|i<i_q} a_i\right) - \left(\delta_q a'_{i_q} + \sum_{i\in S|i<i_q} a'_i\right) \right|$$

The range cost is defined as the sum (integral) of variances $\mathsf{E}[\Delta_q^2]$ across all half-spaces, i.e. all quantiles $0 \leq q \leq 1$ of S.

$$\rho(X) = \mathsf{E}[\int_0^1 \Delta_q^2 dq] \equiv \int_0^1 E[\Delta_q^2] dq \ .$$

In the full version of this paper, we show that this cost is minimized when the pivot set X is a pair of keys (i.e. $X = \{i_1, i_2\}$) and that the range cost is

$$\rho(\{i_1, i_2\}) = a_{i_1} a_{i_2} (\tfrac{1}{3}(a_{i_1} + a_{i_2}) + a_{S_M}) \ , \qquad (6)$$

where $S_M = \{i : i_1 < i < i_2\}$ is the subset of all keys in S that lie strictly between i_1 and i_2.

The range cost can be computed efficiently: Algorithm 3 presents pseudocode of an algorithm which finds $\rho(X)$ in $O(|X|)$ time. It first computes prefix sums W_j of the weights of keys in X (line 6), and values y_j which are the sums of weights between the jth and $j+1$th keys in X (line 9): computing each takes constant time, assuming that we have access to the prefix sums of weights in S. Then lines 13 and 14 use these values to compute the range cost, following our (omitted) analysis Consequently, we can find the range cost of *every* pair i_1, i_2 in time $O(|S|^2)$, and pick the best to pivot on. Section 5 discusses choices which compromise on picking the best pair to reduce this time cost.

4.3 Hierarchy

We use the above analysis of order to analyze the hierarchy case. There are many possible ways to linearize a hierarchy to produce an order. Here we consider *all* possible linearizations, and take

Algorithm 3 RANGE-COST(X)

```
Require: $|X| \geq 2$
 1: $X \leftarrow$ CAND$(a, X)$
 2: $M \leftarrow M(a, X)$
Require: $X = \{i_1, \ldots, i_{|X|}\}, i_1 < i_2 < \cdots < i_{|X|}$

 3: $W_0 \leftarrow 0$
 4: for $j = 1, \ldots, |X|$ do
 5:     $w_j \leftarrow a_{i_j}$                          ▷ weight of key $i_j$.
 6:     $W_j \leftarrow W_{j-1} + w_j$        ▷ weight of $j$ smallest keys in $X$
 7: end for
 8: for $j = 1, \ldots, |X| - 1$ do
 9:     $y_j \leftarrow \sum_{h \in S \setminus X | i_j < h < i_{j+1}} a_h$
                    ▷ Weight of keys in $S$ that are between $i_j$ and $i_{j+1}$.
10: end for
11: $R \leftarrow 0$                                      ▷ Initialize
12: for $\ell = 0, \ldots, |X| - 1$ do
13:     $R \leftarrow R + y_\ell(M\ell - W_\ell)(W_\ell - (\ell - 1)M)$
          ▷ contribution of $q$ when $i_q \in S \setminus X$ is between $i_\ell$ and $i_{\ell+1}$.
14:     $R \leftarrow R + w_{\ell+1}(\ell M - W_\ell)(W_{\ell+1} - \ell M) + \frac{w_{\ell+1}^2}{3}(M - w_{\ell+1})$
                              ▷ contribution of $q$ such that $i_q = i_{\ell+1}$.
15: end for
16: return $R/a_S$
```

the average of the range costs of the corresponding order structure. When the pivot set X is two leaf nodes u, v we use (6) and obtain

$$\rho(\{u, v\}) = a_u a_v (\tfrac{1}{3}(a_u + a_v + a_{S_M}) + \tfrac{1}{2} a_{S_C}) \,. \qquad (7)$$

To define these terms, we let $r = \text{LCA}(u, v)$ be the lowest common ancestor of u and v. Then S_M is the set of leaf nodes in the subtree of r but which share no other ancestors below r, i.e. $y \in S \setminus \{u, v\}$ such that y is a descendant of r and $\text{LCA}(u, y) \equiv \text{LCA}(v, y) \equiv r$. Any $y \in S_M$ is placed between u and v in exactly a $1/3$ fraction of all possible linearizations. S_C corresponds to the remaining leaf nodes in the same subtree of r: the keys $y \in S \setminus \{u, v\}$ such that y is a descendant of $\text{LCA}(u, v)$ and either $\text{LCA}(u, y) \neq r$ or $\text{LCA}(v, y) \neq r$. In this case, the probability, over linearizations, that y lies between u and v is $1/2$.

We use these costs (7) to guide our selection of a pivot set X. Below we discuss how to do so efficiently.

4.4 Product space

Our results extend to the case when we have a multi-dimensional key space and each dimension is itself a partition, order and/or hierarchy. For such product spaces, we define the range cost as the average over dimensions of the 1-dimensional range costs. This corresponds to averaging over all axis-parallel halfspaces, treating all dimensions as equally important. The expressions for range cost rapidly become complex, and the search for an optimal pivot set at each step becomes costly, so we omit detailed analysis and instead identify fast heuristics to find pivot sets.

5. PIVOT SELECTION

In the previous section, we showed that the range cost of a subset X is well-defined and, given X, can be computed efficiently. However, naively searching over all $2^{|S|}$ subsets to find an applicable pivot $X \subset S$ with minimum range-cost is clearly prohibitive; we must find a pivot for every new key in the stream: this does not scale to high-volume streaming environments. The range cost $\rho(X)$ is non-decreasing when keys are added to X, while $M(a, X)$ is non-increasing. Thus, although we might find an X which minimizes $\rho(X)$, it may not be a valid pivot given the requirement on $M(a, X)$ in Algorithm 2. This strict upper bound on $M(a, X)$ forces us to consider a broader range of possible pivots. Here, we propose and evaluate heuristics for finding valid pivots with good range cost, and compare them empirically in our experiments.

5.1 Pair heuristics

Our analysis (details omitted for brevity) suggests that range cost on order structures is minimized for a pair of keys. A pair $\{i_1, i_2\}$ is applicable if and only if $w_{i_1} + w_{i_2} \leq \tau_{k/c}$. However, when $c \geq 2$, we can guarantee that there are always applicable pairs of keys in the sample (below, we show a stronger result, that there are always applicable pairs which are close together). Inspired by this, we propose some heuristics to choose pairs to pivot on.

Best pair: Select the applicable pair $\{i_1, i_2\}$ with least range cost.

For an order, Eq. (6) allows quick elimination in finding such a pair: to identify an applicable pair with minimum range cost, it suffices to consider pairs $i_1 < i_2$ such that

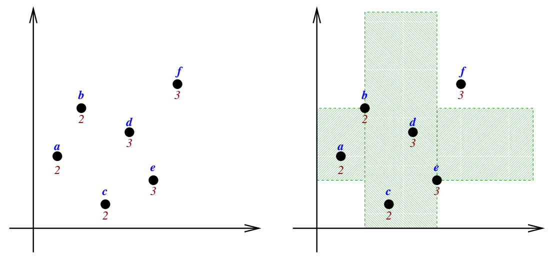
$$\forall i_3 \text{ such that } i_1 < i_3 < i_2,\ w_{i_3} > \max\{w_{i_1}, w_{i_2}\} \qquad (8)$$

To see that this is a necessary condition, consider a triple i_1, i_2, i_3 for which this does not hold. By replacing the heavier key among i_1, i_2 with i_3, we obtain an applicable pair with a lower range cost that i_1, i_2. Consequently, any range of keys in the sample must be either disjoint or in a containment relation, and so there are at most $O(|S|)$ pairs that satisfy (8). We can find them quickly by initializing the set of candidate with all adjacent pairs. We then try to extend each candidate it turn by merging with its left or right neighbor (at most one is possible). This finds all possibilities with $O(|S|)$ work.

For a hierarchy, we maintain for each internal node the total adjusted weight under the node, and the two smallest weight keys in the subtree. In general there are many nodes in the hierarchy, but we only need to monitor those subtrees with at least two keys in the sample, and can also ignore nodes where only one descendant tree is populated. On receiving each new key, we update the stored data to meet the requirements, and pivot on the pair with lowest range cost. Updating subsequent to a pivot is fast, since the cost is proportional to the depth of the hierarchy.

For a product space, it suffices to consider only those pairs such that both keys have smaller weight than any other key in the bounding box containing them. A key is in the bounding box of two keys if on order dimensions, it lies between them, and in a hierarchy dimension, it is a descendant of their LCA. In the worst case, however, there can still be $\Omega(|S|^2)$ such pairs even for $d = 2$, so other heuristics are needed to more quickly find pivots.

Simple Nearest Neighbors heuristic (SNN). Typically, the structures we consider have a natural notion of proximity: the number of intervening keys in an order, distance to a lowest common ancestor in a hierarchy. The SNN heuristic considers one pivot pair for each sampled key i, which is its nearest neighbor in the sample. This is a good heuristic, but does not necessarily find the pair with smallest range cost. Pivoting on NN pairs gives intuitive structure awareness because the movement of weight is localized. Specifically, for order we consider the neighbors of each key; for hierarchy, the key in the same subtree as i with smallest weight; for product spaces, we maintain a KD-tree and treat it as a hierarchy. When $c \geq 2$, there must exist an applicable pair X of NN keys. To see this under order, assume k is odd, and consider the pairs of keys in the sample with non-zero weights, indexed $i_1, i_2 \ldots i_{k+1}$. Consider their IPPS



pivot	M	$L(X)$	$\Sigma V(X)$	M min	ρ^*	ρ
$\{a,b\}$	4	11	8	8	88	22.67
$\{a,c\}$	4	13	8	8	104	30.67
$\{b,d,e\}$	4	20	10	8	200	43.33
$\{b,e\}$	5	20	12	10	240	80
$\{a,b,d\}$	3.5	16	7.5	7	120	22.25
$\{a,b,c\}$	3	18	6	6	108	24

Figure 3: 5 keys with structure that is the product of two orders. Right: Span of $\{b,e\}$: keys b,d,e are counted twice in the span weight, keys a,c are counted once. The span of $\{b,e\}$ and $\{b,d,e\}$ is the same and the span weight is $L(\{b,e\}) \equiv L(\{b,d,e\}) = 20$. The table shows for pivot sets the weight-bound, span weight, ΣV, $M(a,X)\min_{i\in X} a_i$, and (unnormalized, multiplied by $2a_S$) span cost and range cost.

probabilities under $\tau_{k/2}$ in adjacent non-overlapping pairs:

$$\max\{\frac{w_{i_{2j-1}}}{\tau_{k/2}},1\} + \max\{\frac{w_{i_{2j}}}{\tau_{k/2}},1\}$$

Observe that over all $(k+1)/2$ pairs, these sums total $k/2$, and hence the mean is less than 1. Therefore there must be an applicable pair of some key and its left or right NN.

Among applicable NN pairs we can select one (i) arbitrarily (ii) with minimum range cost (iii) with minimum a_1a_2 (minimizing the ΣV variance cost), or (iv) with minimum $M(a,X) = a_1 + a_2$ (automatically ensuring that $c \leq 2$). These variations have differing costs, and we compare their cost and quality experimentally.

5.2 Span cost approximation

We define the *span cost* of a pivot X as an alternative to range cost that is easier to compute. The span cost

$$\rho^*(X) = M(a,X)(\min_{i\in X} a_i)\frac{L(X)}{d\cdot a_S} \ . \tag{9}$$

is a product of two main components. The first one, the *span-weight* $L(X)$ (defined below), is structure-sensitive—the more "spread out" X is, the larger it is. It captures the halfspaces impacted by pivoting on X. The second component $M(a,X)(\min_{i\in X} a_i)$ is oblivious to the structure. It is non increasing when keys are added to X since this holds for both $M(a,X)$ and $\min_{i\in X} a_i$. It captures the maximum expected variance of the pivot over a subset of X.

We now show how we define the span-weight $L(X)$: For order structures, $L(X)$ is the weight of all keys in X and all keys in $S\setminus X$ that lie between keys in X, i.e. $L(X) = \sum_{j\in S:\min(X)\leq j\leq \max(X)} a_j$. For a hierarchy, $L(X)$ is the weight under LCA(X), i.e. $L(X) = \sum_{j\in S:\text{LCA}(j)=\text{LCA}(X)} a_j$. For a product space, it is the sum over dimensions of the 1-dimensional span weights. The normalized span weight is $0 \leq \frac{L(X)}{d\cdot a_S} \leq 1$, where d is the dimension ($d=1$ for a single order or hierarchy). It upper bounds $\frac{L(\textsc{Cand}(X))}{d\cdot a_S}$, the fraction of halfspaces impacted by pivoting on X.[2] Similarly, the range cost $\rho(X)$, defined as the average variance over halfspaces, is also upper bounded by the span cost. An example span cost computation in shown in Figure 3.

[2]However, if $L(\textsc{Cand}(X)) < L(X)$, then there is a smaller range containing $\textsc{Cand}(X)$ with span weight $L(\textsc{Cand}(X))$. This range will have the same variance component, and hence, lower span cost. It is therefore not useful to replace $L(X)$ by (the harder to compute) $L(\textsc{Cand}(X))$ in the span cost formula.

A helpful property in searching for an applicable pivot with minimum span cost is that amongst all pivots with same span $L(X)$, the span cost is minimized by the most inclusive pivot. Consequently, it suffices to consider *complete ranges*: That is, in an order, we include all keys between the least and greatest key in the pivot. For a hierarchy, we consider only pivots that corresponds to *all* descendants of a node in the hierarchy. In a product space, products of complete ranges capture all keys within a box.

Approximate Span Cost. For product spaces, where searching through all ranges can be prohibitive, we can further limit the search space by settling for approximation. We can limit the number of 1-dimension projections, considering a subset of those where a range is included only if its weight is smaller by a factor of at least $(1+\epsilon)$ from the weight of an included range that contains it. It also suffices to consider "boxes" (products of 1-d ranges) where the span weights of each dimension are within ϵ to $1/\epsilon$ factor of each other.

6. EXPERIMENTAL EVALUATION

In this section we describe our experimental study of representative structure-aware sampling methods for summarization. We consider datasets in the 1-dimensional case when the keys take values in a hierarchies, and the 2-dimensional case when keys take values in a product of two hierarchies. We explore the behavior of several heuristics described in Section 5. We compare their performance against existing approaches, namely structure-oblivious VarOpt sampling, and the deterministic Q-digest.

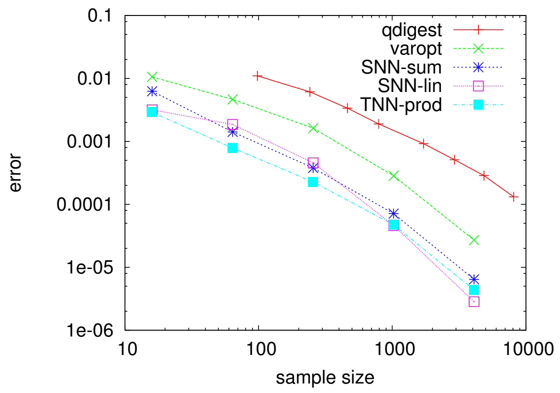
6.1 Data and Hierarchy Description

Our evaluations used IP flow statistics compiled from flows observed at a router attached to network peering point. Each flow record contains 32-bit source and destination IPv4 address, taking values in the natural binary IP address hierarchy. For the 1-dimensional experiment, we projected the data set on the source address, and used both source and destination for the 2-dimensional experiments. There were 63K sources and 50K destinations, and a total of 196K pairs active in the data. These data sets are large enough to show the relative performance of different methods but small enough to enable us to compare the accuracy to methods that are less scalable. In our experiments, the keys were the (distinct) 1- and 2- dimensional addresses, and the weights were the total recorded bytes associated with each key.

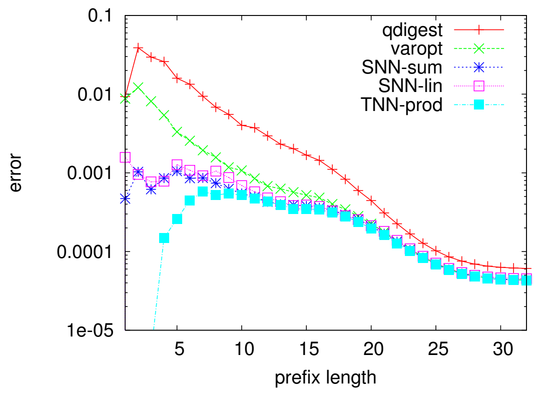
6.2 Summarization Algorithms and Heuristics

We next describe the set of summarization algorithms studied. We first present the existing algorithms used as reference in the 1-dim and 2-dim cases, then the structure-aware sampling heuristics specific to each of those cases.

In choosing efficient heuristics to work with, we aim to pick methods which are comparable to existing methods in terms of their computational costs. The cost of SVarOpt$_k$ is at worst $O(\log k)$ per insertion, and an amortized $O(\log\log k)$ implementation exists [5]. In order to be computationally competitive for large sample sizes, we aim to have algorithms which are no worse than $O(\log k)$. However, we also evaluate some heuristics with higher computational cost in order to determine the potential benefits in accuracy which might be achievable.



(a) Overall MRE as a function of sample size k.



(b) MRE as a function of prefix length for $k = 256$ samples.

Figure 4: Accuracy: 1-dimension.

6.2.1 *Existing Algorithms*

Q-digest. The Q-digest data structure [15] imposes a (conceptual) binary tree over a domain of size U, and associates counts with a subset of nodes in the tree. A top-down version of the maintenance algorithm handles an update of weight w to key i by sharing the weight among the path of nodes from root to i so that the prefix of nodes all have count exactly $\lfloor \epsilon N / \log U \rfloor$ [9]. Here, N denotes the sum of weights observed so far, and ϵ is an error parameter. The algorithm guarantees (deterministically) that the frequency of any key or node can be estimated from the stored counts with error at most ϵN. By periodically moving weight from nodes back along their path to the root (while respecting the $\lfloor \epsilon N / \log U \rfloor$ condition on weights), the algorithm guarantees that the space required is $O(1/\epsilon \log U)$ In comparison to randomized "sketch" algorithms, the Q-digest has been observed to give much more accurate answers given the same space allocation [8].

The Q-digest approach is closely related to other Heavy Hitter focused summarization, as discussed in Section 1. For example, the trie-based solution of Zhang *et al.* [20] is essentially identical, but omits the rebalancing step since it is assumed that the total weight of keys is known in advance. For this reason, we take Q-digest as the sole representative of this class of methods.

In two dimensions (i.e. $U \times U$), the data structure performs the same algorithm on the data projected on the first dimension. But in addition to updating the weight at nodes in the tree, it also keeps a secondary Q-digest data structure at each node, and updates these with the second dimension of the key and corresponding weight. From these, it is possible to estimate the weight of any rectangle with error ϵN in space $O(1/\epsilon \log^2 U)$.

VarOpt: We compared to an implementation of the structure oblivious stream sampling scheme SVarOpt_k [5]. Each key observed in the data stream is added to the sample, then one is chosen for ejection to maintain a summary size of k.

6.2.2 *Structure Aware Sampling in One Dimension*

Based on the discussion in Section 5, we compared a variety of heuristics for pivot selection. Here, we describe how they were implemented to ensure fast per-key processing.

SNN: Simple Nearest Neighbor. As described in Section 5, the SNN heuristic restricts the pivot to be a pair. In one-dimension we maintain the IP addresses present in the sample in a binary search tree (BST) ordered by treating the key as a 32-bit integer; for simplicity we allow both left and right neighbors of a given element to constitute a pair with it. (Note this method generalizes to pairing with nodes that are siblings in the IP address hierarchy of a left or right neighbor, but happen to be further away numerically). To allow quickly finding the best pair to pivot on, the range cost associated with each pair is maintained in an ascending order priority queue. The lowest cost pair is selected as the pivot at each step. We discuss specific range cost variants below. Updates to the BST are $O(\log k)$ in complexity, and updates to the priority queue have the same cost (since there are $O(k)$ NN pairs).

We consider the following SNN variants based on how we define the cost of a pair.

- **SNN-Sum:** the cost is set as the sum of the weights of the pair elements. We also tried using the product of the weights as the cost; the results were similar to SNN-SUM, and are omitted.
- **SNN-Lin:** uses the average range cost over all linearizations of the hierarchy, as given by (7). This is a way of adding further structure awareness to SNN. A pitfall of SNN described above is that an SNN pair can be quite distant in the IP hierarchy. The linearization cost penalizes pairs (x, y) that are distant in the hierarchy by including the weight of other descendants of $\text{LCA}(x, y)$.

TNN-Prod: True Nearest Neighbor. For each node we find its nearest neighbor of minimum weight, then minimize the cost over all such pairs. The pair cost was set as the product of weights, i.e., the estimation variance associated with VarOpt sampling from a pair. Note that implementing this algorithm is more costly than the SNN approaches; we include it to compare accuracy.

6.2.3 *Structure Aware Sampling in Two Dimensions*

VSNN: Very Simple Nearest Neighbor. As described in Section 5.1, IPv4 address pairs are represented as points in a KD-Tree. Ideally we would want to determine true NN pairs using distance based on the IP4v hierarchy, e.g.,

$$\mathcal{D}((s_1, d_1), (s_2, d_2)) = (32 - \text{PRE}(s_1, s_2)) + (32 - \text{PRE}(d_1, d_2))$$

were $\text{PRE}(\cdot, \cdot)$ returns the longest common prefix length. However, finding a nearest neighbor in a KD-Tree can take time linear in the number of keys stored, so we adopt a simpler approach. The VSNN of a given node is determined by finding the element in the KD-tree of minimum $\mathcal{D}$ distance, when testing all nodes (except the node itself) on the path that would be followed by a fresh insertion of the node. We then use the weight product pair cost, maintained over all

VSNN pairs in a priority queue. Similarly to the 1-dim SNN case, the complexity is $O(\log k)$ per key.

SpanApprox. This is a heuristic related to span cost in Section 5.2. In each dimension we compute internal nodes s and d in the respective spanning tree IPv4 hierarchies of the addresses of the nodes currently stored. Let S_s and D_d be the keys in the sample with source address (respectively, destination) in the subtree of s (resp., d). To choose a pivot set, then for each pair (s, d) we search for an applicable set in $S_s \cap D_d$ which induces the minimal value of the tightness parameter c in Section 3.3. The range cost associated with this set is then $W(S_s) + W(D_d)$; this is minimized over all such pairs (s, d) to select the actual pivot. This algorithm is clearly expensive computationally; we include it to compare accuracy.

6.3 Evaluation Metrics

Our principle metric for evaluating summarization accuracy is the error of the estimated weight a_J of a given node or nodes J, as compared with the actual weights w_J. This is normalized by the total weight of the whole dataset, which is equivalent to a_S. The corresponding elementary error value is $\epsilon = |a_J - w_J|/a_S$.

A given set of unsampled data is represented as leaf weights w_j. We consider the set of aggregates of these weights up the tree, i.e., $w_v = \sum_{j \in L_v} w_j$ where L_v is the set of leaf nodes descended from v. For each internal node v with non-zero actual aggregate w_v, we calculate the corresponding error ϵ_v. In order to understand the dependence of accuracy on level in the address hierarchy, we average errors of non-zero aggregates at each prefix prefix length, as the Mean Relative Error (MRE). In the two-dimensional case we consider prefixes of the same length in each of the two dimensions. Finally, we construct a global average of the length-wise error over all prefix lengths.

In the one-dimensional case, we also evaluated performance in identifying hierarchical heavy hitters (HHH). Specifically, we considered prefixes whose true aggregate weight exceeded a given threshold fraction (after discounting the weight all such prefixes under them), then calculated the Root Mean Square relative estimation error over all such nodes, over the set of runs.

6.4 Experimental Results

We extracted from our trace 10 subsets of 16,384 flows each. We employed sample sizes of $k = 64, 256, 1024$ and $4,096$, corresponding to sampling rates of 1 in $256, 64, 16$ and 4 respectively. We computed the error measured described above, averaged over the 10 runs. The algorithm implementations were constructed using Python and Perl.

One-dimensional Results. Figure 4(a) shows the global MRE in the one-dimensional case as the sample size is varied. We observe that although designed to give accurate answers for these kinds of range queries, the Q-digest approach is clearly less accurate than sampling methods on this data. This is due to the method's dependence on $\log U$, the logarithm of the domain size: in this case, it contributes a factor of 32 to the space cost. Beyond this, we still see much variation in the behavior of the sampling-based methods (note that the plot is on a log scale). SVarOpt, the state-of-the-art stream sampling method, can be improved on by up to an order of magnitude by methods which are structure-aware. Across sample sizes, the TNN-prod method consistently achieves the best (or approximately best) accuracy, since it puts more effort into finding the best NN pair to pivot on. The two SNN methods were appreciably faster to process the data, and the simplest SNN-sum method still achieves a clear improvement over SVarOpt while having low computational complexity.

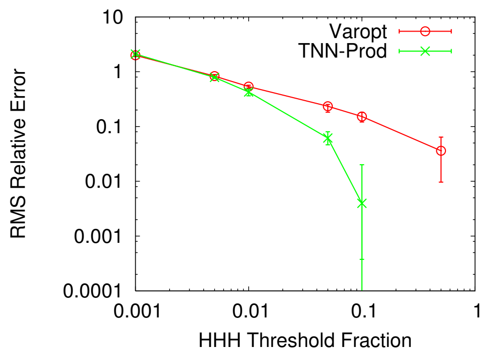


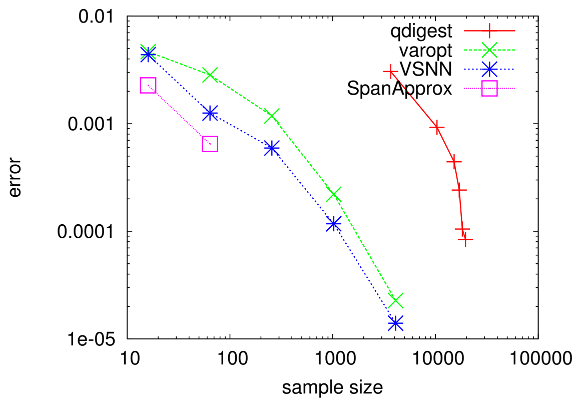
Figure 5: HHH Detection: Relative Error vs. Threshold

The MRE is broken down by prefix length in Figure 4(b) for $k = 256$ samples. For the longer prefixes, which have very low associated weight, our implementation of Q-digest essentially estimates their weight as 0. We see that even here, sampling methods can achieve better accuracy. Structure aware sampling is more accurate than structure-oblivious down to a prefix length of about 16, corresponding to ranges that are only a $1/2^{16}$ fraction of the data domain size and (roughly) the corresponding fraction of the data weight. Even the least costly computational methods (SNN) have accuracy up to an order of magnitude better than SVarOpt, with even greater gains possible for the more computationally intensive TNN. As expected SVarOpt is (barely) the most accurate for long prefixes; it has optimal variance for the case of leaf nodes. This confirms our claims, that for the common case of range queries, there are considerable benefits to making sampling structure aware.

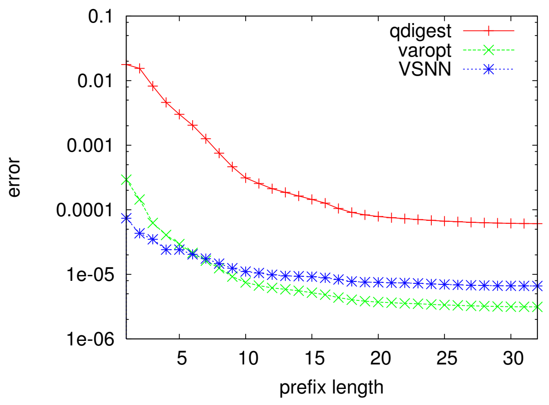
We show an example of how improved sampling can assist data mining applications, in this example, detection of Hierarchical Heavy Hitters (HHH). Figure 5 displays the Root Mean Square relative estimation error of HHH weights for SVarOpt and TNN-prod, as a function of the HHH detection threshold. The central observation here is that the structure-aware sampling can be substantially more accurate than the structure-oblivious approach.

Two-dimensional Results. Figure 6(a) shows the global MRE in the two-dimensional case. Here, even with large values of the parameter ϵ, the Q-digest approach needs to build a summary of size 10^5 or higher to achieve good accuracy. Hence the difference between Q-digest and the sampling-based methods is more striking, being over 2 orders of magnitude on comparable sample sizes. Again, there is a clear improvement of structure aware sampling over structure oblivious. The VSNN method has less than half the error of SVarOpt for moderate sample sizes (from 10s to 1000s of sampled keys). The partial results for SpanApprox indicate that there is room for a further factor of 2 improvement. However, the more extensive search for NN pairs means that the computational cost becomes prohibitive for a sample size of more than 100.

When we compare the accuracy over different prefix lengths (Figure 6(b), we see similar behavior to the 1-dimensional case, but accentuated. Recall that here we compare queries over square ranges, corresponding to prefixes of the same length on each dimension. There is a clear benefit for VSNN over SVarOpt when the prefixes are short, corresponding to queries with an appreciable fraction of the weight. The cross-over point comes when the prefix length is about 8: this corresponds to a $2^{-16} = 1.5 \times 10^{-5}$ fraction of the toal area. So with a sample of 1024 keys, the structure-aware approach still obtains good accuracy for queries down to ranges which touch a relatively small fraction of the data. We also conducted a



(a) Overall MRE as a function of sample size k.



(b) MRE as a function of prefix length for $k = 1024$ samples.

Figure 6: Accuracy: 2-dimensions.

single set of experiments on the full set of 196K pair flows, comparing SVarOpt and VSNN. The behavior was similar to that just described, with VSNN outperforming SVarOpt in accuracy on queries over square ranges down to prefix lengths of 10.

Discussion. In both the one-dimensional and two-dimensional cases we saw (i) the smaller accuracy of Q-digest, relative to other methods; (ii) the greatest accuracy of the most costly heuristics; (iii) The computationally cheapest structure aware approaches based on SNN still give an considerable increase in accuracy relative to SVarOpt, down to prefix length 16 in the 1-dimensional case. That the structure-aware SSN heuristic is most accurate on queries with high weight is a factor in its favor as compared with existing methods. As expected, SVarOpt is more accurate on very small ranges. However, the penalty paid by structure aware sampling in error relative to SVarOpt in these cases is not great, making it a good general-purpose summary.

7. CONCLUDING REMARKS

We have presented a new class of algorithms for stream sampling that is *structure aware*. Structure-aware sampling combines the principal benefits of sampling, as traditionally used in a structure-oblivious manner, and of dedicated summaries designed specifically for range-sum queries and applications such as heavy hitter detection. Our summaries retain the ability to answer arbitrary subset queries and produce unbiased estimates with tail bounds on the error. These have guaranteed performance at least as good as that of smaller VarOpt summaries. At the same time, we can optimize for the structure present in the data which makes range queries much more likely than other subset queries, achieving superior accuracy on such queries. In practice this can be achieved at a low computational cost: $O(\log k)$ per new key in a buffer of k keys.

For our analysis, we used the assumption that the streams are aggregated, i.e. each key appears at most once. Our algorithms naturally extend to handle unaggregated streams, following the structure-oblivious setting [6]: if the key of the stream element is already in S, the adjusted weight is increased by new value (no need to pivot). Our analysis, however, does not cover this case. In future work, we plan to study structure-aware summarization of unaggregated data streams, where keys may appear multiple times and the weight of a key is the sum of weights over occurrences.

8. REFERENCES

[1] C. Buragohain and S. Suri. Quantiles on streams. In *Encyclopedia of Database Systems*. Springer, 2009.
[2] M. T. Chao. A general purpose unequal probability sampling plan. *Biometrika*, 69(3):653–656, 1982.
[3] H. Chernoff. A measure of the asymptotic efficiency for test of a hypothesis based on the sum of observations. *Annals of Math. Statistics*, 23:493–509, 1952.
[4] E. Cohen, G. Cormode and N. Duffield. Structure-aware sampling: Flexible and accurate summarization. arXiv:1102.5146, 2011.
[5] E. Cohen, N. Duffield, H. Kaplan, C. Lund, and M. Thorup. Stream sampling for variance-optimal estimation of subset sums. In *ACM-SIAM SODA*, 2009.
[6] E. Cohen, N. Duffield, H. Kaplan, C. Lund, and M. Thorup. Composable, Scalable, and Accurate Weight Summarization of Unaggregated Data Sets In *VLDB*, 2009.
[7] E. Cohen, N. Duffield, C. Lund, M. Thorup, and H. Kaplan. Variance optimal sampling based estimation of subset sums. Tech. report arXiv:0803.0473v1 [cs.DS], 2008.
[8] G. Cormode and M. Hadjieleftheriou. Finding frequent items in data streams. In *VLDB*, 2008.
[9] G. Cormode, F. Korn, S. Muthukrishnan, and D. Srivastava. Space- and time-efficient deterministic algorithms for biased quantiles over data streams. In *ACM PODS*, 2006.
[10] G. Cormode, F. Korn, S. Muthukrishnan, and D. Srivastava. Finding hierarchical heavy hitters in streaming data. *ACM Trans. Knowl. Discov. Data*, 1(4):1–48, 2008.
[11] R. Gandhi, S. Khuller, S. Parthasarathy, and A. Srinivasan. Dependent rounding and its applications to approximation algorithms. *J. Assoc. Comput. Mach.*, 53(3):324–360, 2006.
[12] J. Hájek. *Sampling from a finite population*. Marcel Dekker, 1981.
[13] D. G. Horvitz and D. J. Thompson. A generalization of sampling without replacement from a finite universe. *J. Amer. Stat. Assoc.*, 47(260):663–685, 1952.
[14] A. Panconesi and A. Srinivasan. Randomized distributed edge coloring via an extension of the Chernoff-Hoeffding bounds. *SIAM J. Comput.*, 26(2):350–368, 1997.
[15] N. Shrivastava, C. Buragohain, D. Agrawal, and S. Suri. Medians and beyond: new aggregation techniques for sensor networks. In *ACM SenSys*, 2004.
[16] A. Srinivasan. Distributions on level-sets with applications to approximation algorithms. In *IEEE FOCS*. 2001.
[17] M. Szegedy and M. Thorup. On the variance of subset sum estimation. In *Proc. ESA*, 2007.
[18] Y. Tillé. *Sampling Algorithms*, Springer, 2006.
[19] J. Vitter. Random sampling with a reservoir. *ACM Trans. Math. Softw.*, 11(1):37–57, 1985.
[20] Y. Zhang, S. Singh, S. Sen, N. Duffield, and C. Lund. Online identification of hierarchical heavy hitters. In *SIGMETRICS*, 2004.

APPENDIX

A. OMITTED PROOFS

Proof of Theorem 3

PROOF OF THEOREM 3. We use the fact that over the set of all random variables $X \in [0,1]$ with expectation p, $\mathsf{Var}[X]$ is maximized for $X \in \{0,1\}$, that is, the Bernoulli random variable which is $X = 1$ with probability p and $X = 0$ otherwise.

Under VAROPT with $\tau_{k^*(M)}$, the HT adjusted weight a_i is equal to $\tau_{k^*(M)}$ with probability $w_i/\tau_{k^*(M)}$ and 0 otherwise. For an M-bounded summary, the adjusted weight a'_i satisfies $a'_i \leq M \leq \tau_{k^*(M)}$.

From unbiasedness, $\mathsf{E}[a_i] = \mathsf{E}[a'_i] = w_i$. The random variables $a_i/\tau_{k^*(M)}$ and $a'_i/\tau_{k^*(M)}$ are both in $[0,1]$ and both have the same expectation. The random variable $a_i/\tau_{k^*(M)}$ is in $\{0,1\}$ and hence $\mathsf{Var}[a_i/\tau_{k^*(M)}] \geq \mathsf{Var}[a'_i/\tau_{k^*(M)}]$, implying $\mathsf{Var}[a_i] \geq \mathsf{Var}[a'_i]$. □

Proof of Theorem 4

PROOF OF THEOREM 4. We establish that the tail bounds on a_J obtained, as explained in Section 2, by first excluding keys with $a_i \geq M$ from the estimate and then applying Chernoff bounds hold for M-bounded summaries.

All keys with $w_i \geq M$ appear with their actual weight in the M-bounded summary, and hence, can also be excluded. We thus assume all keys in J have $w_i < M$.

Consider the random variables $X_i = a_i/M$ which are in $[0,1]$. We have $p_i = \mathsf{E}[X_i] = w_i/M$. We have that for any $J \subseteq [n]$,

$$\text{(I):} \quad \mathsf{E}[\textstyle\prod_{i\in J} X_i] \leq \prod_{i\in J} p_i$$
$$\text{(E):} \quad \mathsf{E}[\textstyle\prod_{i\in J}(1-X_i)] \leq \prod_{i\in J}(1-p_i)$$

It is established in [14] (see also [11, 7]) that for $X_i \in \{0,1\}$, these properties imply Chernoff tail bounds on $\sum_{i\in J} p_i$.

When used with VAROPT [7], we apply the inequalities with respect to the random variables $a_i/\tau \in \{0,1\}$. The bounds are then multiplied by τ to obtain bounds on the weights.

It is not hard to verify that the proofs that (I) and (E) imply these bounds carry over when we allow X_i to assume fractional values $X_i \in [0,1]$. By multiplying the bounds we obtain on $\sum_{i\in J|w_i<M} p_i$ by M, we obtain corresponding bounds on $\sum_{i\in J|w_i<M} w_i$.

Since $M \leq \tau_{k^*(M)}$, the bounds are at least as tight as the tail bounds we obtain for VAROPT$_{k^*(M)}$. □

Proof of Lemma 5

PROOF OF LEMMA 5. Let a' be thr output of PIVOT(a, X). As the other properties are straightforward, it remains to establish (3) and (4). That is, for any subset J, and $\overline{M} \geq M(a,X)$,

$$\text{(I):} \quad \mathsf{E}[\textstyle\prod_{i\in J} a'_i] \leq \prod_{i\in J} a_i$$
$$\text{(E):} \quad \mathsf{E}[\textstyle\prod_{i\in J}(\overline{M}-a'_i)] \leq \prod_{i\in J}(\overline{M}-a_i)$$

It suffices to show this for $J \subset \text{CAND}(X)$, since for $i \notin \text{CAND}(X)$, $a'_i \equiv a_i$, and the contributions of these keys to the products in both sides of the inequalities are the same. To simplify notation, we assume $X = \text{CAND}(X)$. This is without loss of generality since the pivot step is identical when applied to X or to CAND(X).

Inclusion Bound (I). The probability that item $i \in J$ has $a'_i = 0$ is $1 - \frac{a_i(|X|-1)}{a_X}$. Since these are disjoint events for different keys, the probability that one key from J has $a'_i = 0$ is $|J| - \frac{a_J(|X|-1)}{a_X}$.

Hence, the probability that all keys in J have $a'_i = M(a,X)$ is $\frac{a_J(|X|-1)}{a_X} - |J| + 1$. The product $\prod_{i\in J} a'_i = 0$ if a key from J has $a'_i = 0$ and is $M(a,X)^{|J|}$ otherwise. Thus,

$$\mathsf{E}[\prod_{i\in J} a'_i] = \left(\frac{a_J(|X|-1)}{a_X} - |J| + 1\right)(M(a,X))^{|J|}. \quad (10)$$

Because we have $X = \text{CAND}(X)$, it must satisfy the condition $\forall i \in X, a_i \leq M(a,X) = a_X/(|X|-1)$. Therefore,

$$\begin{aligned} a_J &= a_X - a_{X\setminus J} = a_X - \textstyle\sum_{i\in X\setminus J} a_i \\ &\geq a_X - (|X|-|J|)\frac{a_X}{|X|-1} = a_X\frac{|J|-1}{|X|-1}. \\ a_J &\leq a_X\frac{|J|}{|X|-1} \end{aligned}$$

For a given J and total adjusted weight a_J, $\prod_{i\in J} a_i$ is minimized when one key in J has $a_i = a_J - a_X\frac{|J|-1}{|X|-1}$ and the $|J|-1$ other keys in J having adjusted weights $a_i = \frac{a_X}{|X|-1}$. Hence,

$$\begin{aligned} \prod_{i\in J} a_i &\geq \left(\frac{a_X}{(|X|-1)}\right)^{|J|-1}\left(a_J - a_X\frac{|J|-1}{|X|-1}\right) \\ &= \left(\frac{a_X}{(|X|-1)}\right)^{|J|}\left(\frac{a_J(|X|-1)}{a_X} - |J| + 1\right) \quad (11) \end{aligned}$$

The bound (I) follows by combining (10) and (11).

Exclusion bound (E). By definition, $\overline{M} \geq M(a,X)$. Because $J \subset \text{CAND}(X)$, for all $i \in J$, $a'_i \leq M(a,X) \leq \overline{M}$ and $a_i \leq M(a,X) \leq \overline{M}$. For a given J and a_J, $\prod_{i\in J}(\overline{M}-a_i)$ is minimized when one key in J has $a_i = a_J - a_X\frac{|J|-1}{|X|-1}$ and other keys in J have $a_i = \frac{a_X}{|X|-1}$. Therefore,

$$\prod_{i\in J}(\overline{M}-a_i) \geq \left(\overline{M} - a_J + a_X\tfrac{|J|-1}{|X|-1}\right)\left(\overline{M} - \tfrac{a_X}{|X|-1}\right)^{|J|-1} \quad (12)$$

The value of $\prod_{i\in J}(\overline{M}-a'_i)$ is equal to $(\overline{M} - \frac{a_X}{|X|-1})^{|J|}$ if all keys in $i \in J$ have $a'_i = M(a,X)$ and is equal to $\overline{M}(\overline{M} - \frac{a_X}{|X|-1})^{|J|-1}$ otherwise (one of the keys has $a'_i = 0$). Substituting the probabilities of these two outcomes we obtain

$$\begin{aligned} &\mathsf{E}[\prod_{i\in J}(\overline{M}-a'_i)] \\ =& \left(\overline{M} - \frac{a_X}{|X|-1}\right)^{|J|-1}\left(\overline{M}\left(|J| - \frac{a_J(|X|-1)}{a_X}\right)\right) + \\ & \left(\overline{M} - \frac{a_X}{|X|-1}\right)^{|J|}\left(\frac{a_J(|X|-1)}{a_X} - (|J|-1)\right) \\ =& \left(\overline{M} - \tfrac{a_X}{|X|-1}\right)^{|J|-1}\left(\overline{M} - a_J + a_X\tfrac{|J|-1}{|X|-1}\right) \end{aligned} \quad (13)$$

(E) follows by combining (12) and (13). □

Proof of Lemma 7

PROOF OF LEMMA 7. This clearly holds for the $h \leq k$ prefix of the stream, since $a \equiv w$. For $h > k$, the proof proceeds by induction. Assume that at the beginning of the iteration $\ell = h - k$ the sample S is a $\tau_{k/c}(w_1,\ldots,w_{h-1})$-bounded summary of the input $(w_1,\ldots,w_{h-1})$. From the algorithm, all keys for which the adjusted weight had increased during iteration ℓ have final adjusted weight value $\leq \tau_{k/c}(a_{S\cup\{h\}}) = \tau_{k/c}(w_1,\ldots,w_h)$. Thus it is a $\tau_{k/c}(w_1,\ldots,w_h)$ bounded summary. □