

Structure-Aware Sampling: Flexible and Accurate Summarization

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

In processing large quantities of data, a fundamental problem is to obtain a summary which supports approximate query answering. Random sampling yields flexible summaries which naturally support subset-sum queries with unbiased estimators and well-understood confidence bounds.

Classic sample-based summaries, however, are designed for arbitrary subset queries and are oblivious to the structure in the set of keys. The particular structure, such as hierarchy, order, or product space (multi-dimensional), makes *range queries* much more relevant for most analysis of the data.

Dedicated summarization algorithms for range-sum queries have been extensively studied. They can outperform existing sampling schemes in terms of accuracy on range queries per summary size. Their accuracy, however, rapidly degrades when, as is often the case, the query spans multiple ranges. They are also less flexible—being targeted for range sum queries alone—and are often quite costly to build and use.

In this paper we propose and evaluate variance optimal sampling schemes that are *structure-aware*. These summaries considerably improve over the accuracy of existing *structure-oblivious* sampling schemes on range queries while retaining the benefits of sample-based summaries: they provide a flexible summary, with high accuracy on both range queries and arbitrary subset queries.

1. INTRODUCTION

Consider a scenario where a large volume of data is collected on a daily basis: for example, sales records in a retailer, or network activity in a telecoms company. This activity will be archived in a warehouse or other storage mechanism, but the size of the data is too large for data analysts to keep in memory. Rather than go out to the full archive for every query, it is natural to retain accurate summaries of each data table, and use these queries for data exploration and analysis, reducing the need to read through the full history for each query. Since there can be many tables (say, one for every day at each store, in the retailer case, or one for every hour and every router in the network case), we want to keep a very compact summary of each table, but still guarantee accurate answers to any

query. The summary allows approximate processing of queries, in place of the original data (which may be slow to access or even no longer available); it also allows fast ‘previews’ of computations which are slow or resource hungry to perform exactly.

This scenario has motivated a wealth of work on data summarization and approximation, which we survey in detail later on. There are two main themes: methods based on random sampling, and algorithms that build more complex data structures (often deterministic, but also randomized) of the data. Both have their pros and cons. Sampling is fast and efficient, and has useful guaranteed properties. Dedicated summaries can offer greater accuracy for the kind of range queries which are most common over large data, albeit at a greater cost to compute, and providing less flexibility for other query types. Our goal in this work is to provide summaries which combine the best of both worlds: fast, flexible summaries which are very accurate for the all-important range queries. To attain this goal, we must understand existing methods in detail to see how to improve on their properties.

Summaries which are based on random sampling allow us to build (unbiased) estimates of properties of the data set, such as counts of individual identifiers (keys), sums of weights for particular subsets of keys, and so on, all specified after the data has been seen. Having high-quality estimates of these primitives allows us to implement higher-level applications over samples, such as computing order statistics over subsets of the data, heavy hitters detection, longitudinal studies of trends and correlations, and so on.

Summarization of items with weights traditionally uses Poisson sampling, where each item is sampled independently. The approach which sets the probability of including an item in the sample to be proportional to its weight (IPPS) [11] enables us to use the Horvitz-Thompson estimator [14], which minimizes the sum of per-item variances. “VAROPT” samples [3, 25, 6] improve on Poisson samples in that the sample size is fixed and they are more accurate on subset-sum queries. In particular VAROPT samples have *variance optimality*: they achieve variance over the queries that is provably the smallest possible for any sample of that size.

Since sampling is simple to implement and flexible to use, it is the default summarization method for large data sets. Samples support a rich class of possible queries directly: evaluating the query over the sampled data (with appropriately scaled weights) usually provides an unbiased, low variance estimate of the true answer, while not requiring any new code to be written. These summaries provide not only estimates of aggregate values but also a representative sample of keys that satisfy a selection criteria. The fact that estimates are unbiased also means that relative error decreases for queries that span multiple samples or larger subsets and the estimation error is governed by exponential tail bounds: the estimation error, in terms of the number of samples from any particular subset,

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.

Submitted to SIGMOD 2011

Copyright 200X ACM X-XXXXX-XX-X/XX/XX ...\$5.00.

is highly concentrated around the square root of the expectation.

We observe, however, that traditionally sampling has neglected the inherent structure that is present, and which is known before the data is observed. That is, data typically exists within a well-understood schema that exhibits considerable structure. Common structures include *order* where there is a natural ordering over keys; *hierarchy* where keys are leaves within a hierarchy (e.g. geographic hierarchy, network hierarchy); and combinations of these where keys are multi-dimensional points in a *product structure*. Over such data, queries are often *structure-respecting*. For example, on ordered data with n possible key-values, although there are $O(2^n)$ possible subset-sum queries, the most relevant queries may be the $O(n^2)$ possible range queries. In a hierarchy, relevant queries may correspond to particular nodes in the hierarchy (geographic areas, IP address prefixes), which represent $O(n \log n)$ possible ranges. In a product structure, likely queries are boxes—intersections of ranges of each dimension.

As a motivating example, consider network data in the form of IP flow records. Each record has a source and destination IP address, a port number, and size (number of bytes). IP addresses form a natural hierarchy, where prefixes or sets of prefixes define the ranges of interest. Port numbers indicate the generating application, and related applications use ranges of port numbers. Flow summaries are used for many network managements tasks, including planning routing strategies, and traffic anomaly detection. Resources for collection, transport, storage and analysis of network measurements are expensive; therefore, structure-aware summaries are needed by network operators to understand the behavior of their network.

While samples have been shown to work very well for queries which resemble the sums of *arbitrary* subsets of keys, they tend to be less satisfactory when restricted to range queries. Given the same summary size, samples can be out-performed in accuracy by dedicated methods such as (multi-dimensional) histograms [10, 19, 15], wavelet transforms [16, 27], and geometric summaries [23, 13, 1, 8, 28] including the popular Q-digest [21].

These dedicated summaries, however, have inherent drawbacks: they primarily support queries that are sum aggregates over the original weights, and so other queries must be expressed in terms of this primitive. Their accuracy rapidly degrades when the query spans multiple ranges—a limitation since natural queries may span several (time, geographic) ranges within the same summary and across multiple summaries. Dedicated summaries do not provide “representative” keys of selected subsets, and require changes to existing code to utilize. Of most concern is that they can be very slow to compute, requiring a lot of I/O (especially as the dimensionality of the data grows): a method which gives a highly accurate summary of each hour’s data is of little use if it takes a day to build! Lastly, the quality of the summary may rely on certain structure being present in the data, which may not always hold. While these summaries have shown their value in efficiently summarizing one-dimensional data (essentially, arrays of counts), their behavior on even two-dimensional data is less satisfying: troubling since this is where accurate summaries are most needed. For example, in the network data example, we are often interested in the traffic volume between (collections of) various source and destination ranges.

Motivated by the limitations of dedicated summaries, and the potential for improvement over existing (structure-oblivious) sampling schemes, we aim to design sampling schemes that are both VAROPT and *structure-aware*. At the same time, we aim to match the accuracy of deterministic summaries on range sum queries and retain the desirable properties of existing sample-based summaries: unbiasedness, tail bounds on arbitrary subset-sums, flexibility and support for representative samples, and good I/O performance.

1.1 Our Contributions

We introduce a novel algorithmic sampling framework, which we refer to as *probabilistic aggregation*, for deriving VAROPT samples. This framework makes explicit the freedom of choice in building a VAROPT summary which has previously been overlooked. Working within this framework, we design *structure-aware* VAROPT sampling schemes which exploit this freedom to be much more accurate on ranges than their structure-oblivious counterparts.

- For hierarchies, we design an efficient algorithm that constructs VAROPT summaries with bounded “range discrepancy”. That is, for any range, the number of samples deviates from the expectation by less than 1. This scheme has the minimum possible variance on ranges of any unbiased sample-based summary.
- For ordered sets, where the ranges consist of all intervals, we provide a sampling algorithm which builds a VAROPT summary with range discrepancy less than 2. We prove that this is the best possible for any VAROPT sample.
- For multidimensional datasets, we propose sampling algorithms where the discrepancy between $p(R)$, the expected number of sample points in the range R , and the actual number is $O(\min\{ds^{\frac{d-1}{2d}}, \sqrt{p(R)}\})$, where s is the sample size. This improves over structure-oblivious random sampling, where the corresponding discrepancy is $O(\sqrt{p(R)})$.

Construction Cost. For a summary structure to be effective, it must be possible to construct quickly, and with small space requirements. Our main-memory sampling algorithms perform tasks such as sorting keys or (for multidimensional data) building a kd-tree. We propose even cheaper alternatives which perform two read-only passes over the dataset using memory that depends on the desired summary size s (and independent of the size of the data set). When the available memory is $O(s \log s)$, we obtain a VAROPT sample that with high probability $1 - O(1/\text{poly } s)$ is close in quality to the algorithms which store and manipulate the full data set.

Empirical study. To demonstrate the value of our new structure-aware sampling algorithms, we perform experiments comparing to popular summaries, in particular the wavelet transform [27], ϵ -approximations [13], randomized sketches [4] and to structure-oblivious random sampling. These experiments show that it is possible to have the best of both worlds: summaries with equal or better accuracy than the best-in-class, which are dramatically more efficient to construct and work with.

2. BACKGROUND ON SAMPLING

We begin by outlining some key concepts in sampling. The data is modeled as a *weighted set* where each key $i \in I$ has weight $w_i \geq 0$ and a sample is a random subset $S \subset I$. For concreteness, it may be helpful to consider the uniform weighted case, where each key has weight 1—our results are novel and interesting even in this simpler case.

Inclusion Probability Proportional to Size (IPPS) [11]: Many sampling schemes set the inclusion probability of each key in S proportional to its weight, but truncated so as not to exceed 1. When defined with respect to a *threshold* parameter $\tau > 0$, the inclusion probability of i is $p_i = \min\{1, w_i/\tau\}$.

The expected size of the sample is just the sum of the p_i ’s, so we can achieve a sample of expected size k by choosing an appropriate threshold τ_k . This τ_k can be found by solving the equation:

$$\sum_i \min\{1, w_i/\tau_k\} = k. \quad (1)$$

Algorithm 1 STREAM- $\tau(i)$: processing item i

```

1: if  $w_i < \tau$  then  $L \leftarrow L + w_i$ 
2: else  $\text{Insert}((i, w_i), H)$ ;

3: while  $(|H| = k)$  or  $(\min(H) < \frac{L}{k-|H|})$  do
4:    $a \leftarrow \text{delete\_min}(H)$ .
5:    $L \leftarrow L + w_a$ 
6:  $\tau \leftarrow \frac{L}{k-|H|}$ 

```

τ_k can be computed via a linear pass on the data, using a heap H of size at most k . Let L denote the total weight of keys processed that are not present in the heap H . Algorithm 1 presents the algorithm to update the value of τ for each new key. If the weight of the new key is below the current value of τ , then we add it to L , the sum of weights; else, we include the weight in the heap H . Then adjust the heap: if the heap has k items, or the smallest weight in the heap falls below the new value of τ (Line 3), we move the smallest weight from H to L . Finally, we compute the new τ value in Line 6. After processing all keys, we have found τ_k for this data. Moreover, we can use the weights in H and the value of L at the final step to compute $\tau_{k'}$ for any $k' < k$.

The Horvitz Thompson (HT) estimator [14] allows us to accurately answer queries from a sample, by providing *adjusted weights* to use for each key in the sample. For a key i of weight w_i and inclusion probability p_i the adjusted weight is $a(i) = w_i/p_i$ if i is included in the sample and $a(i) \equiv 0$ otherwise. For all i , $a(i)$ is an optimal estimator of w_i in that it minimizes the variance $\text{Var}[a(i)]$.

A summary that includes the keys in S and their HT adjusted weights can be used to estimate the weight $w(J)$ of any subset of keys $J \subset I$: $a(J) = \sum_{i \in J} a(i) = \sum_{i \in S \cap J} a(i)$. The HT estimates are clearly unbiased: for all i , $\mathbb{E}[a(i)] = w_i$ and from linearity of expectation, for all subsets J , $\mathbb{E}[a(J)] = w(J)$.

With IPPS, the HT adjusted weight of an included key is τ if $w_i \leq \tau$ and w_i otherwise. Hence, for any subset J we have

$$a(J) = \sum_{i \in J | w_i \geq \tau} w_i + |\{i \in J \cap S : w_i < \tau\}| \cdot \tau. \quad (2)$$

Within the summary, we can store either the adjusted weights for each key, or the original weights (and compute the adjusted weights via τ on demand). The variance of any adjusted weight is $\text{Var}[a_i] = w_i^2(1/p_i - 1) \equiv w_i(\tau - w_i)$ if $w_i \leq \tau$ and 0 otherwise. Using IPPS inclusion probabilities with HT estimates minimizes the sum $\sum V[a] = \sum_i \text{Var}[a(i)]$ of per-key variances over all inclusion probabilities and estimators with the same (expected) sample size $k = \sum_i p_i$.

Poisson sampling is where the choice of whether to sample any key is made independently of all other choices. A Poisson IPPS sample of a specified expected size k can be computed efficiently with a single pass over the data.

VAROPT sampling [3, 25, 6] improves over Poisson sampling by guaranteeing a fixed sample size (exactly k) and giving tighter estimates [6, 7, 3, 25, 9]: the variance of any subset-sum estimate is at most that obtained by a Poisson IPPS sample. VAROPT samples are optimal in that for *any* subset size, they minimize the average variance of the estimates [24, 6]. A VAROPT sample of size k , denoted VAROPT $_k$, can be computed with a single pass over the data [7], generalizing the notion of reservoir sampling.

A sample distribution is VAROPT for a parameter τ if:

- (i) Inclusion probabilities are IPPS, i.e. for a key i , $p_i = \min\{1, w_i/\tau\}$.

- (ii) The sample size is exactly $k = \sum_{i \in I} p_i$.

- (iii) *High-order inclusion and exclusion probabilities are bounded by the respective product of first-order probabilities.* More precisely for any $J \subseteq [n]$,

$$(I): \quad \mathbb{E}\left[\prod_{i \in J} X_i\right] \leq \prod_{i \in J} p_i$$

$$(E): \quad \mathbb{E}\left[\prod_{i \in J} (1 - X_i)\right] \leq \prod_{i \in J} (1 - p_i)$$

where p_i is the probability that key i is included in the sample S and $\mathbb{E}[\prod_{i \in J} X_i]$ is the probability that all $i \in J$ are included in the sample. Symmetrically $\mathbb{E}[\prod_{i \in J} (1 - X_i)]$ is the probability that all $i \in J$ are excluded from the sample.

Tail bounds. For both Poisson [5] and VAROPT [17, 22, 9, 7] samples we have strong tail bounds on the number of samples $J \cap S$ from a subset J . We state the basic Chernoff bounds on this quantity (other more familiar bounds are derivable from them). Let X_i be an indicator variable for i being in the sample, so $X_i = 1$ if $i \in S$ and 0 otherwise. Then X_J , the number of samples intersecting J is $X_J = \sum_{i \in J} X_i$, with mean $\mu = \mathbb{E}[X_J] = \sum_{i \in J} p_i$.

If $\mu \leq a \leq k$, the probability of getting more than a samples out of k in the subset J is

$$\Pr[X_J \geq a] \leq \left(\frac{k - \mu}{k - a}\right)^{m-a} \left(\frac{\mu}{a}\right)^a \left[\leq e^{a-\mu} \left(\frac{\mu}{a}\right)^a\right]. \quad (3)$$

While for $0 \leq a \leq \mu$, the probability of having fewer than a samples in J is

$$\Pr[X_J \leq a] \leq \left(\frac{k - \mu}{k - a}\right)^{m-a} \left(\frac{\mu}{a}\right)^a \left[\leq e^{a-\mu} \left(\frac{\mu}{a}\right)^a\right]. \quad (4)$$

When sampling with IPPS, these tight bounds on the number of samples in J also imply tight bounds on the estimated weight $a(J)$: It suffices to consider J such that $\forall i \in J, p_i < 1$ (since we have the exact weight of keys with $p_i = 1$). For such a J , the HT estimate is $a(J) = \tau X_J = \tau |J \cap S|$ and thus the estimate is guaranteed to be close to the true value:

$$\Pr[a(J) \leq h], \Pr[a(J) \geq h] \leq e^{(h-w(J))/\tau} (w(J)/h)^{h/\tau}. \quad (5)$$

Range discrepancy. The *discrepancy* of a sample measures the difference between the number of keys sampled in a range to the number expected to be there. Formally, consider keys with attached IPPS inclusion probabilities p_i over a structure with a set of ranges \mathcal{R} . The discrepancy Δ of a set of keys S with respect to range R is

$$\Delta(S, R) = \left| |S \cap R| - \sum_{i \in R} p_i \right|.$$

The *maximum range discrepancy* of S is accordingly the maximum discrepancy over $R \in \mathcal{R}$. For a sample distribution Ω , we define the maximum range discrepancy as

$$\Delta = \max_{S \in \Omega} \max_{R \in \mathcal{R}} \left| |S \cap R| - \sum_{i \in R} p_i \right| \quad (6)$$

The value of the discrepancy has implications for the accuracy of query answering. The absolute error of the HT estimator (2) on R is the product of τ and the discrepancy: $\tau \cdot \Delta(S, R)$. We therefore seek sampling schemes which have a small discrepancy.

For a Poisson IPPS or VAROPT sample, the discrepancy on a range R is subject to tail bounds (Eqns. (3) and (4)) with $\mu = p(R)$ and has expectation $O(\sqrt{p(R)})$.

ϵ -approximations. Maximum range discrepancy is closely related to the concept of an ϵ -approximation [26]. A set of points A is an ϵ -approximation of the range space (X, \mathcal{R}) , if for every range R ,

$$\left| \frac{|A \cap R|}{|A|} - \frac{|X \cap R|}{|X|} \right| < \epsilon.$$

(This is stated for uniform weights but easily generalizes). An ϵ -approximation of size s has maximum range discrepancy $\Delta = \epsilon s$. A seminal result by Vapnik and Chervonenkis bounds the maximum estimation error of a random sample over ranges when the VC dimension is small:

THEOREM 1 (FROM [26]). *For any range space with VC dimension d , there is a constant c , so that a random sample of size*

$$s = c\epsilon^{-2}(d \log(d/\epsilon) + \log(1/\delta))$$

is an ϵ -approximation with probability $1 - \delta$.

The structures we consider have constant VC dimension, and the theorem can be proved from a direct application of Chernoff bounds. Because VAROPT samples satisfy Chernoff bounds, they also satisfy the theorem statement. By rearranging and substituting the bound on ϵ , we conclude that a Poisson IPPS or a VAROPT sample of size s has maximum range discrepancy $\Delta = O(\sqrt{s} \log s)$, with probability $(1 - \text{poly}(1/s))$

3. RELATED WORK

In addition to sampling methods such as Poisson IPPS and VAROPT there have been many other approaches to summarizing data in range spaces. We provide a quick review of the rich literature on summarization methods specifically designed for range-sum queries. Some methods use random sampling in their construction, although the resulting summary is not itself a VAROPT sample.

ϵ -approximations. As noted above, ϵ -approximations accurately estimate the number of points falling in a range. For axis-parallel hyper-rectangles, Suri, Toth, and Zhou presented randomized constructions that with constant probability generates an ϵ -approximation of size $O(\epsilon^{-\frac{2d}{d+1}} \log^d(\epsilon^{-\frac{2}{d+1}} n))$ and an alternative but computationally intensive construction with much better asymptotic dependence on ϵ : $O(\frac{1}{\epsilon} \log^{2d+1} \frac{1}{\epsilon})$ [23]. The best space upper bound we are aware of is $O(\frac{1}{\epsilon} \log^{2d}(\frac{1}{\epsilon}))$ [18].

A proposal in [2] is to construct an ϵ -approximation of a random sample of the dataset instead of the full dataset. This is somewhat related to our I/O efficient constructions that utilize an initial larger random sample. The differences are that we use the sample only as a guide—the final summary is not necessarily a subset of the initial sample—and that the result of our construction is a structure-aware VAROPT sample of the full data set.

Another construction [13] trades better dependence on ϵ with logarithmic dependence on domain size. The data structure is built deterministically by dividing on each dimension in turn, and retaining “heavy” ranges. This can be seen as a multi-dimensional variant of the related q-digest data structure [21]. We compare to this structure in our experimental study.

Sketches and Projections. Random projections are a key tool in dimensionality reduction, which allows large data sets to be compactly summarized and queried. Sketches are particular kinds of random projections, which can be computed in small space [4]. By keeping multiple sketches of the data at multiple levels of granularity, we can provide ϵ -approximation-like bounds in space that depends linearly on ϵ^{-1} and logarithmically on the domain size.

Wavelet transforms and deterministic histograms A natural approach to summarizing large data for range queries is to decompose the range space into “heavy” rectangles. The answer to any

range query is the sum of weights of all rectangles fully contained in by the query, plus partial contributions from those partly overlapped by the query. The accuracy then depends on the number (and weight) of rectangles overlapped by the query. This intuition underlies various attempts based on building (multi-dimensional) histograms [10, 19, 15]. These tend to be somewhat heuristic in nature, and offer little by way of guaranteed accuracy.

More generally, we can represent the data in terms of objects other than rectangles: this yields transformations such as DCT, Fourier transforms and wavelet representations. Of these, wavelet representations are most popular for representing range data [16, 27]. Given data of domain size u , the transform generates u coefficients, which are then *thresholded*: we pick the k largest coefficients (after normalization) to represent the data. When the data is dense, we can compute the transform in time $O(u)$, but when the domain is large and the data sparse, it is more efficient to generate the transform of each key, in time proportional to the product of the logarithms of the size of each dimension per key.

4. PROBABILISTIC AGGREGATION

The first step in our approach to making sampling structure aware is to define a relaxation of VAROPT sampling which allows for *partial* inclusions of keys. These “partial” VAROPT samples, which we refer to as *probabilistic aggregates*, can always be completed to generate a VAROPT sample of the original data. The input is thought of as a vector of (sampling) probabilities, and each step of probabilistic aggregation sets some of these probabilities to 1 or 0. When the process concludes, the keys associated with probability 1 constitute the sample.

A random vector $p^{(1)} \in [0, 1]^n$ is a *probabilistic aggregate* of a vector $p^{(0)} \in [0, 1]^n$ if the following conditions are satisfied:

- (i) (*Agreement in Expectation*) $\forall i, \mathbb{E}[p_i^{(1)}] = p_i^{(0)}$,
- (ii) (*Agreement in Sum*) $\sum_i p_i^{(1)} = \sum_i p_i^{(0)}$, and
- (iii) (*Inclusion-Exclusion Bounds*)

$$(I): \quad \mathbb{E}\left[\prod_{i \in J} p_i^{(1)}\right] \leq \prod_{i \in J} p_i^{(0)}$$

$$(E): \quad \mathbb{E}\left[\prod_{i \in J} (1 - p_i^{(1)})\right] \leq \prod_{i \in J} (1 - p_i^{(0)}).$$

When all entries in $p^{(1)}$ are *set* (are in $\{0, 1\}$), $p^{(1)}$ is a VAROPT summary of $p^{(0)}$.

LEMMA 2. *Consider a sequence $p^{(0)}, p^{(1)}, p^{(2)}, p^{(3)}, \dots$ where $p^{(h)}$ is a probabilistic aggregate of $p^{(h-1)}$.*

- $p_i^{(h)} \in \{0, 1\}$ implies $p_i^{(h+1)} \equiv p_i^{(h)}$. That is, in a sequence of aggregation steps, once an entry is set it remains set. In particular, the number of positive entries in the output is at most the number in the input.
- Probabilistic aggregation is transitive, that is, if $h_1 < h_2$ then $p^{(h_2)}$ is a probabilistic aggregate of $p^{(h_1)}$.

PROOF. We show that (I) holds under transitivity. The proof for (E) is similar, and the other properties are immediate. We show that if $p^{(h+2)}$ is an aggregate of $p^{(h+1)}$, and $p^{(h+1)}$ is an aggregate of $p^{(h)}$, then $p^{(h+2)}$ is an aggregate of $p^{(h)}$.

$$\begin{aligned} \mathbb{E}_{p^{(2)}|p^{(0)}}\left[\prod_{i \in J} p_i^{(2)}\right] &= \mathbb{E}_{p^{(1)}|p^{(0)}}\left[\mathbb{E}_{p^{(2)}|p^{(1)}}\left[\prod_{i \in J} p_i^{(2)}\right]\right] \\ &\leq \mathbb{E}_{p^{(1)}|p^{(0)}}\left[\prod_{i \in J} p_i^{(1)}\right] \leq \prod_{i \in J} p_i^{(0)} \quad \square \end{aligned}$$

Algorithm 2 PAIR-AGGREGATE(p, i, j)

Require: $0 < p_i, p_j < 1$

- 1: **if** $p_i + p_j < 1$ **then**
- 2: **if** $\text{rand}() < \frac{p_i}{p_i + p_j}$ **then**
- 3: $p_i \leftarrow p_i + p_j; p_j \leftarrow 0$
- 4: **else**
- 5: $p_j \leftarrow p_i + p_j; p_i \leftarrow 0$
- 6: **else** $\triangleright p_i + p_j \geq 1$
- 7: **if** $\text{rand}() < \frac{1-p_j}{2-p_i-p_j}$ **then**
- 8: $p_i \leftarrow 1; p_j \leftarrow p_i + p_j - 1$
- 9: **else** \triangleright with probability $\frac{1-p_i}{2-p_i-p_j}$
- 10: $p_i \leftarrow p_i + p_j - 1; p_j \leftarrow 1$
- 11: **return** p

Pair Aggregation. Our summarization algorithms perform a sequence of simple aggregation steps which we refer to as *pair aggregations* (Algorithm 2). Each pair aggregation step modifies only two entries and sets at least one of them to $\{0, 1\}$. The input to *pair aggregation* is a vector p and a pair i, j with each $p_i, p_j \in (0, 1)$. The output vector agrees with p on all entries except i, j and one of the entries i, j is set to 0 or 1. Case analysis verifies that PAIR-AGGREGATE(p, i, j) correctly computes a probabilistic aggregate of its input. The algorithm terminates when all entries are set, and the output is a VAROPT sample.

Pair aggregation is a powerful primitive. It produces a sample of size exactly $k = \sum_i p_i^{(0)}$. Observe that the choice of which pair i, j to aggregate at any point can be arbitrary—and the result is still a VAROPT sample. This observation is what enables our approach. We harness this freedom in pair selection to obtain VAROPT samples that are structure aware: Intuitively, by choosing to aggregate pairs that are “close” to each other with respect to the structure, we control the range impact of the “movement” of probability mass.

5. ONE DIMENSIONAL STRUCTURES

We now use pair aggregation to make sampling structure-aware by describing ways to pick which pair of items to aggregate at each step. For now, we assume the data fits in main-memory, and our input is the list of keys and their associated IPPS probabilities p_i . We later discuss the case when the data exceeds the available memory.

For hierarchy structures (keys are associated with leaves of a tree and \mathcal{R} contains all sets of keys under some internal node) we show how to obtain VAROPT samples with (optimal) maximum range discrepancy $\Delta < 1$. There are two special cases of hierarchies: (i) *disjoint ranges* (where \mathcal{R} is a partition of I)—captured by a flat 2-level hierarchy with mid level nodes corresponding to ranges and (ii) *order* where there is a linear order on keys and \mathcal{R} is the set of all prefixes—the corresponding hierarchy is a path with single leaf below each internal node. For order structures where \mathcal{R} is the set of “intervals” (all consecutive sets of keys) we show that there is always a VAROPT sample with maximum range discrepancy $\Delta < 2$ and prove that this is the best possible.

5.1 Disjoint ranges and hierarchy

We specify pair selection criteria which result in a range-optimal summary:

- **Disjoint ranges:** Pair selection picks pairs where both keys belong to the same range R . When there are multiple choices, we

¹Assuming that $\sum_i p_i^{(0)}$ is integral. This can be ensured (deterministically) by choosing τ as described in Algorithm 1.

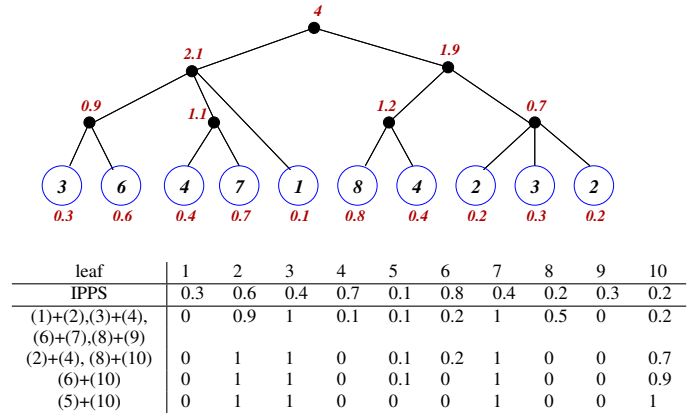


Figure 1: Sampling over a hierarchy structure with weighted leaves

may choose one arbitrarily. Only when for all $R \in \mathcal{R}$ there is at most one key that is not set, we may select a pair that spans two different ranges (arbitrarily if there are multiple choices).

- **Hierarchy:** Pair selection picks pairs with lowest LCA (lowest common ancestor). That is, we pair aggregate (i, j) if there are no other pairs with an LCA that is a descendant of $\text{LCA}(i, j)$.

Following these rules guarantees low range discrepancy: they guarantee that for all ranges $R \in \mathcal{R}$ and for all iterations h where R has at least one entry which is not set, we have $\sum_{i \in R} p_i^{(h)} \equiv \sum_{i \in R} p_i^{(0)}$. This means that at termination, after the last entry in R is set, we have

$$|S \cap R| \in \{ \lfloor \sum_{i \in R} p_i^{(0)} \rfloor, \lceil \sum_{i \in R} p_i^{(0)} \rceil \}.$$

Hence, the maximum range discrepancy is $\Delta < 1$.

EXAMPLE 1. Figure 1 contains an example of structure-aware VAROPT sampling of a hierarchy. It shows the weight of each leaf and respective IPPS inclusion probability for a sample of size 4. For each internal node we show the expected number of samples. The table shows a sequence of pairwise aggregations which follows the hierarchy pair selection rule. The result is a structure-aware VAROPT sample, consisting of the leaves $S = \{2, 3, 7, 10\}$.

One can verify that the example has the promised property that the number of samples under each internal nodes is indeed the floor or ceiling of the expected number. A VAROPT sample that is not structure aware would include the same total number of keys, but could significantly deviate from the expectation at internal nodes. A Poisson IPPS sample has the same expected number of keys but the size varies (could include all keys or no key). For Poisson sampling, the sample distribution depends only on the set of weights and is structure-oblivious.

Query accuracy. We now bound the discrepancy for a query that is a union of several ranges:

LEMMA 3. Let Q be a union of ℓ disjoint ranges R_1, \dots, R_ℓ on a hierarchy structure. The discrepancy is at most ℓ and is distributed as the error of a VAROPT sample on a subset of size $\mu = \sum_{h=1}^{\ell} (p(R_h) - \lfloor p(R_h) \rfloor) \leq \ell$.

PROOF. Consider a truncated hierarchy where the nodes R_1, \dots, R_ℓ are leaves. Include other nodes to form a set L' of disjoint nodes

Algorithm 3 OSSUMMARIZE(p_1, \dots, p_n)

```
1:  $a \leftarrow 1$  ▷ leftover key
2:  $i = 2$  ▷ current key
3: while  $i \leq n$  do ▷ left to right scan of keys
4:   while  $p_a = 1$  and  $a < n$  do
5:      $a++$ ;

6:    $i \leftarrow a + 1$ 
7:   while  $p_i = 1$  and  $i < n$  do
8:      $i++$ ;

9:   if  $p_a < 1$  and  $p_i < 1$  then
10:    PAIR-AGGREGATE( $a, i$ )
11:    if  $p_a = 1$  or  $p_a = 0$  then ▷  $p_a$  is set
12:       $a \leftarrow i$  ▷  $i$  is the new leftover key
13:       $i++$ 
```

which covers all original leaves. For each leaf node in the truncated hierarchy $R_h \in L'$ consider a corresponding “leftover” 0/1 random variable with probabilities $p(R_h) - \lfloor p(R_h) \rfloor$: its value is 1 if the range R_h has $\lceil R_h \rceil$ samples, and its value is 0 if there are $\lfloor R_h \rfloor$ samples. It follows directly from our construction that the sample with respect to these leftovers is a VAROPT sample, since the original summarization from L' up proceeds like a hierarchy summarization, treating the leftovers as leaves. \square

Applying Chernoff bounds, we obtain that the discrepancy on Q is at most $\sqrt{\ell}$ with high probability.

5.2 Ordered Ranges

THEOREM 4. *For the order structure (all prefixes of ordered keys), there always exists a VAROPT sample distribution with maximum range discrepancy $\Delta \leq 2$. For any fixed $\Delta < 2$, there exist inputs for which a VAROPT sample distribution with maximum range discrepancy $\leq \Delta$ does not exist.*

We establish the positive part of the Theorem by presenting an efficient summarization algorithm. Appendix A shows the negative part by demonstrating a family of hard input instances.

Order Structure Summarization. To gain intuition, first consider inputs where there is a partition \mathcal{L} of the ordered keys into non-overlapping intervals such that for each interval $J \in \mathcal{L}$, $p(J) = \sum_{i \in J} p_i \equiv 1$, i.e. the initial probabilities sum to 1. In this case, there are VAROPT samples which pick one key from each interval $J \in \mathcal{L}$. Now observe that any query interval R covers some number of full unit intervals. The only discrepancy comes from the at most 2 end intervals, and so the maximum range discrepancy is bounded by the probability mass therein, $\Delta < 2$. Therefore, this sample has maximum range discrepancy $\Delta < 2$.

The general case is handled by OSSUMMARIZE(p_1, \dots, p_n). (Algorithm 3). Without loss of generality, key i is the i th key in the sorted order, and p_i is its inclusion probability. The algorithm processes the keys in sorted order, maintaining an active key a that is the only key that is not set from the prefix processed so far. At any step, if there is an active key a , the selected pair for the aggregation consists of a and the current key. Otherwise, the aggregation involves the current and the next key. The final sample S is the set of keys with $p_i = 1$.

The algorithm can be viewed as a special case of a hierarchy summarization algorithm where the ordered keys are arranged as a hierarchy which is a long path with a single leaf hanging out of

each path node. The internal nodes in the hierarchy correspond to prefixes of the sorted order, and thus they are estimated optimally (the number of samples is the floor or ceiling of the expectation): For any i , the number of members of S amongst the first i keys is

$$|S \cap [1, i]| \in \left\{ \left\lfloor \sum_{j \leq i} p_j \right\rfloor, \left\lceil \sum_{j \leq i} p_j \right\rceil \right\}.$$

For a key range $R = [i_1, i_2]$ that is not a prefix ($i_2 \geq i_1 > 1$), we can express it as the difference of two prefixes:

$$\begin{aligned} |S \cap R| &= |S \cap [1, i_2]| - |S \cap [1, i_1 - 1]| \\ &\leq \left\lceil \sum_{j \leq i_2} p_j \right\rceil - \left\lfloor \sum_{j \leq i_1 - 1} p_j \right\rfloor \leq 1 + \sum_{j \leq i_2} p_j - (-1 + \sum_{j \leq i_1 - 1} p_j) \\ &= 2 + \sum_{i_1 \leq j \leq i_2} p_j. \\ &\geq \left\lfloor \sum_{j \leq i_2} p_j \right\rfloor - \left\lceil \sum_{j \leq i_1 - 1} p_j \right\rceil \geq -1 + \sum_{j \leq i_2} p_j - (1 + \sum_{j \leq i_1 - 1} p_j) \\ &= -2 + \sum_{i_1 \leq j \leq i_2} p_j. \end{aligned}$$

Hence the maximum discrepancy is at most 2, as claimed.

Summaries with Smaller discrepancy. Requiring the summary to be VAROPT means that it may not be feasible to guarantee Δ strictly less than 2. We can, however, obtain a *deterministic* set with maximum range discrepancy $\Delta < 1$: Associate key i with the interval $H_i = (\sum_{j < i} p_j, \sum_{j \leq i} p_j]$ on the positive axis (with respect to the original vector of inclusion probabilities) and simply include in S all keys where the H_i interval contains an integer. In fact, we can obtain a sample which satisfies the VAROPT requirements (i) and (ii) but not (iii) with $\Delta < 1$: Uniformly pick $\alpha \in [0, 1]$ and store all keys i so that $h + \alpha \in H_i$ for each integer h . This sampling method is known as *systematic sampling* [20]. Systematic samples, however, suffer from positive correlations which mean that estimates on some subsets have high variance (and Chernoff tail bounds do not apply).

6. PRODUCT STRUCTURES

We now consider summarizing multi-dimensional data, where the key structure projected on each axis is an order or a hierarchy. Ranges are axis parallel hyper rectangles: a product of one-dimensional key ranges (order) and/or internal nodes of a hierarchy.

We develop a VAROPT sampling algorithm where the discrepancy on a range R is that of a (structure oblivious) VAROPT sample on a subset with $\mu \leq \min\{p(R), 2ds^{\frac{d-1}{d}}\}$. Hence, the estimation error is subjected to tail bounds (3) and (4) with this value of μ and concentrated around $\sqrt{\mu} \leq \min\{\sqrt{p(R)}, \sqrt{2ds^{\frac{d-1}{2d}}}\}$.

6.1 Algorithm

As in the one-dimensional case, the intuition behind our approach is to limit the range discrepancy by preferring pairwise aggregations that result in “localized” movement of “probability mass.”

Uniform case. We start with the special case of a uniform distribution over a d -dimensional hyper cube with measure $s = h^d$. Our algorithm partitions the hypercube into s unit cells and selects the sample by independently picking a single point uniformly from each unit cell. The resulting sample S is a VAROPT sample (of size s) of the uniform hypercube. For analysis, observe that any axis-parallel hyperplane intersects at most $h^{d-1} = s^{\frac{d-1}{d}}$ unit cells. Therefore, any axis-parallel box query R intersects at most $2ds^{(d-1)/d}$ cells that are not contained in R . The only error in

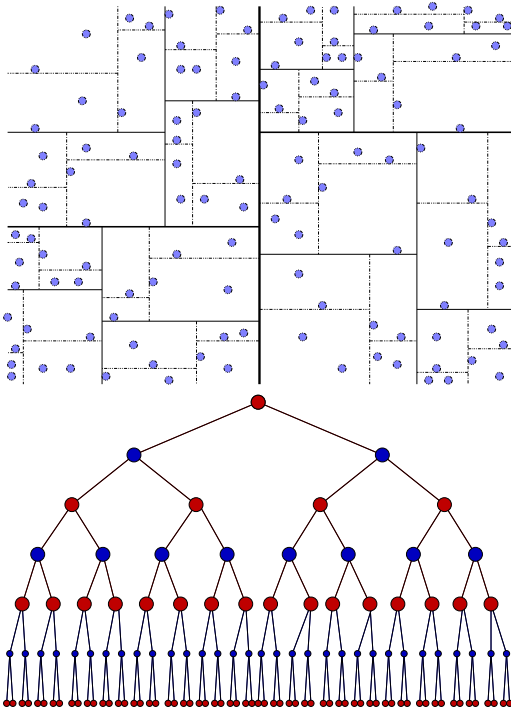


Figure 2: KD Hierarchical partition of two dimensional data

our estimation comes from these “boundary” cells which we denote $B(R)$: all other cells are either fully inside or fully outside R , and so do not contribute to the discrepancy. We map each boundary cell $C \in B$ to a 0/1 random variable which is 1 with probability proportional to the size of the overlap, $|C \cap R|$. These random variables are independent Poisson with $\mu = \sum_{C \in B} |C \cap R| \leq \min\{p(R), |B(R)|\}$, and so the tail bounds hold.

General Case. In general, the probability mass is not distributed uniformly throughout the space as in the previous case. So instead, we seek to build a partition of the space into regions so that the probability mass is (approximately) equal. In particular, we consider using kd-trees to obtain a partition into cells containing keys whose sum of probabilities is $O(1)$ (in general it is not possible to partition the discrete probabilities to sum to exactly 1). Choosing kd-trees means that every axis-parallel hyperplane intersects $O(s^{\frac{d-1}{d}})$ cells. Since cells are not exact units, we have to carefully account for aggregations of the “leftover” probabilities.

Let K be the input set of weighted multi-dimensional keys. Then:

- Compute IPPS inclusion probabilities for K and set aside all keys with $p_i = 1$ (they must all be included in the summary). Hence, wlog, we have that all keys in K have $p_i < 1$.
- Compute a hierarchy T over the (multidimensional) keys in K :

$$T \leftarrow \text{KD-HIERARCHY}(0, K) .$$

- Apply the hierarchy summarization algorithm (Section 5) to T .

Algorithm KD-HIERARCHY builds a kd-tree, splitting on each dimension in turn. At each internal node we select a hyperplane perpendicular to the current axis that partitions the probability weight in half (or as equally as possible.) Figure 2 shows a two-dimensional set of keys and the corresponding hierarchy, a balanced binary tree of depth 6. The cuts alternate horizontally and vertically. Each leaf

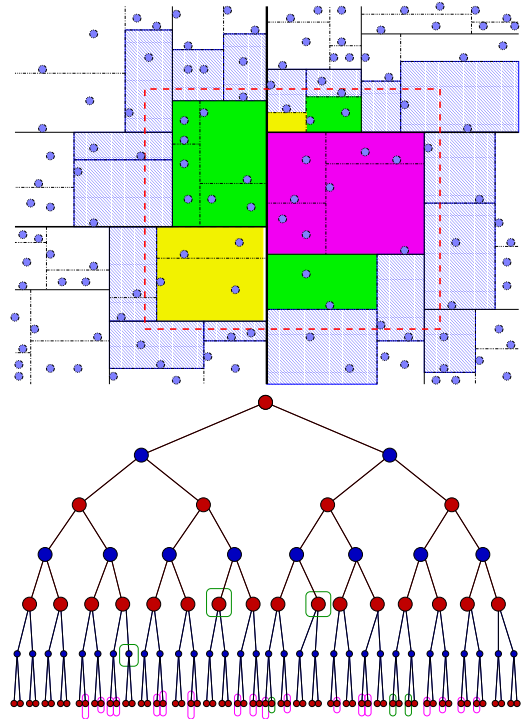


Figure 3: Query rectangle on a hierarchical partition

of the tree therefore corresponds to an undivided rectangle containing approximately unit probability mass.

6.2 Analysis of KD-hierarchy

We now analyze the resulting summary, based on the properties of the space partitioning performed by the kd-tree. We use v to refer interchangeably to a node in the tree and the hyperrectangle induced by node v in the tree. A node v at depth d in the tree has probability mass $p(v) \leq s/2^d + 2$. We refer to the set of minimum depth nodes that satisfy $p(v) \leq 1$ as *super leaves* (v is a super leaf iff $p(v) \leq 1$ and its immediate ancestor $a(v)$ has $p(a(v)) > 1$). The depth of a super-leaf (and of the hierarchy when truncated at super-leaves) is at most $D = 2 + \lceil \log_2 s \rceil$.

LEMMA 5. *Any axis-parallel hyperplane intersects at most $s^{\frac{d-1}{d}}$ super-leaves.*

PROOF. Consider the hierarchy level-by-level, top to bottom. Each level where the axis was not perpendicular to the hyperplane at most doubles the number of nodes that intersect the hyperplane. Levels where the partition axis is perpendicular to the hyperplane do not increase the number of intersecting nodes. Because axes were used in a round-robin fashion, the fraction of levels that can double the number of intersecting nodes is $\frac{d-1}{d}$. Hence, when we reach the super-leaf level, the number of intersecting nodes is at most $2^{\frac{d-1}{d}D} = O(s^{\frac{d-1}{d}})$. \square

As an immediate corollary, we obtain that the boundary of an axis-parallel box R may intersect at most $2ds^{\frac{d-1}{d}}$ super-leaves. We denote by $B(R)$ this set of boundary super leaves. Let $U(R)$ be a minimum size collection of nodes in the hierarchy such that no internal node contains a leaf from $B(R)$. Informally, $U(R)$ consists of the (maximal) hyperrectangles which are fully contained in R or fully disjoint from R . Figure 3 illustrates a query rectangle R ,

Algorithm 4 KD-HIERARCHY($depth, key_set$)

```
1: if  $|key\_set| = 1$  then
2:    $h.val \leftarrow key\_set$ 
3:    $h.left \leftarrow null; h.right \leftarrow null;$ 
4:   return  $h$   $\triangleright$  kd-hierarchy  $h$  is a leaf node
5: else
6:    $h.val \leftarrow null$ 
7:    $a \leftarrow depth \bmod d$   $\triangleright$  axis on which partition is made
8:   if axis  $a$  has an order structure then
9:      $m \leftarrow \arg \min_m \left| \sum_{i|key_a(i) \leq m} p_i - \sum_{i|key_a(i) > m} p_i \right|$ 
 $\triangleright m$  is the weighted median of  $key\_set$  when ordered according to axis  $a$ 
10:     $left\_set \leftarrow \{i|key_a(i) \leq m\};$ 
11:     $right\_set \leftarrow \{i|key_a(i) > m\};$ 
12:    else  $\triangleright$  axis  $a$  has a hierarchy structure  $H_a$ 
13:    find an internal node  $v \in H_a$  and a 2-partition of
     $key\_set$  into  $left\_set$  and  $right\_set$  according to a 2-partition of
    the components of  $H_a$  obtained by  $v$ 's removal, with minimum
     $\left| \sum_{i \in left\_set} p_i - \sum_{i \in right\_set} p_i \right|$ 
14:     $h.left \leftarrow KD-HIERARCHY(depth + 1, left\_set)$ 
15:     $h.right \leftarrow KD-HIERARCHY(depth + 1, right\_set)$ 
16:    return  $h$ 
```

the boundary $B(R)$, and all nodes $v \in U(R)$ such that $v \subset R$. The maximal interior nodes contained in R ($v \in U(R)|v \subset R$) are marked in solid colors (green circles in the tree layout) and the boundary super leaves $B(R)$ in stripes (magenta circles in the tree layouts).

LEMMA 6. *The size of $U(R)$ is at most $O(ds^{\frac{d-1}{d}} \log s)$.*

PROOF. Each node in U must have a sibling such that the sibling, or some of its descendants, are in $B(R)$. If this is not the case, then the two siblings can be replaced by their parent, decreasing the size of $U(R)$, which contradicts its minimality. We bound the size of $U(R)$ by bounding the number of potential siblings. The number of ancestors of each boundary leaf is at most the depth which is $\leq 2 + \lceil \log_2 s \rceil$. Thus, the number of potential siblings is at most the number of boundary leaves times the depth. By substituting a bound on $|B(R)|$, we obtain the stated upper bound. \square

These lemmas allow us to bound the estimation error, by applying Lemma 3. That is, for each $v \in U(R)$ such that $v \subset R$ we have a 0/1 random variable that is 1 with probability $p_v - \lfloor p_v \rfloor$ and is 0 otherwise (The value is 0 if v includes $\lfloor p_v \rfloor$ samples and 1 otherwise). For each $v \in B(R)$, we have a random variable that is 1 with probability $p(v \cap R)$. This is the probability that S contains one key from $v \cap R$ (the sample can contain at most one key from each superleaf). The sample is VAROPT over these random variables with

$$\mu = \sum_{v \in U(R)|v \subset R} (p(v) - \lfloor p(v) \rfloor) + \sum_{v \in B(R)} p(v \cap R) \leq |U(R)| + |B(R)|.$$

Substituting our bounds on $|U(R)|$ and $|B(R)|$ from Lemmas 5 and 6 gives accuracy bound claimed at the start of the section.

7. I/O EFFICIENT SAMPLING

The algorithms presented in previous sections assume that we can hold the full data set in memory to generate the summary. As data sets grow we require summarization methods that are more

Algorithm 5 IO-AGGREGATE(i)

```
Process key  $i$ :
1:  $p_i \leftarrow \min\{1, w_i/\tau_s\},$ 
2: if  $p_i = 1$  then  $\triangleright i$  is included in sample
3:    $S \leftarrow S \cup \{i\}$ 
4: if  $a(L(i)) = \emptyset$  then  $\triangleright$  There is no active key in  $L(i)$  (an active key has current  $p_a \in (0, 1)$ ),  $i$  becomes the active key
5:    $a(L(i)) \leftarrow i$ 
6: else  $\triangleright L(i)$  has an active key
7:    $a \leftarrow a(L(i))$ 
8:   PAIR-AGGREGATE( $p, i, a$ )
9: if  $p_a = 0$  then
10:   $a(L(i)) \leftarrow \emptyset$ 
11:  if  $p_i < 1$  then
12:     $a(L(i)) \leftarrow i$ 
13:  else  $\triangleright p_i = 1$ 
14:     $S \leftarrow S \cup \{i\}$ 
15:  else
16:    if  $p_a = 1$  then
17:       $S \leftarrow S \cup \{a\}$ 
18:      if  $p_i = 0$  then
19:         $a(L(i)) \leftarrow \emptyset$ 
20:      else  $\triangleright p_i \neq 0$ 
21:         $a(L(i)) \leftarrow i$ 
22:      else  $\triangleright 0 < p_a < 1$ 
23:        if  $p_i = 1$  then
24:           $S \leftarrow S \cup \{i\}$ 
```

I/O efficient. In particular, the reliance on being able to sort data, locate data in hierarchies, and build kd-trees may not be realistic for large data sets. In this section, we present I/O efficient alternatives that generate a structure-aware VAROPT sample while only slightly compromising on range discrepancy with respect to the main-memory variants. The intuition behind our approach is that a structure-oblivious VAROPT sample of sufficient size is useful to guide the construction of a structure-aware summary because with high probability it “hits” all sufficiently large ranges.

Overview. Our algorithms perform two read-only passes over the (unsorted) input dataset. When using memory of size $\tilde{O}(s)$ (where s is the desired sample size), the range discrepancy is similar to that of the main memory algorithm with probability $1 - 1/\text{poly} \log s$. In the first pass we compute a (classic) random sample S' of size $s' > s$ using memory s' via Poisson IPPS sampling or stream VAROPT (reservoir sampling if keys are not weighted). After completion of the first pass, we use S' to construct a partition \mathcal{L} of the universe of key. In the second pass we perform probabilistic aggregations, guided by the partition \mathcal{L} , using IPPS probabilities for a sample of size s . We maintain at most one active key $a(L)$ for each cell $L \in \mathcal{L}$. When processing a key i , if there is no active key in the cell of i , then i becomes the active key. If there is an active key a , we pair aggregate i and a . If the updated p value of one of them becomes 1, we include this key in the sample S . The key with p value in $(0, 1)$ (if there is one) is the new active key for the current cell. The storage required is $O(s + |\mathcal{L}|)$, since we maintain at most one active key for each part and the number of keys included in S is at most s . Finally, after passing over the whole data, we perform pair aggregation on the remaining active keys.

Detailed Description. In the first pass, we compute a VAROPT (or Poisson IPPS) sample S' (of size $s' = |S'|$) of the data set. In par-

allel, compute the IPPS threshold value τ_s to yield a sample of size s (described in Section 2). In main memory, we use S' to construct a partition \mathcal{L} of the universe of keys. Let $L(i) \in \mathcal{L}$ denote the cell containing key i . In the second pass, we begin with an empty sample S and no active keys in each cell of \mathcal{L} (so $a(L) = \emptyset$ for all $L \in \mathcal{L}$). For each key, we apply the algorithm IO-AGGREGATE, in Algorithm 5. Lastly, we aggregate all the leftover active keys of different cells (there is at most one such key per cell).

The initial random sample S' guides the formation of the partition \mathcal{L} . A sample of size $S' = O(s \log s)$ suffices to hit *all* sufficiently large ranges (those with combined probability at least 1) with high probability. (In geometric terms, it forms an ϵ -net of the range space [12]). This property enables the sample to form an effective partition \mathcal{L} . The partition has the property that with high probability $p(L) \leq 1$ for all $L \in \mathcal{L}$.

Partition and aggregation choices. The specifics of the main-memory operations—the construction of the partition and the final aggregation of active keys—depend on the range structure considered. We start with product structures and then refine the construction to obtain stronger results for one-dimensional structures.

Product structures: We (temporarily) remove from S' all keys with $\min\{1, w_i/\tau_s\} = 1$ and compute $h \leftarrow \text{KD-HIERARCHY}(0, S')$. This hierarchy h induces a partition according to the splitting criteria in each node. The partition \mathcal{L} corresponds to the leaves of h . The aggregation of active keys in the final phase follows the hierarchy h .

Disjoint ranges: There is a cell in the partition for every range from \mathcal{R} that is contains some key present in the sample. We then induce an arbitrary order over the ranges and put a cell for each union of ranges in \mathcal{R} which lies between two consecutive ranges represented in the sample. In total we obtain $2s'$ cells. In the final phase, active keys can be aggregated arbitrarily.

When $s' = \Omega(s \log s)$, with probability $1 - \text{poly}(1/s)$, all ranges of size ≥ 1 are intersected by S' and each cell L that is a union of ranges not seen in S' has size at most 1 (thus, each range R with probability mass $p(R) < 1$ can obtain at most one sample in S , and so will not be over-represented in the final sample). Thus, the maximum discrepancy is $\Delta < 1$ with probability $1 - \text{poly}(1/s)$.

Order: We sort S' according to the order and exclude keys with $w_i \geq \tau_s$. These keys are placed directly in S —as S' includes all keys with $w_i \geq \tau_{s'}$ and since $\tau_s > \tau_{s'}$, it must include all keys with $w_i \geq \tau_s$. If i_1, \dots, i_t are the remaining keys in sorted order, there is a cell L for each subinterval $(i_j, i_{j+1}]$ and two more, one for all keys $\leq i_1$ and the other for keys $> i_t$. The final aggregation of active keys follows the main-memory aggregation of ordered keys.

When $s' = \Omega(s \log s)$, with high probability, the maximum probability distance between consecutive keys is 1 and therefore, the maximum range discrepancy is $\Delta < 2$ with high probability.

Hierarchy: A natural solution is to linearize the hierarchy, i.e. generate a total order consistent with the hierarchy, and then apply the algorithm for this order structure. This obtains $\Delta < 2$ with high probability. Alternatively, we can select all ancestors in the hierarchy of keys in S and form a partition by matching each key to its lowest selected ancestor. This will give us maximum range discrepancy $\Delta < 1$ with high probability. The number of ancestors, however, can be large and therefore this approach is best for shallow hierarchies.

8. EXPERIMENTAL STUDY

We conducted an experimental study of our methods, and compared them with existing approaches to summarizing large data sets.

8.1 Experimental Environment

Data Sets and Query Sets. We compared our approach on a variety of 2D data sets, and present results on two examples:

The *Network* dataset consists of network flow data summarizing traffic volumes exchanged between a large number of sources and destination, observed at a network peering point. In total, there are 63K sources and 50K destinations, and a total of 196K pairs active in the data. The space of these records is determined by the IP hierarchy, i.e. $X=2^{32}$ and $Y=2^{32}$.

The *Technical Ticket* data is derived from calls to a customer care center of a broadband wireline access network that resulted in a technical trouble ticket. Each key consists of: (i) an anonymous identifier for unique customers; (ii) a trouble code, representing a point in a hierarchy determining the nature of the problem identified from a predetermined set by the customer care staff; and (iii) a network code, indicating points on the network path to the customer’s location. Both attributes are hierarchical with varying branching factor at each level, representing a total of approximately 2^{24} possibilities in each dimension, i.e. $X = 2^{24}$ and $Y = 2^{24}$. There are 4.8K distinct trouble codes present in the data, 80K distinct network locations, and 500K distinct observed combinations.

Each query is produced as a collection of non-overlapping rectangles in the data space. To study the behavior of different summaries over different conditions, we generated a variety of queries of two types. In the *uniform area* case, each rectangle is placed randomly, with height and width chosen uniformly within a range $[0, h]$ and $[0, w]$. We tested a variety of scales to determine h and w , varying these from covering almost the whole space, to covering only a very small fraction of the data (down to a 10^{-4} fraction). In the *uniform weight* case, each rectangle is chosen to cover roughly the same fraction of the total weight of keys. We implement this by building a kd-tree over the whole data, and picking cells from the same level (note, this is independent of any kd-tree built over sampled data by our sampling methods). For each query, we compute the exact range sum over the data, and compare the absolute, sum-squared and relative errors of our methods across a collection of queries. In our plots below, we show results from a battery of 50 queries with varying number of rectangles.

Methods. We implemented the structure-aware sampling method in Python. For comparison, we also implemented examples of various other approaches that have been proposed, in the same language. The methods we compared are as follows:

Aware, the structure aware sampling method. We follow the 2 pass algorithm (Section 7 and Section 6), and first draw a sample of size several times larger than s , the target size (in our experiments, we set $s' = 5s$: increasing the factor did not significantly improve the accuracy). We then built the kd-tree on this sample, and took a second pass over the data to populate the tree. Lastly, we perform pair aggregation within the tree structure to produce a sample of size exactly s .

Obliv, is a structure-oblivious sampling method. We implemented VAROPT sampling to give a sample size of exactly s .

Wavelet, implements the (two-dimensional) standard Haar wavelet scheme. In a single pass we compute the full wavelet transform of the data: each input data point contributes to $\log X \cdot \log Y$ wavelet coefficients. We then prune these coefficients to retain only the w largest (normalized) coefficients for query answering.

Qdigest, implements the (two-dimensional) q-digest data structure [13]. This deterministically builds a summary of the data. Given a

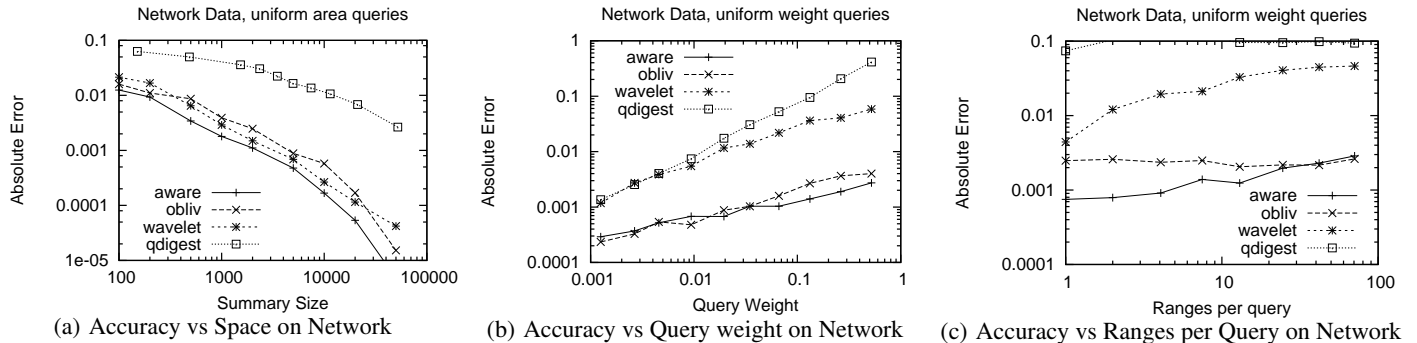


Figure 4: Experimental results on Network Data set

parameter ϵ , the structure is promised to be at worst $O(\frac{1}{\epsilon^2} \log X \cdot \log Y)$, but in practice materializes much fewer nodes than this, so we count the number of materialized nodes as its size.

Sketch, implements the Count-sketch, a randomized summary of the data [4]. Each input item contributes to $O(\log X \cdot \log Y)$ sketches of dyadic rectangles. We choose the size of each sketch so that the total size is bounded by a parameter s .

Our implementations were run on the same machine, on a 2.4GHz core with access to 4GB of memory.

8.2 Network Data Accuracy

Figure 4 shows accuracy results on network data. The y-axis shows accuracy measured as the error in the query results divided by the total weight of all data (the absolute error). Our experiments which computed other metrics such as sum-squared error showed the same trends, and so are omitted.

On this data, the structure-aware sampling typically achieved the minimal error. Figure 4(a) shows this behavior across a variety of summary sizes with uniform area queries each containing 25 ranges. For comparison, we measure the space used by each summary in terms of elements on the original data, so in this case the smallest sample contains 100 IP address pairs (and weights). This is compared to keeping the 100 largest wavelet coefficients, and a q-digest of 100 nodes. We also compared to sketch summaries with an equivalent amount of space. However, the space required before a sketch of two-dimensional data becomes accurate is much larger than for the other summaries considered. The total error for the queries shown was off the scale on the graphs, so we omit sketches from further accuracy comparison.

Across most summary sizes, the structure-aware sampling is several times more accurate than its structure-oblivious counterpart (note that Figure 4(a) is in log-scale on both axes). The deterministic q-digest structure is one to two orders of magnitude less accurate in the same space. Only the Haar wavelet comes close to structure-aware sampling in terms of accuracy. This is partly due to the nature of uniform area queries: on this data, these correspond to either very small or very large weight.

Figure 4(b) shows the accuracy under uniform weight queries, where each query contains 10 ranges of approximately equal weight. The graph shows the results for a fixed summary size of 2700 keys, corresponding to a total summary size of approximately 32KB. Over this query set, the accuracy of sampling methods is improved, while wavelets and q-digest are less accurate: note that the error of q-digest is close to the total weight of the query. The general trend is for the absolute error to increase with the fraction of data

included in the sample. However, note that when the gradient of these lines is shallow, the *relative* error is actually improving, as our analysis indicates: the structure aware summary obtains 0.001 error on a query that covers more than 0.1 of the whole data weight, meaning that the observed relative error is less than 1%.

Figure 4(c) shows the case when we hold the total weight of the query steady (in this case, at approximately 0.12 of the total data weight), and vary the number of ranges in each query. We see that the structure oblivious accuracy does not vary much: as far as the sample is concerned, all the queries are essentially subset queries with similar weight. However, on a small number of ranges, the structure aware sampling can be several times better. As the number of ranges per query increase, each range becomes correspondingly smaller, and so for queries with 40 or more ranges there is minimal difference between the structure aware and structure oblivious. On this query set, wavelets are an order of magnitude less accurate, given the same summary size.

8.3 Scalability

To study scalability of the different methods, we measured the time to build the summaries, and the time to answer 2500 range queries. Figure 5 shows these results on the network data, as we vary the size of the summary. Although our implementations are in an interpreted language, and not honed for performance, we believe that these results show the general trend of the costs. Structure-oblivious is the fastest, requiring essentially negligible time to build (Figure 5(a)). Structure-aware sampling still completes in a matter of seconds: the extra time is due to the cost of locating nodes in the kd-tree. We expect that a more engineered implementation could reduce this building cost further.

The q-digest and sketch summaries both take around 100 seconds to build the summary. These structures are quite fast in one-dimension, but have to do more work in higher dimensions. For example, the sketch needs to update a number of locations which grows with the product of the logarithm of the dimension sizes. On pairs of 32 bit addresses, this factor is proportional to $32^2=1024$. The cost of building the 2D Haar wavelet summary is 4 orders of magnitude more than sampling. The reason is that each point in the data contributes to 1024 coefficients. In our implementation, we generate the exact transform of the data before thresholding, which is very large, and access to the coefficients as they are built is essentially random, so there is little benefit from caching.

A similar trend is seen for the query time (Figure 5(b)). Since the samples, once built, have the same form, query answering takes the same time with both obliv and aware: we just pass through and compute the intersection of the sample with each query rectangle.

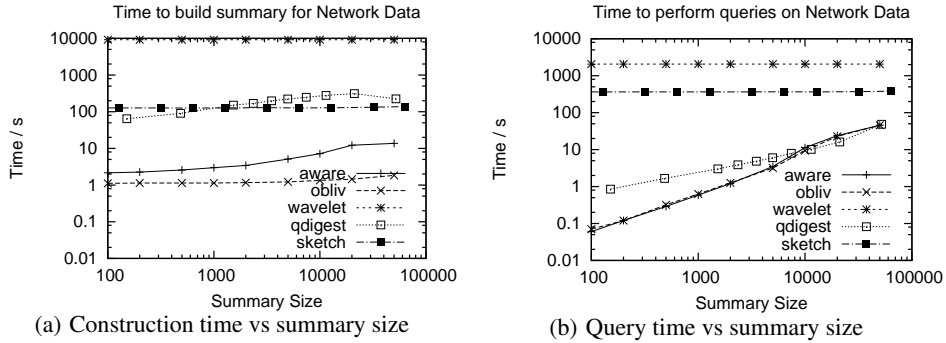


Figure 5: Time costs as summary size varies

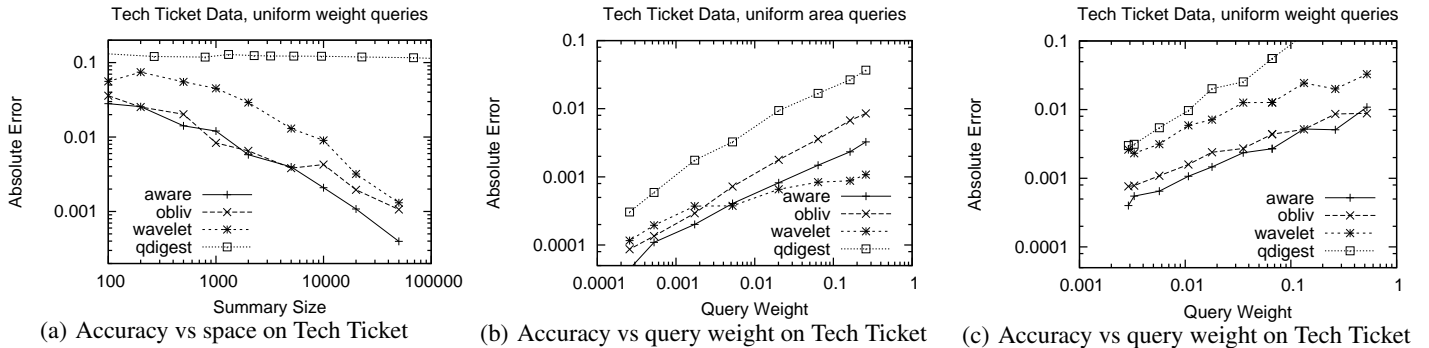


Figure 6: Accuracy on the Tech Ticket Data

The cost grows with increasing sample size, as we are just scanning through the sample to find which points fall within each rectangle. Still, this naive approach to query answering can process thousands of query rectangles per second (recall, the y-axis is showing the time to complete the battery of 2500 rectangle queries). In comparison, asking this many queries over the full data takes 2 minutes. Again, we see the disadvantage of the wavelet approach: each rectangle query takes of the order of a second—in other words, it is about 1000 times slower than using a sample. The reason is that each rectangle is decomposed into dyadic rectangles. In the worst case, there are 1000 dyadic rectangles formed, and each dyadic rectangle requires looking up the value of 1000 coefficients. The effect as we go to higher dimensions only gets worse, growing exponentially with the dimension. While there should be more efficient ways to use wavelets, the overall cost is offputtingly high.

8.4 Tech Ticket Data Accuracy

The plots in Figure 6 show accuracy experiments on the tech ticket data set. Here, the relative performance is less clear cut, but the conclusion is similar. Figure 6(a) shows that structure-aware and structure-oblivious sampling behave similarly for smaller sample sizes: this is because this data set has many high weight keys which must be included in both samples. For large sample sizes, the methods diverge, and the benefit of structure awareness is seen. On this data set, this occurs for samples that are between 1% and 10% of the data size.

The next two plots compare the case for uniform area queries (over 25 ranges, Figure 6(b)) and uniform weight queries (10 ranges, Figure 6(c)). We see that on uniform area queries, wavelets can become competitive for higher weights, but this is not seen when the

weight of each range is controlled. In either case, for these queries of several ranges, structure-aware sampling seems to give the best results overall. Certainly, wavelets do not seem to scale with this type of data: tens to hundreds of millions of coefficients are generated before thresholding, leading to a high time and space cost. The timing results for this dataset are similar to the previous case: generating and using samples takes seconds, while using wavelets takes (literally) hours.

9. CONCLUDING REMARKS

We introduce structure-aware sampling as an alternative to structure-oblivious sampling and tailored deterministic summaries. Our structure-aware samples are VAROPT—they retain the full benefits of state-of-the-art sampling over deterministic summaries: flexibility and support for arbitrary subset queries, accuracy on these queries, unbiased estimation, and exponential tail bounds on the error. By optimizing the sample distribution for range queries, we obtain superior accuracy with respect to structure-oblivious samples and are able to match or surpass the accuracy of tailored deterministic summaries.

Our proposed algorithms are simple to implement and are I/O efficient, requiring only two sequential read passes over the data and memory which is independent of the input size. In future work, we aim to make our samples even more scalable by proposing variants which require only a single (streaming) pass over the data, instead of (at least) two: this would enable building samples of data which is never stored in full.

10. REFERENCES

- [1] A. Bagchi, A. Chaudhary, D. Eppstein, and M. T. Goodrich. Deterministic sampling and range counting in geometric data streams. *ACM Trans. Algorithms*, 3(2):16, 2007.
- [2] H. Brönnimann, B. Chen, M. Dash, P. Haas, and P. Scheuermann. Efficient data reduction with EASE. In *KDD*, 2003.
- [3] M. T. Chao. A general purpose unequal probability sampling plan. *Biometrika*, 69(3):653–656, 1982.
- [4] M. Charikar, K. Chen, and M. Farach-Colton. Finding frequent items in data streams. In *ICALP*, 2002.
- [5] 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.
- [6] E. Cohen, N. Duffield, H. Kaplan, C. Lund, and M. Thorup. Stream sampling for variance-optimal estimation of subset sums. In *SODA*, 2009.
- [7] E. Cohen, N. Duffield, C. Lund, M. Thorup, and H. Kaplan. Variance optimal sampling based estimation of subset sums. arXiv:0803.0473v1, 2008.
- [8] 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.
- [9] 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.
- [10] D. Gunopulos, G. Kollios, V. J. Tsotras, and C. Domeniconi. Approximating multi-dimensional aggregate range queries over real attributes. In *SIGMOD*, 2000.
- [11] J. Hájek. *Sampling from a finite population*. Marcel Dekker, New York, 1981.
- [12] D. Haussler and E. Welzl. Epsilon nets and simplex range queries. *Discrete Comput. Geom.*, 2, 1987.
- [13] J. Hersherberger, N. Shrivastava, S. Suri, and C. D. Tóth. Adaptive spatial partitioning for multidimensional data streams. In *ISAAC*, 2004.
- [14] 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.
- [15] J.-H. Lee, D.-H. Kim, and C.-W. Chung. Multi-dimensional selectivity estimation using compressed histogram information. *SIGMOD Rec.*, 28(2):205–214, 1999.
- [16] Y. Matias, J. S. Vitter, and M. Wang. Wavelet-based histograms for selectivity estimation. In *SIGMOD*, 1998.
- [17] 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.
- [18] J. M. Phillips. Algorithms for epsilon-approximations of terrains. In *ICALP*, 2008.
- [19] V. Poosala and Y. Ioannidis. Selectivity estimation without the attribute value independence assumption. In *VLDB*, 1997.
- [20] C.-E. Särndal, B. Swensson, and J. Wretman. *Model Assisted Survey Sampling*. Springer, 1992.
- [21] N. Shrivastava, C. Buragohain, D. Agrawal, and S. Suri. Medians and beyond: new aggregation techniques for sensor networks. In *SenSys*, 2004.
- [22] A. Srinivasan. Distributions on level-sets with applications to approximation algorithms. In *FOCS*, 2001.
- [23] S. Suri, C. D. Tóth, and Y. Zhou. Range counting over multidimensional data streams. In *SCG*, 2004.
- [24] M. Szegedy and M. Thorup. On the variance of subset sum estimation. In *ESA*, 2007.
- [25] Y. Tillé. *Sampling Algorithms*. Springer, 2006.
- [26] V. Vapnik and A. Y. Chervonenkis. On the uniform convergence of relative frequencies of events to their probabilities. *Theory of Probability and its applications*, 16:264–280, 1971.
- [27] J. S. Vitter, M. Wang, and B. Iyer. Data cube approximation and histograms via wavelets (extended abstract). In *CIKM*, 1998.
- [28] Y. Zhang, S. Singh, S. Sen, N. Duffield, and C. Lund. Online identification of hierarchical heavy hitters: algorithms, evaluation, and applications. In *IMC*, 2004.

APPENDIX

A. LOWER BOUND ON DISCREPANCY FOR ORDER STRUCTURE

We prove the second part of Theorem 4 by showing that for any $\Delta < 2$, there are ordered weighted sets for which we can not obtain a VAROPT summary with maximum range discrepancy Δ .

PROOF OF THEOREM 4. For any positive integer m , we show that for some sequence, there is no VAROPT sample with $\Delta \leq 2 - 1/m$.

We use a sequence where $p_i = \epsilon \ll 1/(4m)$ and $\sum_i p_i \geq 5m$. Let $i_1 < i_2 < i_3, \dots$ be the included keys, sorted by order. With each key i_ℓ we associate the position $s(i_\ell) = \sum_{j \leq i_\ell} p_j$.

We give a proof by contradiction. Consider keys i_ℓ, i_j with $\ell < j$. If an interval contains at most h sampled keys, it must be of size at most $h + \Delta$. If an interval contains at least h sampled keys, it must be of size at least $h - \Delta$.

The interval $[s(i_\ell) - \epsilon, s(i_j)]$, which is of size $s(i_j) - s(i_\ell) + \epsilon$, contains $j - \ell + 1$ sampled keys.

We obtain that $s(i_j) - s(i_\ell) + \epsilon \geq j - \ell + 1 - \Delta$. Rearranging, $s(i_j) \geq s(i_\ell) - \epsilon - \Delta + 1 + j - \ell$.

The interval $(s(i_\ell), s(i_j) - \epsilon)$, which is of size $s(i_j) - s(i_\ell) - \epsilon$, contains $j - \ell - 1$ sampled keys. Hence, $s(i_j) - s(i_\ell) - \epsilon \leq j - \ell - 1 + \Delta$. Rearranging, $s(i_j) \leq s(i_\ell) + \epsilon + j - \ell + \Delta - 1$.

From the above, for $j > \ell$, we have

$$s(i_j) \in (s(i_\ell) + j - \ell - \Delta + 1 - \epsilon, s(i_\ell) + j - \ell + \Delta + 1 + \epsilon). \quad (7)$$

For $j \geq 2$,

$$s(i_j) \in (s(i_1) - \epsilon + j - \Delta, s(i_1) + j - 2 + \Delta + \epsilon). \quad (8)$$

Fixing the first j included keys, i_1, \dots, i_j , the conditional probability on $i_{j+1} = h$ is at most p_h . We have $s(i_{j+1}) \in (s(i_j) + 2 - \Delta - \epsilon, s(i_j) + \Delta + \epsilon)$. Therefore, there must be a positive probability for the event

$$s(i_{j+1}) < s(i_j) + \Delta + \epsilon - (1 - \epsilon) = s(i_j) + 1 - 1/m + 2\epsilon$$

which is contained in the event $s(i_{j+1}) \leq s(i_j) + 1 - 1/(2m)$.

Iterating this argument, we obtain that for all $k > 1$, we must have positive probability for $s(i_k) < s(i_1) + (k-1)(1-1/(2m)) = s(i_1) + k - 1 - (k-1)/2m$. From (8) we have $s(i_k) \geq s(i_1) + k - 1 - (1-1/m) = s(i_1) + k - 2 + 1/m$. Taking $k = 4m$, we get a contradiction. \square