Mergeable Summaries

Pankaj K. Agarwal, Graham Cormode, Zengfeng Huang, Jeff M. Phillips, Zhewei Wei, and Ke Yi

We study the *mergeability* of data summaries. Informally speaking, mergeability requires that, given two summaries on two data sets, there is a way to merge the two summaries into a single summary on the two data sets combined together, while preserving the error and size guarantees. This property means that the summaries can be merged in a way akin to other algebraic operators such as sum and max, which is especially useful for computing summaries on massive distributed data. Several data summaries are trivially mergeable by construction, most notably all the *sketches* that are linear functions of the data sets. But some other fundamental ones like those for heavy hitters and quantiles, are not (known to be) mergeable. In this paper, we demonstrate that these summaries are indeed mergeable or can be made mergeable after appropriate modifications. Specifically, we show that for ε-approximate heavy hitters, there is a deterministic mergeable summary of size $O(1/\varepsilon)$; for ε-approximate quantiles, there is a deterministic summary of size $O((1/\varepsilon)\log(\varepsilon n))$ that has a restricted form of mergeability, and a randomized one of size $O((1/\varepsilon)\log^{3/2}(1/\varepsilon))$ with full mergeability. We also extend our results to geometric summaries such as ε-approximations and ε-kernels. While most of the results in this paper are theoretic in nature, some of the algorithms are actually very simple and even perform better than the previously best known algorithms, which we demonstrate through experiments in a simulated a sensor network.

We also achieve two results of independent interest: (1) we provide the best known randomized streaming bound for ε-approximate quantiles that depends only on ε, of size $O((1/\varepsilon)\log^{3/2}(1/\varepsilon))$, and (2) we demonstrate that the MG and the SpaceSaving summaries for heavy hitters are isomorphic.

1. INTRODUCTION

Data summarization is an important tool for answering queries on massive data sets, especially when they are distributed over a network or change dynamically, as working with the full data is computationally infeasible. In such situations, it is desirable to compute a compact summary S of the data D that preserves its important properties, and to use the summary for answering queries, hence occupying considerably less resources. Since summaries have much smaller size, they answer queries approximately, and there is a trade-off between the size of the summary and the approximation error. A variety of data summaries have been proposed in the past, starting with statistical summaries like heavy hitters, quantile summaries, histograms, various sketches and synopses, to geometric summaries like ε-approximations and ε-kernels, and to graph summaries like distance oracles. Note that the error parameter ε has different interpretations for different types of summaries.

Algorithms for constructing summaries have been developed under several models. At the most basic level, we have the data set D accessible in its entirety, and the summary S is constructed offline. More generally, we often want the summary to be maintained in the presence of updates, i.e., when a new element is added to D, S can be updated to reflect the new arrival without recourse to the underlying D. Much progress has been made on incrementally maintainable summaries in the past years, mostly driven by the study of data stream algorithms. Some applications, especially when data is distributed over a network, call for a stronger requirement on summaries, namely, one should be able to *merge* the ε-summaries of two (separate) data sets to obtain an ε-summary of the union of the two data sets, without increasing the size of the summary or its approximation error. This merge operation can be viewed as a simple algebraic operator like sum and max; it is commutative and associative. We motivate the need for such a merge operation by giving two specific applications.

Motivating Scenario 1: Distributed Computation. The need for a merging operation arises in the MUD (Massive Unordered Distributed) model of computation [Feld-

man et al. 2008], which describes large-scale distributed programming paradigms like MapReduce and Sawzall. In this model, the input data is broken into an arbitrary number of pieces, each of which is potentially handled by a different machine. Each piece of data is first processed by a *local* function, which outputs a message. All the messages are then pairwise combined using an *aggregation* function in an arbitrary fashion, eventually producing an overall message. Finally, a post-processing step is applied. This exactly corresponds to our notion of mergeability, where each machine builds a summary of its share of the input, the aggregation function is the merging operation, and the post-processing step corresponds to posing queries on the summary. The main result of [Feldman et al. 2008] is that any deterministic streaming algorithm that computes a symmetric function defined on all inputs can be simulated (in small space but with very high time cost) by a MUD algorithm, but this result does not hold for indeterminate functions, i.e., functions that may have many correct outputs. Many popular algorithms for computing summaries are indeterminate, so the result in [Feldman et al. 2008] does not apply in these cases.

Motivating Scenario 2: In-network aggregation. Nodes in a sensor network organize themselves into a routing tree rooted at the base station. Each sensor holds some data and the goal of *data aggregation* is to compute a summary of all the data. Nearly all data aggregation algorithms follow a bottom-up approach [Madden et al. 2002]: Starting from the leaves, the aggregation propagates upwards to the root. When a node receives the summaries from its children, it *merges* these with its own summary, and forwards the result to its parent. Depending on the physical distribution of the sensors, the routing tree can take arbitrary shapes. If the size of the summary is independent of $|D|$, then this performs load-balancing: the communication along each branch is equal, rather than placing more load on edges closer to the root.

These motivating scenarios are by no means new. However, results to this date have yielded rather weak results. Specifically, in many cases, the error increases as more merges are done [Manjhi et al. 2005; Manjhi et al. 2005; Greenwald and Khanna 2004; Chazelle and Matousek 1996]. To obtain any overall guarantee, it is necessary to have a bound the number of rounds of merging operations in advance so that the error parameter ε can be scaled down accordingly. Consequently, this weaker form of mergeability fails when the number of merges is not pre-specified; generates larger summaries (due to the scaled down ε); and is not mathematically elegant.

1.1. Problem statement

Motivated by these and other applications, we study the mergeability property of various widely used summarization methods and develop efficient merging algorithms. We use $S()$ to denote a summarization method. Given a data set (multiset) D and an error parameter ε, $S()$ may have many valid outputs (e.g., depending on the order in which it processes D, it may return different valid ε-summaries), i.e., $S()$ could be a one-to-many mapping. We use $S(D, \varepsilon)$ to denote any valid summary for data set D with error ε produced by this method, and use $k(n, \varepsilon)$ to denote the maximum size of any $S(D, \varepsilon)$ for any D of n items.

We say that $S()$ is *mergeable* if there exists an algorithm $\mathcal{A}$ that produces a summary $S(D_1 \uplus D_2, \varepsilon)$[1] from any two input summaries $S(D_1, \varepsilon)$ and $S(D_2, \varepsilon)$. Note that, by definition, the size of the merged summary produced by $\mathcal{A}$ is at most $k(|D_1| + |D_2|, \varepsilon)$. If $k(n, \varepsilon)$ is independent of n, which we can denote by $k(\varepsilon)$, then the size of each of $S(D_1, \varepsilon), S(D_2, \varepsilon)$, and the summary produced by $\mathcal{A}$ is at most $k(\varepsilon)$. The merge algorithm $\mathcal{A}$ may represent a summary $S(D, \varepsilon)$ in a certain way or may store some ad-

[1] $\uplus$ denotes multiset addition.

ditional information (e.g., a data structure to expedite the merge procedure). With a slight abuse of notation, we will also use $S(D, \varepsilon)$ to denote this representation of the summary and to include the additional information maintained. We will develop both randomized and deterministic merging algorithms. For randomized algorithms, we require that the produced summary is valid with constant probability after any number of merging operations. The success probability can always be boosted to $1 - \delta$ by building $O(\log(1/\delta))$ independent summaries, and the bound is sometimes better with more careful analysis. But we state our main results below assuming δ is a small constant for simplicity and fair comparison with prior results, while the detailed bounds will be given in the later technical sections.

Note that if we restrict the input so that $|D_2| = 1$, i.e., we always merge a single item at a time, then we recover a streaming model: $S(D, \varepsilon)$ is the summary (and the data structure) maintained by a streaming algorithm, and $\mathcal{A}$ is the algorithm to update the summary with every new arrival. Thus mergeability is a strictly stronger requirement than streaming, and the summary size should be at least as large.

Some summaries are known to be mergeable. For example, all *sketches* that are linear functions of (the frequency vector of) D are trivially mergeable. These include the F_2 AMS sketch [Alon et al. 1999], the Count-Min sketch [Cormode and Muthukrishnan 2005], the ℓ_1 sketch [Feigenbaum et al. 2003; Nelson and Woodruff 2010], among many others. Summaries that maintain the maximum or top-k values can also be easily merged, most notably summaries for estimating the number of distinct elements [Bar-Yossef et al. 2002; Kane et al. 2010]. However, several fundamental problems have summaries that are based on other techniques, and are not known to be mergeable (or have unsatisfactory bounds). Designing mergeable summaries for these problems will be the focus of this paper.

Finally, we note that our algorithms operate in a comparison model, in which only comparisons are used on elements in the data sets. In this model we assume each element, as well as any integer no more than n, can be stored in one unit of storage. Some prior work on building summaries has more strongly assumed that elements are drawn from a bounded universe $[u] = \{0, \ldots, u - 1\}$ for some $u \geq n$, and one unit of storage has $\log u$ bits. Note that any result in the comparison model also holds in the bounded-universe model, but not vice-versa.

1.2. Previous results

In this subsection we briefly review the previous results on specific summaries that we study in this paper.

Frequency estimation and heavy hitters. For a multiset D, let $f(x)$ be the frequency of x in D. A ε-approximate frequency estimation summary of D can be used to estimate $f(x)$ for any x within an additive error of εn. A heavy hitters summary allows one to extract all frequent items approximately, i.e., for a user-specified ϕ, it returns all items x with $f(x) > \phi n$, no items with $f(x) < (\phi - \varepsilon)n$, while an item x with $(\phi - \varepsilon)n \leq f(x) \leq \phi n$ may or may not be returned.

In the bounded-universe model, the frequency estimation problem can be solved by the Count-Min sketch [Cormode and Muthukrishnan 2005] of size $O((1/\varepsilon) \log u)$, which is a linear sketch, and is thus trivially mergeable. Since the Count-Min sketch only allows querying for specific frequencies, in order to report all the heavy hitters efficiently, we need a hierarchy of sketches and the space increases to $O((1/\varepsilon) \log u \log(\frac{\log u}{\varepsilon}))$ from the extra sketches with adjusted parameters. The Count-Min sketch is randomized; while there is also a deterministic linear sketch for the problem [Ganguly and Majumder 2007], its size is $O((1/\varepsilon^2) \log^2 u \log n)$. In some cases $\log u$ is large, for example when the elements are strings or user-defined types, so we seek to avoid such factors.

The counter-based summaries, most notably the MG summary [Misra and Gries 1982] and the SpaceSaving summary [Metwally et al. 2006], have been reported [Cormode and Hadjieleftheriou 2008a] to give the best results for both the frequency estimation and the heavy hitters problem (in the streaming model). They are deterministic, simple, and have the optimal size $O(1/\varepsilon)$. They also work in the comparison model. However, only recently were they shown to support a weaker model of mergeability, where the error is bounded provided the merge is always "into" a single summary [Berinde et al. 2010]. Some merging algorithms for these summaries have been previously proposed, but the error increases after each merging step [Manjhi et al. 2005; Manjhi et al. 2005].

Quantile summaries. For the quantile problem we assume that the elements are drawn from a totally ordered universe and D is a set (i.e., no duplicates); this assumption can be removed by using any tie breaking method. For any $0 < \phi < 1$, the *ϕ-quantile* of D is the item x with rank $r(x) = \lfloor \phi n \rfloor$ in D, where the *rank* of x is the number of elements in D smaller than x. An *ε-approximate ϕ-quantile* is an element with rank between $(\phi - \varepsilon)n$ and $(\phi + \varepsilon)n$, and a quantile summary allows us to extract an ε-approximate ϕ-quantile for any $0 < \phi < 1$. It is well known [Cormode and Hadjieleftheriou 2008a] that the frequency estimation problem can be reduced to an ε'-approximate quantile problem for some $\varepsilon' = \Theta(\varepsilon)$, by identifying elements that are quantiles for multiples of ε' after tie breaking. Therefore, a quantile summary is automatically a frequency estimation summary (ignoring a constant-factor difference in ε), but not vice versa.

Quite a number of quantile summaries have been designed [Gilbert et al. 2002; Greenwald and Khanna 2004; 2001; Shrivastava et al. 2004; Manku et al. 1998; Cormode and Muthukrishnan 2005], but all the mergeable ones work only in the bounded-universe model and have dependency on $\log u$. The Count-Min sketch (more generally, any frequency estimation summary) can be organized into a hierarchy to solve the quantile problem, yielding a linear sketch of size $O((1/\varepsilon)\log^2 u \log(\frac{\log n}{\varepsilon}))$ after adjusting parameters [Cormode and Muthukrishnan 2005]. The q-digest [Shrivastava et al. 2004] has size $O((1/\varepsilon)\log u)$; although not a linear sketch, it is still mergeable. Neither approach scales well when $\log u$ is large. The most popular quantile summary technique is the GK summary [Greenwald and Khanna 2001], which guarantees a size of $O((1/\varepsilon)\log(\varepsilon n))$. A merging algorithm has been previously designed, but the error could increase to 2ε when two ε-summaries are merged [Greenwald and Khanna 2004].

ε-approximations. Let $(D, \mathcal{R})$ be a *range space*, where D is a finite set of objects and $\mathcal{R} \subseteq 2^D$ is a set of *ranges*. In geometric settings, D is typically a set of points in $\mathbb{R}^d$ and the ranges are induced by a set of geometric regions, e.g., points of D lying inside axis-aligned rectangles, half-spaces, or balls. A subset $S \subseteq D$ is called an *ε-approximation* of $(D, \mathcal{R})$ if

$$\max_{R \in \mathcal{R}} \mathsf{abs}\left(\frac{|R \cap D|}{|D|} - \frac{|R \cap S|}{|S|}\right) \leq \varepsilon,$$

where $\mathsf{abs}(x)$ denotes the absolute value of x. Over the last two decades, ε-approximations have been used to answer several types of queries, including range queries, on multidimensional data.

For a range space $(D, \mathcal{R})$ of VC-dimension[2] ν, Vapnik and Chervonenkis [1971] showed that a random sample of $O((\nu/\varepsilon^2)\log(1/\varepsilon\delta)))$ points from D is an ε-approximation with probability at least $1 - \delta$; the bound was later im-

[2]The VC-dimension of (X, R) is the size of the largest subset $N \subset D$ such that $\{N \cap R \mid R \in \mathcal{R}\} = 2^N$.

proved to $O((1/\varepsilon^2)(\nu + \log(1/\delta)))$ [Li et al. 2001]. Random samples are easily mergeable, but they are far from optimal. It is known that, if $\mathcal{R}$ is the set of ranges induced by d-dimensional axis-aligned rectangles, there is an ε-approximation of size $O((1/\varepsilon)\log^{d+1/2}(1/\varepsilon))$ [Larsen 2011], and an ε-approximation of size $O((1/\varepsilon)\log^{2d}(1/\varepsilon))$ [Phillips 2008] can be computed efficiently. More generally, an ε-approximation of size $O(1/\varepsilon^{2\nu/(\nu+1)})$ exists for a range space of VC-dimension ν [Matoušek 2010]. Furthermore, such an ε-approximation can be constructed using the algorithm by Bansal [2010]; see also [Bansal 2012; Lovett and Meka 2012].

These algorithms for constructing ε-approximations are not known to be mergeable. Although they proceed by partitioning D into small subsets, constructing ε-approximations of each subset, and then repeatedly combining pairs and reducing them to maintain a fixed size, the error accumulates during each reduction step of the process. In particular, the reduction step is handled by a low-discrepancy coloring, and an intense line of work (see books of Matoušek [2010] and Chazelle [2000]) has gone into bounding the discrepancy, which governs the increase in error at each step. We are unaware of any mergeable ε-approximations of $o(1/\varepsilon^2)$ size.

ε-kernels. Finally, we consider ε-kernels [Agarwal et al. 2004] which are summaries for approximating the convex shape of a point set P. Specifically, they are a particular type of coreset that approximates the width of P within a relative $(1+\varepsilon)$-factor in any direction. These summaries have been extensively studied in computational geometry [Agarwal et al. 2010; Chan 2006; 2009; Yu et al. 2004] as they can be used to approximate many other geometric properties of a point set having to do with its convex shape, including diameter, minimum enclosing annulus, and minimum enclosing cylinder. In the static setting in $\mathbb{R}^d$, ε-kernels of size $O(1/\varepsilon^{(d-1)/2})$ [Chan 2006; Yu et al. 2004] can always be constructed, which is optimal. In the streaming setting, several algorithms have been developed [Agarwal et al. 2004; Chan 2006; Agarwal and Yu 2007] ultimately yielding an algorithm using $O((1/\varepsilon^{(d-1)/2})\log(1/\varepsilon))$ space [Zarrabi-Zadeh 2011].

However, such ε-kernels, including those maintained by streaming algorithms, are not known to be mergeable. Combining two ε-kernels will in general double the error or double the size.

Table I: Best constructive summary size upper bounds under different models; the generality of model increases from left to right.

problem	offline	streaming	mergeable
heavy hitters	$1/\varepsilon$	$1/\varepsilon$ [Misra and Gries 1982] [Metwally et al. 2006]	$1/\varepsilon$ (§2)
quantiles (deterministic)	$1/\varepsilon$	$(1/\varepsilon)\log(\varepsilon n)$ [Greenwald and Khanna 2001]	$(1/\varepsilon)\log u$ [Shrivastava et al. 2004] $(1/\varepsilon)\log(\varepsilon n)$ (§3.1, restricted)
quantiles (randomized)	$1/\varepsilon$	$(1/\varepsilon)\log^{3/2}(1/\varepsilon)$ (§3.3)	
ε-approximations (rectangles)	$(1/\varepsilon)\log^{2d}(1/\varepsilon)$	$(1/\varepsilon)\log^{2d+1}(1/\varepsilon)$ [Suri et al. 2006]	$(1/\varepsilon)\log^{2d+3/2}(1/\varepsilon)$ (§4)
ε-approximations (VC-dim ν)	$1/\varepsilon^{\frac{2\nu}{\nu+1}}$	$1/\varepsilon^{\frac{2\nu}{\nu+1}}\log^{\frac{2\nu+1}{\nu+1}}(1/\varepsilon)$ (§4)	
ε-kernels	$1/\varepsilon^{\frac{d-1}{2}}$	$1/\varepsilon^{\frac{d-1}{2}}\log(1/\varepsilon)$ [Zarrabi-Zadeh 2011]	$1/\varepsilon^{\frac{d-1}{2}}$ (§5, w/assumptions)

1.3. Our results

In this paper we provide the best known mergeability results for the problems defined above.

— We first show that the (deterministic) MG and SpaceSaving summaries are mergeable (Section 2): we present a merging algorithm that preserves the size $O(1/\varepsilon)$ and the error parameter ε. Along the way we make the surprising observation that the two summaries are isomorphic, namely, an MG summary can be mapped to a SpaceSaving summary and vice versa.
— In Section 3 we first show a limited result, that the (deterministic) GK summary for ε-approximate quantiles satisfies a weaker mergeability property with no increase in size. Then using different techniques, we achieve our main result of a randomized quantile summary of size $O((1/\varepsilon)\log^{3/2}(1/\varepsilon))$ that is mergeable. This in fact even improves on the previous best randomized streaming algorithm for quantiles, which had size $O((1/\varepsilon)\log^{3}(1/\varepsilon))$ [Suri et al. 2006].
— In Section 4 we present mergeable ε-approximations of range spaces of near-optimal size. This generalizes quantile summaries (for intervals) to more general range spaces. Specifically, for d-dimensional axis-aligned rectangles, our mergeable ε-approximation has size $O((1/\varepsilon)\log^{2d+3/2}(1/\varepsilon))$; for range spaces of VC-dimension ν (e.g., ranges induced by halfspaces in $\mathbb{R}^\nu$), the size is $O((1/\varepsilon^{2\nu/(\nu+1)})\log^{(2\nu+1)/(\nu+1)}(1/\varepsilon))$. The latter bound again improves upon the previous best streaming algorithm which had size $O((1/\varepsilon^{2\nu/(\nu+1)})\log^{\nu+1}(1/\varepsilon))$ [Suri et al. 2006].
— In Section 5 we provide a mergeable ε-kernel for a restricted, but reasonable variant. We assume that we are given a constant factor approximation of the width in every direction ahead of time. This allows us to specify a fixed reference frame, and we can maintain a mergeable ε-kernel of optimal size $O(1/\varepsilon^{(d-1)/2})$ with respect to this fixed reference frame. We leave the unrestricted case as an open question.

Table I gives the current best summary sizes for these problems under various models. The running times of our merging algorithms are polynomial (in many cases near-linear) in the summary size.

In addition to the above theoretical results, we find that our merging algorithm for the MG summary (and hence the SpaceSaving summary) and one version of the mergeable quantile are very simple to implement (in fact, they are even simpler than the previous non-mergeable algorithms). And due to the mergeability property, they can be used in any merging tree without any prior knowledge of the tree's structure, which makes them particularly appealing in practice. In Section 6, we conduct some experiments on a simulated sensor network, and find that, despite their simplicity and lack of knowledge of the merging structure, our algorithms actually perform as well as, sometimes even better than, the previous best known algorithms which need to know the size or the height of the merging tree in advance. This shows that mergeability is not only a mathematically elegant notion, but may also be achieved with simple algorithms that display good performance in practice.

1.4. Conference version

This is an extended version of a paper [Agarwal et al. 2012] appearing in PODS 2012. Unlike this version, the conference version did not contain any of the experimental evaluation, and hence did not demonstrate the simplicity and utility of these summaries and framework. This extended version also contains a new variant of the mergeable heavy hitters algorithm; this version is more aggressive in minimizing the space required for guaranteeing error of at most ε in all frequency estimates. Empiri-

cally it demonstrates to have the best space of any algorithms making this guarantee. Finally, this version has several sometimes subtle tightening of bounds. These include more specific analysis on the probability of failure in the randomized algorithms for quantiles, and also improved analysis of the summary size for ε-approximation for range spaces with better VC-dimension. This second improvement notably results in these summaries having the best bound of any streaming summary, improving upon the previous best bound of Suri et al. [2006].

2. HEAVY HITTERS

The MG summary [Misra and Gries 1982] and the SpaceSaving summary [Metwally et al. 2006] are two popular counter-based summaries for the frequency estimation and the heavy hitters problem. We first recall how they work on a stream of items. For a parameter k, an MG summary maintains up to k items with their associated counters. There are three cases when processing an item x in the stream: (1) If x is already maintained in the summary, its counter is increased by 1. (2) If x is not maintained and the summary currently maintains fewer than k items, we add x into the summary with its counter set to 1. (3) If the summary maintains k items and x is not one of them, we decrement all counters by 1 and remove all items with counters being 0. The SpaceSaving summary is the same as the MG summary except for case (3). In SpaceSaving, if the summary is full and the new item x is not currently maintained, we find any item y with the minimum counter value, replace y with x, and increase the counter by 1. Previous analysis shows that the MG and the SpaceSaving summaries estimate the frequency of any item x with error at most $n/(k+1)$ and n/k, respectively, where n is the number of items processed. By setting $k+1 = 1/\varepsilon$ (MG) or $k = 1/\varepsilon$ (SpaceSaving), they solve the frequency estimation problem with additive error εn with space $O(k) = O(1/\varepsilon)$, which is optimal. They can also be used to report the heavy hitters in $O(1/\varepsilon)$ time by going through all counters; any item not maintained cannot have frequency higher than εn.

We show that both MG and SpaceSaving summaries are mergeable. We first prove the mergeability of MG summaries by presenting two merging algorithms that preserve the size and error. Then we show that SpaceSaving and MG summaries are fundamentally the same, which immediately leads to the mergeability of the SpaceSaving summary.

We start our proof by observing that the MG summary provides a stronger error bound. Let $f(x)$ be the true frequency of item x and let $\hat{f}(x)$ be the counter of x in MG (set $\hat{f}(x) = 0$ if x is not maintained).

LEMMA 2.1. *For any item x, $\hat{f}(x) \leq f(x) \leq \hat{f}(x) + (n - \hat{n})/(k+1)$, where $\hat{n}$ is the sum of all counters in MG.*

PROOF. It is clear that $\hat{f}(x) \leq f(x)$. To see that $\hat{f}(x)$ underestimates $f(x)$ by at most $(n - \hat{n})/(k+1)$, observe that every time the counter for a particular item x is decremented, we decrement all k counters by 1 and ignore the new item. All these $k+1$ items are different. This corresponds to deleting $k+1$ items from the stream, and exactly $(n-\hat{n})/(k+1)$ such operations must have been done when the sum of counters is $\hat{n}$. □

This is related to the result that the MG error is at most $F_1^{res(k)}/k$, where $F_1^{res(k)}$ is the sum of the counts of all items except the k largest [Berinde et al. 2010]. Since each counter stored by the algorithm corresponds to (a subset of) actual arrivals of the corresponding item, we have that $\hat{n} \leq n - F_1^{res(k)}$. However, we need the form of the error bound to be as in the lemma above in order to show mergeability.

Given two MG summaries with the property stated in Lemma 2.1, we present two merging algorithms that produce a merged summary with the same property. More precisely, let S_1 and S_2 be two MG summaries on data sets of sizes n_1 and n_2, respectively. Let $\hat{n}_1$ (resp. $\hat{n}_2$) be the sum of all counters in S_1 (resp. S_2). We know that S_1 (resp. S_2) has error at most $(n_1 - \hat{n}_1)/(k+1)$ (resp. $(n_2 - \hat{n}_2)/(k+1)$).

Merging algorithm that minimizes the actual error. Our first merging algorithm is very simple. We first combine the two summaries by adding up the corresponding counters. This could result in up to $2k$ counters. We then perform a prune operation: Take the $(k+1)$-th largest counter, denoted C_{k+1}, and subtract it from all counters, and then remove all non-positive ones. Clearly this is an efficient procedure: it can be completed with a constant number of sorts and scans of summaries of size $O(k)$. This merging algorithm always uses k counters (provided that there are at least k unique elements in the data set) while maintaining the guarantee in Lemma 2.1 (shown below). In practice this can potentially result in even smaller than ε actual errors. We use MERGEABLEMINERROR to denote this merging algorithm.

Merging algorithm that minimizes the summary size. Our second algorithm tries to minimize the size of the merged summary by eliminating as many counters as possible while maintaining the guarantee of Lemma 2.1. More precisely, we first combine the two summaries by adding up the corresponding counters. Let $C_1, C_2, \ldots, C_s$ denote the counters sorted in descending order and let C_{j+1} be the largest counter that satisfies

$$(k-j)C_{j+1} \leq \sum_{i=j+2}^{s} C_i. \tag{1}$$

We will then subtract C_{j+1} from all counters, and then remove all non-positive ones. We use MERGEABLEMINSPACE to denote this merging algorithm. It is easy to see that $j = k$ always satisfies inequality (1), so the summary produced by MERGEABLEMINSPACE is no larger than the summary produced by MERGEABLEMINERROR. It also upholds the guarantee of Lemma 2.1 (shown below), although its actual error could be larger than that of MERGEABLEMINERROR.

THEOREM 2.2. *MG summaries are mergeable with either of the above merging algorithms. They have size $O(1/\varepsilon)$.*

PROOF. Set $k+1 = \lceil 1/\varepsilon \rceil$. It is clear that in either case the size of the merged summary is at most k, so it only remains to show that the merged summary still has the property of Lemma 2.1, i.e., it has error at most $(n_1 + n_2 - \hat{n}_{12})/(k+1)$ where $\hat{n}_{12}$ is the sum of counters in the merged summary. Then the error will be $(n - \hat{n})/(k+1) \leq \varepsilon n$.

The combine step clearly does not introduce additional error, so the error after the combine step is the sum of the errors from S_1 and S_2, that is, at most $(n_1 - \hat{n}_1 + n_2 - \hat{n}_2)/(k+1)$.

For MERGEABLEMINERROR, the prune operation incurs an additional error of C_{k+1}. If we can show that

$$C_{k+1} \leq (\hat{n}_1 + \hat{n}_2 - \hat{n}_{12})/(k+1), \tag{2}$$

we will arrive at the desired error in the merged summary. If after the combine step, there are no more than k counters, $C_{k+1} = 0$. Otherwise, the prune operation reduces the sum of counters by at least $(k+1)C_{k+1}$: the $k+1$ counters greater than or equal to C_{k+1} get reduced by C_{k+1} and they remain non-negative. So we have $\hat{n}_{12} \leq \hat{n}_1 + \hat{n}_2 - (k+1)C_{k+1}$ and the inequality (2) follows.

For MERGEABLEMINSPACE, we similarly need to show the following inequality:

$$C_{j+1} \leq (\hat{n}_1 + \hat{n}_2 - \hat{n}_{12})/(k+1). \quad (3)$$

Note that after the prune operation, the summation of all counters is

$$\hat{n}_{12} = \sum_{i=1}^{j}(C_i - C_{j+1}) = (\sum_{i=1}^{j} C_i) - jC_{j+1}.$$

We have

$$\begin{aligned}\hat{n}_1 + \hat{n}_2 - \hat{n}_{12} &= \sum_{i=1}^{s} C_i - ((\sum_{i=1}^{j} C_i) - jC_{j+1}) \\ &= (\sum_{i=j+1}^{s} C_i) + jC_{j+1} \\ &= (\sum_{i=j+2}^{s} C_i) + (j+1)C_{j+1}.\end{aligned}$$

Thus, rearranging (3) we obtain

$$(k+1)C_{j+1} \leq (\sum_{i=j+2}^{s} C_i) + (j+1)C_{j+1},$$

which is equivalent to the condition (1). □

The time to perform the merge under the MERGEABLEMINERROR approach is $O(k)$: we can merge the counter sets in time linear in their size (assuming a suitable dictionary data structure such as a hash table to hold the items and weights). Then we can extract the kth largest counter in time $O(k)$ via a standard selection algorithm, and adjust weights or prune items with another pass. For the MERGEABLEMINSPACE approach, we can merge and sort the items by weight, and then pass through to find the counter satisfying (1).

The isomorphism between MG and SpaceSaving. Next we show that MG and SpaceSaving are isomorphic. Specifically, consider an MG summary with k counters and a SpaceSaving summary of $k+1$ counters, processing the same stream. Let min^{SS} be the minimum counter of the SpaceSaving summary (set $min^{SS} = 0$ when the summary is not full), and $\hat{n}^{MG}$ be the sum of all counters in the MG summary. Let $\hat{f}^{MG}(x)$ (resp. $\hat{f}^{SS}(x)$) be the counter of item x in the MG (resp. SpaceSaving) summary, and set $\hat{f}^{MG}(x) = 0$ (resp. $\hat{f}^{SS}(x) = min^{SS}$) if x is not maintained.

LEMMA 2.3. *After processing n items, $\hat{f}^{SS}(x) - \hat{f}^{MG}(x) = min^{SS} = (n - \hat{n}^{MG})/(k+1)$ for all x.*

PROOF. We prove $\hat{f}^{SS}(x) - \hat{f}^{MG}(x) = min^{SS}$ for all x by induction on n. For the base case $n = 1$, both summaries store the first item with counter 1, and we have $min^{SS} = 0$ and the claim trivially holds. Now suppose the claim holds after processing n items. We analyze the MG summary case by case when inserting the $(n+1)$-th item, and see how SpaceSaving behaves correspondingly. Suppose the $(n+1)$-th item is y.

(1) y is currently maintained in MG with counter $\hat{f}^{MG}(y) > 0$. In this case MG will increase $\hat{f}^{MG}(y)$ by 1. By the induction hypothesis we have $\hat{f}^{SS}(y) = \hat{f}^{MG}(y) +$

$min^{SS} > min^{SS}$ so y must be maintained by SpaceSaving, too. Thus SpaceSaving will also increase $\hat{f}^{SS}(y)$ by 1. Meanwhile min^{SS} remains the same and so do all $\hat{f}^{SS}(x), \hat{f}^{MG}(x)$ for $x \neq y$, so the claim follows.

(2) y is not maintained by the MG summary, but it is not full, so it will create a new counter set to 1 for y. By the induction hypothesis $\hat{f}^{SS}(y) = min^{SS}$, which means that y either is not present in SpaceSaving or has the minimum counter. We also note that $\hat{f}^{SS}(y)$ cannot be a unique minimum counter in SpaceSaving with $k+1$ counters; otherwise by the induction hypothesis there would be k items x with $\hat{f}^{MG}(x) > 0$ and the MG summary with k counters would be full. Thus, min^{SS} remains the same and $\hat{f}^{SS}(y)$ will become $min^{SS}+1$. All other $\hat{f}^{SS}(x), \hat{f}^{MG}(x), x \neq y$ remain the same so the claim still holds.

(3) y is not maintained by the MG summary and it is full. MG will then decrease all current counters by 1 and remove all zero counters. By the induction hypothesis $\hat{f}^{SS}(y) = min^{SS}$, which means that y either is not present in SpaceSaving or has the minimum counter. We also note that in this case there is a unique minimum counter (which is equal to $\hat{f}^{SS}(y)$), because the induction hypothesis ensures that there are k items x with $\hat{f}^{SS}(x) = \hat{f}^{MG}(x) + min^{SS} > min^{SS}$. SpaceSaving will then increase $\hat{f}^{SS}(y)$, as well as min^{SS}, by 1. It can then be verified that we still have $\hat{f}^{SS}(x) - \hat{f}^{MG}(x) = min^{SS}$ for all x after inserting y.

To see that we always have $min^{SS} = (n - \hat{n}^{MG})/(k+1)$, just recall that the sum of all counters in the SpaceSaving summary is always n. If we decrease all its $k+1$ counters by min^{SS}, it becomes MG, so $min^{SS}(k+1) = n - \hat{n}^{MG}$ and the lemma follows. □

Therefore, the two algorithms are essentially the same. The difference is that MG gives lower bounds on the true frequencies while SpaceSaving gives upper bounds, while the gap between the upper and lower bound is $min^{SS} = \varepsilon(n - \hat{n}^{MG})$ for any element in the summary.

Due to this correspondence, we can immediately state:

COROLLARY 2.4. *The SpaceSaving summaries are mergeable.*

That is, we can perform a merge of SpaceSaving summaries by converting them MG summaries (by subtracting min^{SS} from each counter), merging these, and converting back (by adding $(n - \hat{n}^{MG})/(k+1)$ to each counter). It is also possible to more directly implement merging. For example, to merge under the MERGEABLEMINERROR process, we simply merge the counter sets as above, and find C_k, the k'th largest weight. We then drop all counters that have value C_k or less.

3. QUANTILES

We first describe a result of a weaker form of mergeability for a deterministic summary, the GK algorithm [Greenwald and Khanna 2001]. We say a summary is "one-way" mergeable if the summary meets the criteria of mergeability under the restriction that one of the inputs to a merge is not itself the output of a prior merge operation. One-way mergeability is essentially a "batched streaming" model where there is a main summary S_1, into which we every time insert a batch of elements, summarized by a summary S_2. As noted in Section 1.2, prior work [Berinde et al. 2010] showed similar one-way mergeability of heavy hitter algorithms.

After this, the bulk of our work in this section is to show a randomized construction which achieves (full) mergeability by analyzing quantiles through the lens of ε-approximations of the range space of intervals. Let D be a set of n points in one dimension. Let $\mathcal{I}$ be the set of all half-closed intervals $I = (-\infty, x]$. Recall that an

ε-approximation S of D (w.r.t. $\mathcal{I}$) is a subset of points of D such that for any $I \in \mathcal{I}$, $n|S \cap I|/|S|$ estimates $|D \cap I|$ with error at most εn. In some cases we may use a weighted version, i.e., each point p in S is associated with a weight $w(p)$. A point p with weight $w(p)$ represents $w(p)$ points in D, and we require that the weighted sum $\sum_{p \in S \cap I} w(p)$ estimates $|D \cap I|$ with error at most εn. Since $|D \cap I|$ is the rank of x in D, we can then do a binary search[3] to find an ε-approximate ϕ-quantile for any given ϕ. We will first develop a randomized mergeable ε-approximation of size $O((1/\varepsilon)\log(\varepsilon n)\sqrt{\log(1/\varepsilon)})$ inspired by low-discrepancy halving. Then after we review some classical results about random sampling, we combine the random-sample-based and low-discrepancy-based algorithms to produce a hybrid mergeable ε-approximation whose size is independent of n.

3.1. One-way mergeability

We define a restricted form of mergeability where the merging is always "one-way".

Definition 3.1 (*One-way mergeability*). A summary $S(D, \varepsilon)$ is *one-way mergeable* if there exist two algorithms $\mathcal{A}_1$ and $\mathcal{A}_2$ such that, (1) given any D, $\mathcal{A}_2$ creates a summary of D, as $S(D, \varepsilon)$; (2) given any $S(D_2, \varepsilon)$ produced by $\mathcal{A}_2$ and any $S(D_1, \varepsilon)$ produced by $\mathcal{A}_1$ or $\mathcal{A}_2$, $\mathcal{A}_1$ builds a merged summary $S(D_1 \uplus D_2, \varepsilon)$.

Note that one-way mergeability degenerates to the standard streaming model when we further restrict to $|D_2| = 1$ and assume without loss of generality that $S(D_2, \varepsilon) = D_2$ in this case. One-way mergeability is essentially a "batched streaming" model where there is a main summary, into which we insert batches of elements at a time, summarized by a summary in S_2. As noted in Section 1.2, prior work showed one-way mergeability of heavy hitter algorithms.

THEOREM 3.2. *Any quantile summary algorithm which is incrementally maintainable is one-way mergeable.*

PROOF. Given a quantile summary S, it promises to approximate the rank of any element by εn. Equivalently, since D defines an empirical frequency distribution f (where, as in the previous section, $f(x)$ gives the count of item x) we can think of S as defining an approximate cumulative frequency function $\hat{F}$, that is, $\hat{F}(i)$ gives the (approximate) number of items in the input which are dominated by i. The approximation guarantees mean that $\|F - \hat{F}\|_\infty \leq \varepsilon n$, where F is the (true) cumulative frequency function (CFF) of f, and the ∞-norm, $\|\cdot\|_\infty$, takes the maximal value. Further, from $\hat{F}$ and n, we can derive $\hat{f}$, the distribution whose cumulative frequency function is $\hat{F}$[4].

Given summaries S_1 and S_2, which summarize n_1 and n_2 items respectively with error ε_1 and ε_2, we can perform a one-way merge of S_2 into S_1 by extracting the distribution $\hat{f}_2$, and interpreting this as n_2 updates to S_2. The resulting summary is a summary of $f' = f_1 + \hat{f}_2$, that is, $f'(x) = f_1(x) + \hat{f}_2(x)$. This summary implies a cumulative frequency function $\hat{F}'$, whose error relative to the original data is

$$\begin{aligned}\|\hat{F}' - (F_1 + F_2)\|_\infty &\leq \|\hat{F}' - (\hat{F}_2 + F_1)\|_\infty + \|(\hat{F}_2 + F_1) - (F_1 + F_2)\|_\infty \\ &\leq \varepsilon_1(n_1 + n_2) + \|\hat{F}_2 - F_2\|_\infty\end{aligned}$$

[3]We will need all $O(\log(1/\varepsilon))$ comparisons in the binary search to succeed, so there is actually an $O(\log\log(1/\varepsilon))$ difference between the two problems, which we omit to keep the expressions simple.

[4]We assume that this is straightforward to extract, as is the case with existing quantile summaries. If not, we can use the summary as a black box, and extract the ε-quantiles of the distribution, from which it is straightforward to construct a distribution f which has these quantiles.

$$= \varepsilon_1(n_1 + n_2) + \varepsilon_2 n_2.$$

By the same argument, if we merge in a third summary S_3 of n_3 items with error ε_3, the resulting error is at most $\varepsilon_1(n_1+n_2+n_3)+\varepsilon_2 n_2+\varepsilon_3 n_3$. So if this (one-way) merging is done over a large number of summaries $S_1, S_2, S_3 \ldots S_s$, then the resulting summary has error at most

$$\varepsilon_1(\sum_{i=1}^{s} n_i) + \sum_{i=2}^{s} \varepsilon_i n_i \leq (\varepsilon_1 + \max_{1<i\leq s} \varepsilon_i)N.$$

Setting $\varepsilon_1 = \varepsilon_2 = \ldots \varepsilon_i = \varepsilon/2$ is sufficient to meet the requirements on this error. □

An immediate observation is that the GK algorithm [Greenwald and Khanna 2001] (along with other deterministic techniques for streaming computation of quantiles which require more space [Manku et al. 1998]) meets these requirements, and is therefore one-way mergeable. The merging is fast, since it takes time linear in the summary size to extract an approximate distribution, and near-linear to insert into a second summary.

COROLLARY 3.3. *The GK algorithm is one-way mergeable, with a summary size of* $O((1/\varepsilon)\log(\varepsilon n))$.

This analysis implies a step towards full-mergeability. We can apply the rule of always merging the summary of the smaller data set into the larger. This ensures that in the summarization of n items, any item participates in at most $\log(\varepsilon n)$ one-way merges (we only incur errors for data sets of at least $1/\varepsilon$ points). Thus the total error is $\varepsilon \log(\varepsilon n)$, and the summary has size $O((1/\varepsilon)\log(\varepsilon n))$. If we know n in advance, we can rescale ε by a $\log(\varepsilon n)$ factor, and achieve a space of $O((1/\varepsilon)\log^2(\varepsilon n))$, matching the result of [Greenwald and Khanna 2004]. However, this does not achieve full mergeability, which does not allow foreknowledge of n.

3.2. Low-discrepancy-based summaries

Unfortunately, we cannot show that the GK summary is (fully) mergeable, nor can we give a negative proof. We conjecture it is not, and in fact we conjecture that any deterministic mergeable quantile summary must have size linear in n in the comparison model. On the hand, in this section we give a randomized mergeable quantile summary of size $O((1/\varepsilon)\log^{1.5}(1/\varepsilon))$. The idea is to adopt the merge-reduce algorithm [Matoušek 1991; Chazelle and Matousek 1996] for constructing deterministic ε-approximations of range spaces, but randomize it in a way so that error is preserved.

Same-weight merges. We first consider a restricted merging model where each merge is applied only to two summaries (ε-approximations) representing data sets of the same size. Let S_1 and S_2 be the two summaries to be merged. The algorithm is very simple: Set $S' = S_1 \cup S_2$, and sort S'. Then let S_e be all even points in the sorted order and S_o be all odd points in the sorted order. We retain either S_e or S_o with equal probability as our merged summary S. We call this a *same-weight merge*. We note essentially the same algorithm was used by Suri et al. [2006], but their analysis shows that the error increases gradually after a series of merges. Below we give our analysis which shows that the error is actually preserved. We first consider a single merge.

LEMMA 3.4. *For any interval* $I \in \mathcal{I}$, $2|I \cap S|$ *is an unbiased estimator of* $|I \cap S'|$ *with error at most* 1.

PROOF. If $|I \cap S'|$ is even, then $I \cap S'$ contains the same number of even and odd points. Thus $2|I \cap S| = |I \cap S'|$ no matter whether we choose the even or odd points.

If $|I \cap S'|$ is odd, it must contain exactly one more odd point than even points. Thus if we choose the odd points, we overestimate $|I \cap S'|$ by 1; if we choose the even points, we underestimate by 1. Either happens with probability $1/2$. $\square$

Below we generalize the above lemma to multiple merges, but each merge is a *same-weight* merge. We set the summary size to be k_ε, and note that each merge operation takes time $O(k_\varepsilon)$ to merge the sorted lists and pick every other point. Let D be the entire data set of size n. We assume that n/k_ε is a power of 2 (this assumption will be removed later). Thus, the whole merging process corresponds to a complete binary tree with $m = \log(n/k_\varepsilon)$ levels. Each internal node in the tree corresponds to the (same-weight) merge of its children. Let S be the final merged summary, corresponding to the root of the tree. Note that each point in S represents 2^m points in D. Recall that (randomized) mergeability requires that S is a valid ε-summary after any number of merges, so it important that the merging algorithm is oblivious to m (hence n). In fact, our algorithm only has one parameter k_ε. We first analyze the correctness of S for any *one* query.

LEMMA 3.5. *If we set $k_\varepsilon = O((1/\varepsilon)\sqrt{\log(1/\delta)})$, then for any interval $I \in \mathcal{I}$ with probability at least $1 - \delta$,*

$$\mathsf{abs}(|I \cap D| - 2^m |I \cap S|) \leq \varepsilon n.$$

PROOF. Fix any I. We prove this lemma by considering the over-count error $X_{i,j}$ (which could be positive or negative) produced by a single merge of two sets S_1 and S_2 to get a set $S^{(j)}$ in level i. Then we consider the error $M_i = \sum_{j=1}^{r_i} X_{i,j}$ of all $r_i = 2^{m-i}$ merges in level i, and sum them over all m levels using a single Chernoff-Hoeffding bound. We will show that the errors for all levels form a geometric series that sums to at most εn with probability at least $1 - \delta$.

Start the induction at level 1, before any sets are merged. Merging two sets S_1 and S_2 into $S^{(j)}$ causes the estimate $2|S^{(j)} \cap I|$ to have over-count error

$$X_{1,j} = 2|S^{(j)} \cap I| - |(S_1 \cup S_2) \cap I|.$$

Now $\mathsf{abs}(X_{1,j}) \leq 1 = \Delta_1$, by Lemma 3.4. There are $r_1 = 2^{m-1}$ such merges in this level, and since each choice of even/odd is made independently, this produces r_1 independent random variables $\{X_{1,1}, \ldots, X_{1,r_1}\}$. Let their total over-count error be denoted $M_1 = \sum_{j=1}^{r_1} X_{1,j}$. So, now except for error M_1, the set of r_1 sets $S^{(j)}$, each the result of an independent merge of two sets, can be used to represent $|D \cap I|$ by $2|(\bigcup_j S^{(j)}) \cap I|$.

Then inductively, up to level i, we have accumulated at most $\sum_{s=1}^{i-1} M_s$ error, and have $2r_i$ point sets of size k_ε, where $r_i = 2^{m-i}$. We can again consider the merging of two sets S_1 and S_2 into $S^{(j)}$ by a same-weight merge. This causes the estimate $2^i|S^{(j)} \cap I|$ to have error

$$X_{i,j} = 2^i|S^{(j)} \cap I| - 2^{i-1}|(S_1 \cup S_2) \cap I|,$$

where $\mathsf{abs}(X_{i,j}) \leq 2^{i-1} = \Delta_i$, by Lemma 3.4. Again we have r_i such merges in this level, and r_i independent random variables $\{X_{i,1}, \ldots, X_{i,r_i}\}$. The total error in this level is $M_i = \sum_{j=1}^{r_i} X_{i,j}$, and except for this error M_i and $M_{i-1}, \ldots, M_1$, we can accurately estimate $|D \cap I|$ as $2^i|(\bigcup_j S^{(j)}) \cap I|$ using the r_i sets $S^{(j)}$.

We now analyze $M = \sum_{i=1}^{m} M_i$ using the following Chernoff-Hoeffding bound. Given a set $\{Y_1, \ldots, Y_t\}$ of independent random variables such that $\mathsf{abs}(Y_j - E[Y_j]) \leq \Upsilon_j$, then for $T = \sum_{j=1}^{t} Y_j$ we can bound $\Pr[\mathsf{abs}(T - \sum_{j=1}^{t} E[Y_j]) > \alpha] \leq 2e^{-2\alpha^2/(\sum_{j=1}^{t}(2\Upsilon_j)^2)}$. In our case, the random variables are m sets of r_i variables $\{X_{i,j}\}_j$, each with $E[X_{i,j}] = 0$ and

$\mathsf{abs}(X_{i,j} - E[X_{i,j}]) = \mathsf{abs}(X_{i,j}) \leq \Delta_i = 2^{i-1}$. There are m such sets for $i \in \{1, \ldots, m\}$. Setting $\alpha = h2^m$ for some parameter h, we can write

$$\begin{aligned}
\mathbf{Pr}\left[\mathsf{abs}(M) > h2^m\right] &\leq 2\exp\left(-\frac{2\left(h2^m\right)^2}{\sum_{i=1}^{m}\sum_{j=1}^{r_i}(2\Delta_i)^2}\right) \\
&= 2\exp\left(-\frac{2\left(h2^m\right)^2}{\sum_{i=1}^{m}(r_i)(2^{2i})}\right) \\
&= 2\exp\left(-\frac{2h^2\left(2^{2m}\right)}{\sum_{i=1}^{m}(2^{m-i})(2^{2i})}\right) \\
&= 2\exp\left(-\frac{2h^2\left(2^{2m}\right)}{\sum_{i=1}^{m}2^{m+i}}\right) \\
&= 2\exp\left(-\frac{2h^2}{\sum_{i=1}^{m}2^{i-m}}\right) \\
&= 2\exp\left(-\frac{2h^2}{\sum_{i=1}^{m}2^{-i}}\right) \\
&< 2\exp\left(-2h^2\right).
\end{aligned}$$

Thus if we set $h = \sqrt{(1/2)\ln(2/\delta)}$, with probability at least $1-\delta$ we have $\mathsf{abs}(M) < h2^m = hn/k_\varepsilon$. Thus for $k_\varepsilon = O(h/\varepsilon)$ the error will be smaller than εn, as desired. □

An ε-approximation is required to be correct for *all* intervals $I \in \mathcal{I}$, but this can be easily achieved by increasing k_ε appropriately. There is a set of $1/\varepsilon$ evenly spaced intervals $\mathcal{I}_\varepsilon$ such that any interval $I \in \mathcal{I}$ has

$$\mathsf{abs}(|D \cap I| - |D \cap I'|) \leq \varepsilon n/2$$

for some $I' \in \mathcal{I}_\varepsilon$. We can then apply the union bound by setting $\delta' = \delta\varepsilon$ and run the above scheme with $k_\varepsilon = O((1/\varepsilon)\sqrt{\log(1/\delta')})$. Then with probability at least $1-\delta$, no interval in $\mathcal{I}_\varepsilon$ has more than $\varepsilon n/2$ error, which means that no interval in $\mathcal{I}$ has more than εn error.

THEOREM 3.6. *There is a same-weight merging algorithm that maintains a summary of size $O((1/\varepsilon)\sqrt{\log(1/\varepsilon\delta)})$ which is a one-dimensional ε-approximation with probability at least $1-\delta$.*

Uneven-weight merges. We next reduce uneven-weight merges to $O(\log(n/k_\varepsilon))$ weighted instances of the same-weight ones. This follows the so-called *logarithmic technique* used in many similar situations [Greenwald and Khanna 2004].

Set $k_\varepsilon = O((1/\varepsilon)\sqrt{\log(1/\varepsilon\delta)})$ as previously. Let n be the size of data set currently being summarized. We maintain $\log(n/k_\varepsilon)$ layers, each of which summarizes a disjoint subset of data points. Each layer is either empty or maintains a summary with exactly k_ε points. In the 0th layer, each summary point has weight 1, and in the ith layer, each summary point has weight 2^i. We assume n/k_ε is an integer; otherwise we can always store the extra $\leq k_\varepsilon$ points exactly without introducing any error.

We merge two such summaries S_1 and S_2 via same-weight merging, starting from the bottom layer, and promoting retained points to the next layer. At layer i, we may have $0, 1, 2$, or 3 sets of k_ε points each. If there are 0 or 1 such sets, we skip this layer and proceed to layer $i+1$; if there are 2 or 3 such sets we merge any two of them using

a same-weight merge, and promote the merged set of k_ε points to layer $i+1$. Consequently, each merge takes time $O(k_\varepsilon \log \varepsilon n)$, linear in the total size of both summaries.

The analysis of this logarithmic scheme is straightforward because our same-weight merging algorithm preserves the error parameter ε across layers: Since each layer is produced by only same-weight merges, it is an ε-approximation of the set of points represented by this layer, namely the error is εn_i for layer i where n_i is the number of points being represented. Summing over all layers yields a total error of εn. Again it should be clear that this algorithm works without the *a priori* knowledge of the number of merges.

THEOREM 3.7. *There is a mergeable summary of size* $O((1/\varepsilon)\sqrt{\log(1/\varepsilon\delta)}\log(\varepsilon n))$ *which is a one-dimensional* ε*-approximation with probability at least* $1-\delta$.

3.3. Hybrid quantile summaries

In this section, we build on the above ideas to remove the dependence on n in the size of the summary.

Random sampling. A classic result [Vapnik and Chervonenkis 1971; Talagrand 1994] shows that a random sample of $k_\varepsilon = O((1/\varepsilon^2)\log(1/\delta))$ points from D is an ε-approximation with probability $1-\delta$. So an ε-approximation can also be obtained by just retaining a random sample of D. Random samples are easily mergeable: A standard way of doing so is to assign a random value $u_i \in [0,1]$ for each point $p_i \in D$, and we retain in $S \subset D$ the k_ε elements with the smallest u_i values. On a merge of two summaries S_1 and S_2, we retain the set $S \subset S_1 \cup S_2$ that has the k_ε smallest u_i values from the $2k_\varepsilon$ points in $S_1 \cup S_2$. It is also easy to show that finite precision ($O(\log n)$ bits with high probability) is enough to break all ties.

FACT 1. *A random sample of size* $k_\varepsilon = O((1/\varepsilon^2)\log(1/\delta))$ *is mergeable and is an* ε*-approximation with probability at least* $1-\delta$.

Random sampling and mergeability. We now show an intermediate result that achieves $\tilde{O}(1/\varepsilon)$ space usage, independent of n, before the main result in this section, which reduces the dependency on $\log 1/\varepsilon$. The idea is that since a sample of size $O((1/\varepsilon_s^2)\log(1/\delta_s))$ provides an ε_s-approximation, we can reduce the size further by applying the mergeable summary from Theorem 3.7 with parameter ε_h over a sample. The resulting summary provides an $(\varepsilon_s+\varepsilon_h)$-approximation. The challenge is that we do not know the size of the data in advance, so cannot choose the appropriate sampling rate. We address this by making multiple guesses of the sampling rate in parallel. That is, we sample the input data at rates of $p = 1, \frac{1}{2}, \frac{1}{4}, \ldots$, and feed each of these into a different instance of the mergeable summary from Theorem 3.7. We merge two instances of this collection of summaries by merging those summaries with the corresponding p values. To bound the size of this collection, we can track the smallest p value which has sampled sufficiently many ($\Omega((1/\varepsilon_s^2)\log(1/\delta_s))$) data items, and prune the summaries associated with larger p values. A Chernoff bound shows that we maintain summaries for only $O(\log(1/\varepsilon_s))$ different p values at a time. The size of each summary is then $O((1/\varepsilon_h)\sqrt{\log(1/\varepsilon_h\delta_h)}\log((\varepsilon_h/\varepsilon_s^2)\log(1/\delta_s)))$ Setting $\varepsilon_h = \varepsilon_s = \varepsilon/2$, and treating δ_h and δ_s as constants to simplify the expression, we obtain a total summary size of $O((1/\varepsilon)\log^{5/2}(1/\varepsilon))$. However, it seems redundant to track all these guesses of p in parallel. We next remove this factor of $\log(1/\varepsilon)$ by a more involved argument.

Specifically, we show how to combine the approaches of random sampling and the low-discrepancy-based method to achieve a summary size independent of n. At an intuitive level, for a subset of points, we maintain a random sample of size about

$(1/\varepsilon)\log(1/\varepsilon)$. The sample guarantees an error of $\sqrt{\varepsilon}$ for any range, so we make sure that we only use this on a small fraction of the points (at most εn points). The rest of the points are processed using the logarithmic method. That is, we maintain $O(\log(1/\varepsilon))$ levels of the hierarchy, and only in the bottom level use a random sample. This leads to a summary of size $(1/\varepsilon)$ poly $\log(1/\varepsilon)$.

Hybrid structure. We now describe the summary structure in more detail for n points, where $2^{j-1}k_\varepsilon \leq n < 2^j k_\varepsilon$ for some integer j, and $k_\varepsilon = (4/\varepsilon)\sqrt{\ln(4/\varepsilon\delta)}$. Let $g_\varepsilon = (64/\varepsilon^2)\ln(16/\varepsilon\delta)$. For each level l between $i = j - \log_2(g_\varepsilon)$ and $j-1$ we either maintain k_ε points, or no points. Each point at the lth level has weight 2^l. The remaining $m \leq 2^i k_\varepsilon$ points are in a *random buffer* at level i, represented by a random sample of k_ε points (or only m if $m < k_\varepsilon$). Each point in the sample has weight m/k_ε (or 1 if $m < k_\varepsilon$). Note the total size is $O(k_\varepsilon \log(g_\varepsilon)) = O((1/\varepsilon)\log^{1.5}(1/\varepsilon\delta))$.

Merging. Two hybrid summaries S_1 and S_2 are merged as follows. Let n_1 and n_2 be the sizes of the data sets represented by S_1 and S_2, and w.l.o.g. we assume $n_1 \geq n_2$. Let $n = n_1 + n_2$. Let j be an integer such that $2^{j-1}k_\varepsilon \leq n < 2^j k_\varepsilon$, and let $i = j - \log_2(g_\varepsilon)$.

First consider the random buffer in the merged summary; it now contains both random buffers in S_1 and S_2, as well as all points represented at level $i-1$ or below in either S_1 or S_2. Note that if $n_1 \geq 2^{j-1}k_\varepsilon$, then S_1 cannot have points at level $l \leq i-1$. Points from the random buffers of S_1 and S_2 already have u_i values. For every p of weight $w(p) = 2^l$ that was in a level $l \leq i-1$, we insert $w(p)$ copies of p into the buffer and assign a new u_i value to each copy. Then the k_ε points with the largest u_i values are retained.

When the random buffer is full, i.e., represents $2^i k_\varepsilon$ points, then it performs an "output" operation, and outputs the sample of k_ε points of weight 2^i each, which is then merged into the hierarchy at level i. It is difficult to ensure that the random buffer represents exactly $m = 2^i k_\varepsilon$ points when it outputs points, but it is sufficient if this occurs when the buffer has this size in expectation. There are two ways the random buffer may reach this threshold of representing m points:

(1) On insertion of a point from the hierarchy of level $l \leq i-1$. Since copies of these points are inserted one at a time, representing 1 point each, it reaches the threshold exactly. The random buffer outputs and then inserts the remaining points in a new random buffer.
(2) On the merge of two random buffers B_1 and B_2, which represent b_1 and b_2 points, respectively. Let $b_1 \geq b_2$, and let B be the union of the two buffers and represent $b = b_1 + b_2$ points. If $b < m$ we do not output; otherwise we have $m/2 \leq b_1 < m \leq b < 2m$. To ensure the output from the random buffer represents m points in expectation we either:
   (i) With probability $\rho = (b-m)/(b-b_1)$, we do not merge, but just output the sample of B_1 and let B_2 be the new random buffer.
   (ii) With probability $1-\rho = (m-b_1)/(b-b_1)$, output the sample of B after the merge, and let the new random buffer be empty.

   Note that the expected number of points represented by the output from the random buffer is $\rho b_1 + (1-\rho)b = \frac{b-m}{b-b_1}b_1 + \frac{m-b_1}{b-b_1}b = m$.

Next, the levels of the hierarchy of both summaries are merged as before, starting from level i. For each level if there are 2 or 3 sets of k_ε points, two of them are merged using a same-weight merge, and the merged set is promoted to the next level. See Figure 1 for illustration of hybrid structure.

Analysis. First we formalize the upward movement of points.

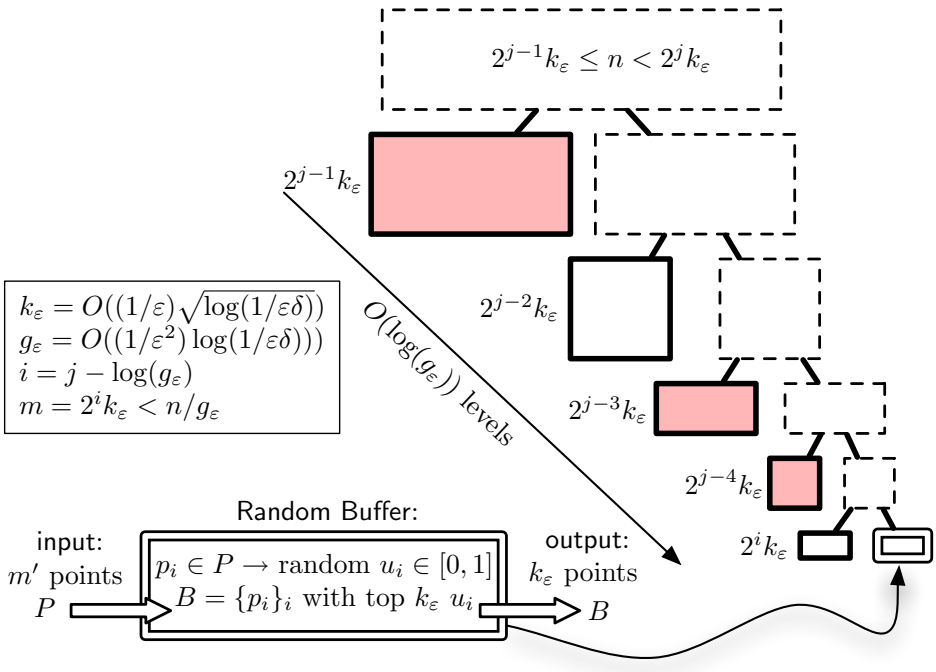

Fig. 1: Illustration of the hybrid summary. The labels at each level of the hierarchy shows the number of points represented at that layer. Each filled box contains exactly k_ε summary points, and empty boxes contain no summary points.

LEMMA 3.8. *Over time, a point only moves up in the hierarchy (or is dropped): it never decreases in level.*

PROOF. For this analysis, the random buffer is considered to reside at level i at the end of every action. There are five cases we need to consider.

(1) A point is involved in a same weight merge at level l. After the merge, it either disappears, or is promoted to level $l + 1$.
(2) A point is merged into a random buffer from the hierarchy. The point must have been at level $l \le i - 1$, and the random buffer resides at level i, so the point moves up the hierarchy. If its u_i value is too small, it may disappear.
(3) A point is in a random buffer B that is merged with another random buffer B'. The random buffer B could not be at level greater than i before the merge, by definition, but the random buffer afterward is at level i. So the point's level does not decrease (it may stay the same). If the u_i value is too small, it may disappear.
(4) A point is in a random buffer when it performs an output operation. The random buffer was at level i, and the point is now at level i in the hierarchy.
(5) Both j and i increase. If the point remains in the hierarchy, it remains so at the same level. If it is now at level $i - 1$, it gets put in the random buffer at level i, and it may be dropped. If the point is in the random buffer, it remains there but the random buffer is now at level i where before it was at level $i - 1$. Again the point may disappear if too many points moved to the random buffer have larger u_i values. □

Now we analyze the error in this hybrid summary. We will focus on a single interval $I \in \mathcal{I}$ and show the over-count error X on I has $\mathsf{abs}(X) \leq \varepsilon n/2$ with probability $1 - \varepsilon\delta$. Then applying a union bound will ensure the summary is correct for all $1/\varepsilon$ intervals in $\mathcal{I}_\varepsilon$ with probability at least $1 - \delta$. This will imply that for all intervals $I \in \mathcal{I}$ the summary has error at most εn.

The total over-count error can be decomposed into two parts. First, we invoke Theorem 3.7 to show that the effect of all same-weight merges has error at most $\varepsilon n/4$ with probability at least $1 - \varepsilon\delta/2$. This step assumes that *all* of the data that ever comes out of the random buffer has no error, it is accounted for in the second step. Note that the total number of merge steps at each level is at most as many as in Theorem 3.7, even those merges that are later absorbed into the random buffer. Second, (the focus of our analysis) we show the total error from all points that pass through the random buffer is at most $\varepsilon n/4$ with probability at least $1 - \varepsilon\delta/2$. This step assumes that all of the weighted points put into the random buffer have no error, this is accounted for in the first step. So there are two types of random events that affect X: same-weight merges and random buffer outputs. We bound the effect of each event, independent of the result of any other event. Thus after analyzing the two types separately, we can apply the union bound to show the total error is at most $\varepsilon n/2$ with probability at least $1 - \varepsilon\delta$.

It remains to analyze the effect on I of the random buffer outputs. First we bound the number of times a random buffer can output to level l, i.e., output a set of k_ε points of weight 2^l each. Then we quantify the total error attributed to the random buffer output at level l.

LEMMA 3.9. *A summary of size n, for $2^{j-1}k_\varepsilon \leq n < 2^j k_\varepsilon$, has experienced $h_l \leq 2^{j-l} = 2^{i-l}g_\varepsilon$ random buffer promotions to level l within its entire merge history.*

PROOF. By Lemma 3.8, if a point is promoted from a random buffer to the hierarchy at level l, then it can only be put back into a random buffer at a level $l' > l$. Thus the random buffer can only promote, at a fixed level l, points with total weight $n < 2^j k_\varepsilon$. Since each promotion outputs points with a total weight of $2^l k_\varepsilon$, this can happen at most $h_l < 2^j k_\varepsilon / 2^l k_\varepsilon = 2^{j-l}$ times. The proof concludes using $g_\varepsilon = 2^{j-i}$. □

LEMMA 3.10. *When the random buffer promotes a set B of k_ε points representing a set P of m' points (where $m/2 < m' < 2m$), for any interval $I \in \mathcal{I}$ the over-count*

$$X = (m/k_\varepsilon)|I \cap B| - |I \cap P|$$

has expectation 0 and $\mathsf{abs}(X) \leq 2m$.

PROOF. The expectation of over-count X has two independent components. B is a random sample from P, so in expectation it has the correct proportion of points in any interval. Also, since $E[|P|] = m$, and $|B| = k_\varepsilon$, then m/k_ε is the correct scaling constant in expectation.

To bound $\mathsf{abs}(X)$, we know that $|P| < 2m$ by construction, so the maximum error an interval I could have is to return 0 when it should have returned $2m$, or vice-versa. So $\mathsf{abs}(X) < 2m$. □

Since $m \leq n/g_\varepsilon$ at level i, then $m \leq 2^{l-i}n/g_\varepsilon$ at level l, and we can bound the over-count error as $\Delta_l = \mathsf{abs}(X) \leq 2m \leq 2^{l-i+1}n/g_\varepsilon$. Now we consider a random buffer promotion that causes an over-count $X_{l,s}$ where $l \in [0, i]$ and $s \in [1, h_l]$. The expected value of $X_{l,s}$ is 0, and $\mathsf{abs}(X_{l,s}) \leq \Delta_l$. These events are independent so we can apply another Chernoff-Hoeffding bound on these $\sum_{l=0}^{i} h_l$ events. Recall that $g_\varepsilon = (64/\varepsilon^2)\ln(4/\varepsilon\delta)$

and let $\hat{T} = \sum_{i=0}^{i} \sum_{s=1}^{h_l} X_{l,s}$, which has expected value 0. Then

$$\begin{aligned}
\mathsf{Pr}[\mathsf{abs}(\hat{T}) \geq \varepsilon n/4] &= 2\exp\left(-2\frac{(\varepsilon n/4)^2}{\sum_{l=0}^{i} h_l \Delta_l^2}\right) \\
&\leq 2\exp\left(-2\frac{(\varepsilon n/4)^2}{\sum_{l=0}^{i} \left(2^{i-l} g_\varepsilon\right)\left(2^{l-i+1} n/g_\varepsilon\right))^2}\right) \\
&\leq 2\exp\left(-g_\varepsilon \frac{\varepsilon^2}{8} \frac{1}{\sum_{l=0}^{i} 2^{i-l} 2^{2(l-i)+2}}\right) \\
&= 2\exp\left(-g_\varepsilon \frac{\varepsilon^2}{32} \frac{1}{\sum_{l=0}^{i} 2^{l-i}}\right) \\
&= 2\exp\left(-2\ln(4/\varepsilon\delta) \frac{1}{\sum_{l=0}^{i} 2^{-l}}\right) \\
&\leq 2\exp(-\ln(4/\varepsilon\delta)) \\
&= 2(\varepsilon\delta/4) = \varepsilon\delta/2.
\end{aligned}$$

THEOREM 3.11. *A fully mergeable one-dimensional ε-approximation of size $O((1/\varepsilon)\log^{1.5}(1/\varepsilon\delta))$ can be maintained with probability at least $1-\delta$.*

We first note that, for $\delta \geq \varepsilon$, the bound above is always $O((1/\varepsilon)\log^{1.5}(1/\varepsilon))$. For $\delta < \varepsilon$, the bound is $O((1/\varepsilon)\log^{1.5}(1/\delta))$, but this can be further improved as follows. We run r independent copies of the structure each with failure probability $\delta' = \varepsilon/2$. Then for any interval query, we return the median value from all r such summaries. The query thus fails only when at least $r/2$ copies of the structure fail, which by the Chernoff bound, is at most

$$\left[\frac{e^{1/\varepsilon - 1}}{(1/\varepsilon)^{1/\varepsilon}}\right]^{\varepsilon r/2} < (e\varepsilon)^{1/\varepsilon \cdot \varepsilon r/2} = (e\varepsilon)^{r/2}.$$

Setting $r = O(\log(\delta)/\log(\varepsilon))$ will make the above probability at most $\varepsilon\delta$, which means that the structure will be correct for all interval queries with probability at least $1-\delta$. The total size of all r copies is thus $O((1/\varepsilon)\sqrt{\log(1/\varepsilon)}\log(1/\delta))$. Note that, however, the resulting structure is not technically a single ε-approximation.

COROLLARY 3.12. *A fully mergeable ε-approximate quantile summary can be maintained with probability at least $1-\delta$. The size is $O((1/\varepsilon)\log^{1.5}(1/\varepsilon\delta))$ for $\delta \geq \varepsilon$, and $O((1/\varepsilon)\sqrt{\log(1/\varepsilon)}\log(1/\delta))$ for $\delta < \varepsilon$.*

4. ε-APPROXIMATIONS OF RANGE SPACES

In this section, we generalize the approach of the previous section to ε-approximations of higher dimensional range spaces. Let D be a set of points in $\mathbb{R}^d$, and let $(D, \mathcal{R})$ be a range space of VC-dimension ν (see Section 1.2 for the definition). We use $\mathcal{R}_d$ to denote the set of ranges induced by a set of d-dimensional axis-aligned rectangles, i.e., $\mathcal{R}_d = \{D \cap \rho \mid \rho \text{ is a rectangle}\}$.

The overall merge algorithm is the same as in Section 3, except that we use a more intricate procedure for each same-weight merge operation of two summaries S_1 and S_2. Suppose $|S_1| = |S_2| = k$, and let $S' = S_1 \cup S_2$. Using the algorithm in [Bansal 2010], we compute a low-discrepancy coloring $\chi : S' \to \{-1, +1\}$ such that for any $R \in \mathcal{R}$, $\sum_{a \in S' \cap R} \chi(a) = O(k^{1/2 - 1/2\nu})$. Let $S^+ = \{a \in S' \mid \chi(a) = +1\}$ and $S^- = \{a \in S' \mid \chi(a) =$

$-1\}$. Then we choose to retain either S^+ or S^- at random as the merged summary S. We can then generalize Lemma 3.4 as follows.

LEMMA 4.1. *Given any range $R \in \mathcal{R}$, $2|S \cap R|$ is an unbiased estimator of $|S' \cap R|$ with error at most $\Lambda_\nu = O(k^{1/2-1/2\nu})$.*

For the range space $(P, \mathcal{R}_d)$, we can reduce the discrepancy of the coloring to $O(\log^{2d} k)$ using the algorithm by Phillips [2008], leading to the following generalization of Lemma 3.4.

LEMMA 4.2. *Given any range $R \in \mathcal{R}_d$, $2|S \cap R|$ is an unbiased estimator of $|S' \cap R|$ with error at most $\Lambda_d = O(\log^{2d} k)$.*

Lemma 3.5 and its proof generalize in a straightforward way, the only change being that now $\Delta_i = 2^{i-1}\Lambda_\nu$, hence implying $h > O(\Lambda_v \log^{1/2}(1/\varepsilon\delta))$ and

$$\varepsilon' = \varepsilon/h = \Omega(\varepsilon/(\Lambda_v \log^{1/2}(1/\varepsilon\delta))).$$

Solving $n/k_\varepsilon = n\varepsilon'$ for k_ε yields

$$\begin{aligned} k_\varepsilon =& \frac{1}{\varepsilon} O\left(\Lambda_\nu \sqrt{\log \frac{1}{\varepsilon\delta}}\right) \\ =& O\left(\frac{1}{\varepsilon}\left(k_\varepsilon^{1/2-1/(2\nu)}\right)\sqrt{\log \frac{1}{\varepsilon\delta}}\right) \\ =& O\left(\left(\frac{1}{\varepsilon}\right)^{2\nu/(\nu+1)} \log^{\nu/(\nu+1)}\left(\frac{1}{\varepsilon\delta}\right)\right). \end{aligned}$$

For $(P, \mathcal{R}_d)$ we get

$$k_\varepsilon = O\left(\frac{1}{\varepsilon}\Lambda_d\sqrt{\log(1/\varepsilon\delta)}\right) = O\left(\frac{1}{\varepsilon}\log^{2d}\left(\frac{\log(1/\delta)}{\varepsilon}\right)\sqrt{\log(1/\varepsilon\delta)}\right).$$

Applying the rest of the machinery as before we can achieve ε-approximations of size k_ε under same-weight merges, with probability at least $1-\delta$.

LEMMA 4.3. *For a range space $(D, \mathcal{R})$ of VC-dimension ν, an ε-approximation of size $O((1/\varepsilon)^{2\nu/(\nu+1)} \log^{\nu/(\nu+1)}(1/\varepsilon\delta))$ can be maintained under the framework of same-weight merges, with probability at least $1-\delta$. For the range space $(D, \mathcal{R}_d)$, the size of the ε-approximation is $O((1/\varepsilon)\log^{2d}(\log(1/\delta)/\varepsilon)\cdot\sqrt{\log(1/\varepsilon\delta)})$.*

This algorithm extends to different-weight merges with an extra $\log(n\varepsilon)$ factor, as with the one-dimensional ε-approximation. The random buffer maintains a random sample of the same asymptotic size $O((1/\varepsilon^2)\log(1/\delta))$ and 0 expected over-count error. Substituting the increased k_ε value, the generalizations of Lemma 3.9 and Lemma 3.10 follow.

THEOREM 4.4. *A mergeable ε-approximation of a range space $(D, \mathcal{R})$ of VC-dimension ν of size $O((1/\varepsilon^{\frac{2\nu}{\nu+1}})\log^{\frac{2\nu+1}{\nu+1}}(1/\varepsilon\delta))$ can be maintained with probability at least $1-\delta$. If $\mathcal{R} = \mathcal{R}_d$, then the size is $O\left((1/\varepsilon)\log^{2d+3/2}(1/\varepsilon\delta)\right)$.*

As with the one-dimensional ε-approximations, for $\delta < \varepsilon$, we can replace all but one $\log(1/\varepsilon\delta)$ by $\log(1/\varepsilon)$, by running $O(\log(\delta)/\log(\varepsilon))$ independent copies of the structure, each with failure probability $\delta' = \varepsilon/2$. The resulting structure is not technically a

single ε-approximation, but it can be used to answer all range counting queries within εn error, with probability at least $1 - \delta$.

5. ε-KERNELS

A unit vector $u \in \mathbb{S}^{d-1}$ defines a direction in $\mathbb{R}^d$, and a point $p \in \mathbb{R}^d$ is projected to the line through u using the inner product $\langle u, p \rangle$. Given a point set $P \subset \mathbb{R}^d$, the *extent* in direction u is $\mathsf{E}[P, u] = \max_{p \in P} \langle u, p \rangle$, and the width $\mathsf{wid}[P, u] = \mathsf{E}[P, u] + \mathsf{E}[P, -u]$. An *$\varepsilon$-kernel* is a subset $K \subset P$ such that over all directions u,

$$\max_{u \in \mathbb{S}^{d-1}} \frac{\mathsf{E}[P, u] - \mathsf{E}[K, u]}{\mathsf{wid}[P, u]} \leq \varepsilon.$$

An ε-kernel of size $O(1/\varepsilon^{(d-1)/2})$ can be computed in time $O(|P| + 1/\varepsilon^{d-3/2})$ [Chan 2006; Yu et al. 2004]. It is also known that the union of two ε-kernels is an ε-kernel [Chan 2006], but this observation alone is not sufficient to have mergeable ε-kernels of size $1/\varepsilon^{O(1)}$ since the size of the kernel doubles after taking the union. We therefore need a merge procedure that reduces $K_1 \cup K_2$ to an appropriate size without increasing the error.

Reference frames and ε-kernel basics. We say a point set P is *β-fat* if over all directions u, v we can bound the width ratio $\max_{u,v}(\mathsf{wid}(P, u)/\mathsf{wid}(P, v)) \leq \beta$. Given a box $B \supset P$, P is *β-fat with respect to B* if

$$\max_{u,v \in \mathbb{S}^{d-1}} (\mathsf{wid}(B, u)/\mathsf{wid}(P, v)) \leq \beta.$$

If β is less than some fixed constant (that depends only on d) we say that P is just *fat* (with respect to B). B represents a *reference frame* in that it fixes a set of axes, as well as a relative scale along those axes. That is, the d orthogonal directions the of box's face normals $\{b_1, \ldots, b_d\}$ define coordinate axes, and the width of the box in each of these directions provides a relative scale of the contained point sets. Given P and B, we will use this reference frame to construct/merge kernels.

Most standard techniques to create ε-kernels use the following observations.

— Let A be an affine transform. If K is an ε-kernel of P, then $A(K)$ is an ε-kernel of $A(P)$ [Agarwal et al. 2004].
— Let $I = [-1, 1]^d$ and $\beta_d = 2^d d^{5/2} d!$. There exists an $O(d^2|P|)$ time algorithm to construct an affine transform A such that $A(P) \subset I$ and $A(P)$ is β_d-fat with respect to I [Barequet and Har-Peled 2001; Har-Peled 2010].
— Place a grid G_ε on I so that each grid cell has width ε/β_d. For each grid cell $g \in G_\varepsilon$, place one point (if it exists) from $g \cap A(P)$ in K. Then K is an ε-kernel of $A(P)$. Clearly the same holds if for each column in the grid, you only retain the most extreme such points, reducing the size to $O(1/\varepsilon^{d-1})$ [Agarwal et al. 2004].

Because of the first two observations, for static ε-kernel algorithms it is convenient to simply assume that $P \subset I$ and P is fat with respect to I. The main difficulty in incremental ε-kernel algorithms is maintaining such a reference frame I.

The size of ε-kernels can be reduced to $O(1/\varepsilon^{(d-1)/2})$ using an additional trick [Agarwal et al. 2004], given that we have an $(\varepsilon/3)$-kernel K_1 of P that is β-fat with respect to I. Consider a sphere S of radius $\sqrt{d} + 1$ centered at the origin, and place a set Q_ε of $O(1/(\varepsilon\beta)^{(d-1)/2})$ evenly spaced points on S. For each point $q \in Q_\varepsilon$ place the closest point to q from K_1 into K. Then K is an ε-kernel of P (of size $O(1/\varepsilon^{(d-1)/2})$).

Mergeable ε-kernels in a common reference frame. Let B be a d-dimensional box, and let $P_1, P_2 \subset I$ be two point sets that are fat with respect to B. A result in [Agarwal

et al. 2004] implies that we can assume $B = [-1, +1]^d$. We can now create mergeable ε-kernels for P_1 and P_2. More precisely, we create ε-kernels K_1, K_2 of P_1, P_2, respectively, such that (1) $|K_1|, |K_2| \leq b_{\varepsilon,d} = c(1/\varepsilon)^{(d-1)/2}$ for some constant c, and (2) from K_1 and K_2 we can create an ε-kernel K of $P = P_1 \cup P_2$ such that $|K| \leq b_{\varepsilon,d}$.

First we create K_1 and K_2 using the three observations above and the additional trick to reduce the size. For each point $q \in Q_\varepsilon$, we retain in K, the merged ε-kernel of $P_1 \cup P_2$, the closest point to q in $K_1 \cup K_2$. This approach can be used to merge as many ε-kernels as desired, without increasing the ε error factor or the size beyond $b_{\varepsilon,d}$, provided each of the input point sets is fat with respect to the same box B.

THEOREM 5.1. *A mergeable ε-kernels of size $O(1/\varepsilon^{(d-1)/2})$ can be maintained, assuming all input point sets are fat with respect to a fixed box.*

Although the requirement of being fat with respect to the same box may seem too restrictive, it is a reasonable assumption for most data sets in practice. Given a distributed data set, we probably have upper bounds and rough lower bounds on the total extent of point sets. This is enough to provide a bounding box for which the point sets are fat with respect to. Even if one partition of the data set is not fat with respect to the full bounding box (it may be a small localized subset), the full result will be fat.

We leave open the question of maintaining a mergeable ε-kernel that does not restrict the points to a fixed reference frame.

6. EXPERIMENTS

6.1. Experiment setup

We chose to experiment with our algorithms on the routing tree of a simulated sensor network, as it is irregular, unbalanced, with possibly long branches, a situation where mergeability is an appealing property. Both the heavy hitters and the quantiles problems have been studied in the sensor network setting, under the name *data aggregation*. Here we compare our algorithms with the previous best known heavy hitters algorithms [Manjhi et al. 2005; Manjhi et al. 2005] as well as the best quantile algorithms [Greenwald and Khanna 2004; Shrivastava et al. 2004; Huang et al. 2011]. Note that all these algorithms do not produce mergeable summaries and need to know the structure of the routing tree (size or height) in order to set their parameters correctly. On the other hand, all our algorithms only need a single parameter ε, the target error bound.

We used the standard procedure (as in e.g. [Shrivastava et al. 2004]) for generating a sensor network and its routing tree. Specifically, we first distributed 1024 sensor nodes over a unit square uniformly at random. The sensor nodes have a fixed radio range, and two nodes can be connected if they are within the radio range of each other. Then we randomly picked a node as the root. Then starting from the root, we grew a routing tree by a breadth-first search.

For each experiment, we first give each sensor node some initial data, which is used to compute the initial local summary at the node. Then starting from the leaves, we merge the summaries in a bottom-up fashion, all the way to the root. We measure the maximum summary size in this merging process, as well as the actual error observed in the final summary at the root, which could be smaller than the error parameter ε.

6.2. Heavy hitters

In this subsection we experimentally evaluate our two merging algorithms for the heavy hitters problem (MERGEABLEMINERROR and MERGEABLEMINSPACE), comparing with two previous algorithms: the TRIBUTARYANDDELTA algorithm [Manjhi et al. 2005] and the MINMAXLOAD algorithm in [Manjhi et al. 2005]. These

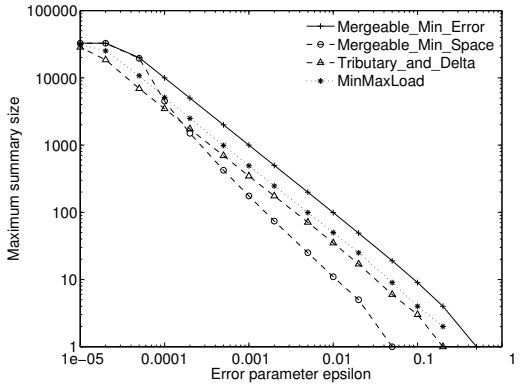

Fig. 2: Heavy hitters: ε vs. summary size.

two algorithms follow the same framework: They first compute a *precision gradient* $\varepsilon(1) \leq \varepsilon(2) \leq \cdots \leq \varepsilon(h) = \varepsilon$, where h is the height of the tree (so need to know h in advance). For level-1 nodes (leaf nodes), they compute summaries with error $\varepsilon(1)$. As summaries are merged along the tree, the errors gradually grow following the precision gradient and eventually reach ε when it comes to the root. The two algorithms use essentially the same merging algorithm, and only differ in the way the precision gradient is computed. The final error is guaranteed but there is no guarantee on the summary size.

We generated synthetic data following Zipf distribution for the experiments. More precisely, we first generated a total of 1 billion values from a Zipf distribution with parameter $\alpha = 1$ with items in the range $[0, 2^{32} - 1]$. Then we randomly distributed these 1 billion integers to the 1024 nodes. There are roughly $33,000$ distinct items in this data set.

ε vs. summary size. We set the error parameter ε from 1 to 10^{-5} and run all four algorithms on the same sensor network and the same data. The results are plotted in Figure 2 in log-log scale. From the figure, we see that the general conclusion is that MERGEABLEMINSPACE produces smaller summaries than TRIBUTARYANDDELTA and MINMAXLOAD, which in turn produce smaller summaries than MERGEABLEMINERROR, except when $1/\varepsilon$ approaches the number of distinct items (recall that there are $33,000$ distinct items in the data set). We also observe that the summary size of MERGEABLEMINERROR is almost always $k = 1/\varepsilon - 1$. This is as expected, since MERGEABLEMINERROR tries to eliminate counters only when there are more than k counters.

ε vs. actual error. We also examined the actual errors in the final summary received by the root of the routing tree. Specifically, we extract the frequencies of the $1/\varepsilon$ most frequent items from the summary, compute their differences with the true frequencies, and take the maximum and average of these errors. We also divide the error by n so that it can be compared with ε. The results are plotted in Figure 3. A subtle issue when it comes to measuring actual errors is that, although MG and SpaceSaving are isomorphic, they give different estimates: MG gives lower bounds while SpaceSaving gives upper bounds. It has been observed [Cormode and Hadjieleftheriou 2008b] that SpaceSaving tends to give better estimates in practice, so we actually convert the final summary at the root to SpaceSaving before extracting the item frequencies, though in the merging process we always work with MG summaries.

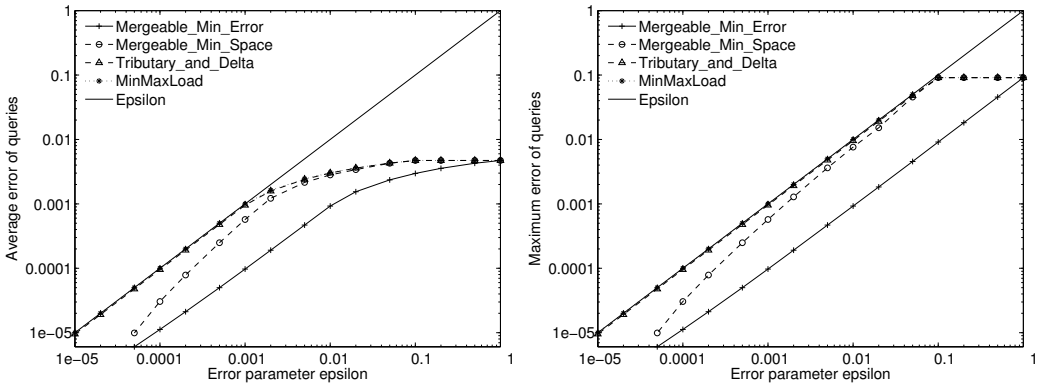

Fig. 3: Heavy hitters: ε vs. actual error.

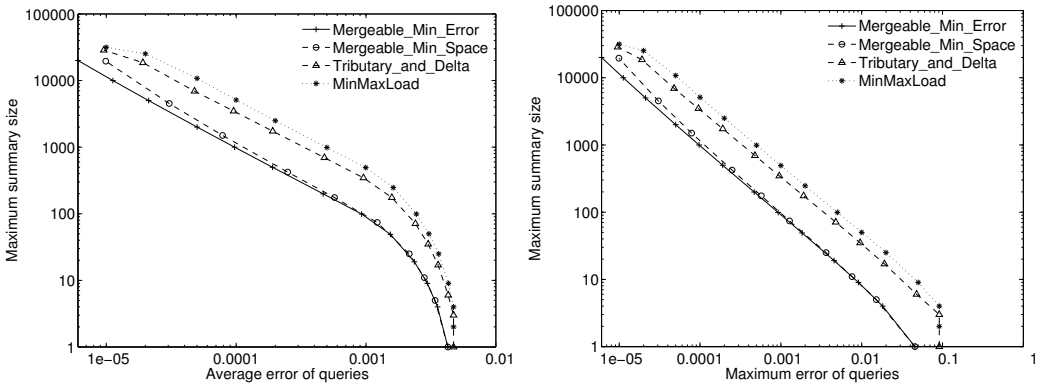

Fig. 4: Heavy hitters: actual error vs. summary size.

From Figure 3 we see that TRIBUTARYANDDELTA and MINMAXLOAD produce errors that are very close to the error guarantee ε. This is especially true for small ε, where even the average error is close to ε. On the other hand, the actual error of MERGEABLEMINERROR is 5 to 10 times smaller than ε. MERGEABLEMINSPACE also produces smaller actual errors but not as much. This is because the error bound of the MG summary is actually $\varepsilon(n-\hat{n})$, while MERGEABLEMINERROR always gives a larger $\hat{n}$. It We also note that the, for small ε, the average error is close to the maximum error. When ε becomes very large, most items are not included in the summary, and so have no estimate. Thus, the average error tracks the average count, while the maximum error approaches the maximum count. This explains the flattening of the curves for all four algorithms.

Actual error vs. summary size. We also plot the actual error vs. summary size tradeoff in Figure 4, to better understand the performance of the algorithms. It is not surprising to see that MERGEABLEMINERROR and MERGEABLEMINSPACE have almost the same tradeoff. After all, they are fundamentally the same: the difference is that one favors error while the other favors space. Meanwhile, TRIBUTARYANDDELTA

Table II: Maximum summary size with the bloom tree.

ε	MERGEABLEMINSPACE	MERGEABLEMINERROR	TRIBUTARYANDDELTA	MINMAXLOAD
1	0	0	14	2
0.5	0	1	14	2
0.2	4	4	4097	8194
0.1	9	9	8194	8197
0.05	15	19	12291	12293
0.02	47	49	20485	20492
0.01	93	99	28679	28681
0.005	192	199	36873	36873
0.002	490	498	40970	40972
0.001	994	999	49164	49164
0.0005	1999	1999	53261	53261
0.0002	4999	4999	57358	57358
0.0001	9999	9999	57372	57386

and MINMAXLOAD also exhibit very similar tradeoffs, which are worse than that of our algorithms.

A contrived example. Finally, we designed a contrived example, where the merging tree consists of a linked list of 4096 nodes, with the lowest node attached to 4096 leaves. Each node receives a carefully chosen data set of 8192 items, and no two nodes share any common items. On this example, both TRIBUTARYANDDELTA and MINMAXLOAD failed miserably, as shown in Table II (note that these two algorithms are still given the knowledge of the height of the tree). Such examples do not exist for our algorithms, due to their theoretical guarantees.

6.3. Quantiles

Our $O((1/\varepsilon)\log^{1.5}(1/\varepsilon))$ algorithm is moderately complex to implement, but the one presented in Section 3.2 is more direct to use. It has a slightly worse bound of $O((1/\varepsilon)\log^{0.5}(1/\varepsilon)\log(\varepsilon n))$ but it is very simple. Recall that its only operation is to merge two sorted lists and then take even or odd positioned items randomly, combined with the standard logarithmic method. In this subsection we experimentally compare it with three existing algorithms for computing quantiles in a sensor network: *GK* [Greenwald and Khanna 2004] is a quantile summary of size $O((1/\varepsilon)\log n\log(h/\varepsilon))$. It is not mergeable as it requires the knowledge of both n, the total number of items in the sensor network, as well as h, the height of the routing tree. *q-digest* [Shrivastava et al. 2004] is actually a mergeable quantile summary as mentioned in Section 1.2, but it needs a bounded universe $\{0,\ldots,u-1\}$. In our experiments, we choose to use integers in the range $\{0,\cdots,2^{32}-1\}$ so that q-digest can be applied. It has size $O((1/\varepsilon)\log u)$. *SB-p* [Huang et al. 2011] is a sampling based, randomized quantile summary for sensor networks. It has size $O((1/\varepsilon)\log(k/h))$[5] where k is the number of nodes in the sensor network. This algorithm needs to know n, k, and h to set its parameters correctly to work.

We used the same sensor network and its routing tree as in the heavy hitters experiments. For the data sets, we first generated a total of $n = 1$ billion floating-point numbers from a Gaussian distribution with mean 0 and variance 1. These values were then scaled to the range $[0, 2^{32}-1]$ and round them to integers. Then we randomly distributed these n integers to the 1024 nodes in the sensor network.

[5]This bound is for getting one quantile within ε-error with constant probability. For getting all quantiles right (as we do in this paper), the bound becomes $O((1/\varepsilon)\log(k/h)\log(1/\varepsilon))$.

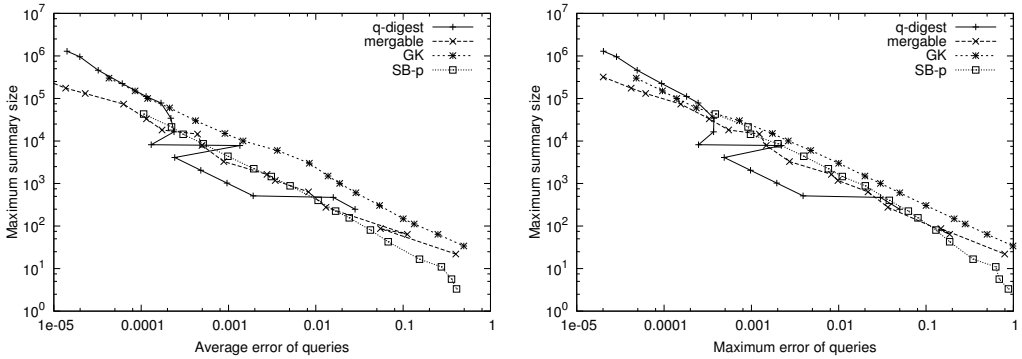

Fig. 5: Quantiles: actual error vs. summary size.

Since two of these four algorithms are deterministic and two are randomized, it will not be fair to compare them with the same error parameter ε. The deterministic algorithms provide a worst-case ε-error guarantee while the randomized ones only provide probabilistic guarantees. So we directly look at the actual error vs. summary size tradeoff. For summary size, we as before measure the maximum summary produced by any node. For the actual error, we query the final summary at the root for the 1%-quantile, the 2%-quantile, ..., 99%-quantile, compute the differences from their true percentiles in the whole data set, and then take the maximum and average.

In Figure 5 we plot the actual error vs. summary size tradeoff curves of the four algorithms, where the actual error is either the average error or the maximum of the 100 quantiles extracted from the summary. When average error is considered, we see that SB-p and our algorithm exhibit similar behavior, and both are better than GK. When we look at the maximum error, however, the three algorithms are similar. Arguably SB-p and our algorithms demonstrate slightly better performance. This could be due to the randomness and the determinism of these algorithms: The randomized algorithms produce small errors most of the time, but may be off for a few quantiles, while the deterministic GK algorithm has more consistent errors for all quantiles. q-digest is generally also very good, but its behavior is sometimes erratic: the actual error might suddenly go down even when ε becomes larger. Recall also that q-digest depends crucially on the bound u on the size of the universe, and does not apply when such a bound is not known in advance (e.g. when the input domain consists of arbitrary floating point or string values).

Finally, we note that the comparison between q-digest, GK, and SB-p[6] has been previously made in [Huang et al. 2011]. Here we further compare our new mergeable quantile summary. The general conclusion is that it has similar performance as SB-p, but with the additional mergeable property, which also implies that it does not need the knowledge of the merging tree as SB-p does.

7. CONCLUDING REMARKS

We have formalized the notion of mergeable summaries, and demonstrated fully mergeable summaries for the central problems of heavy hitters, quantiles, ε-approximations and ε-kernels. The obvious open question is for what other problems do there exist fully mergeable summaries. In some cases, it may be possible to adapt existing solutions from the streaming literature to this setting. For example, consider

[6]Another algorithm, SB-1, was also introduced in [Huang et al. 2011], but since it has similar performance as SB-p, so we omit it here.

the problem of k-median clustering. Guha et al. [2000] show that clustering the union of cluster centers from disjoint parts of the input gives a guaranteed approximation to the overall clustering. In our terminology, this means that clusterings can be merged, although since the accuracy degrades by a constant amount each time, we may think of this as a one-way merge algorithm. Similarly, results on k-center clustering on the stream can generate a mergeable summary of size $O((k/\varepsilon)\log(1/\varepsilon))$ that provides a $2+\varepsilon$ guarantee [Guha 2009].

Recently, mergeable summaries for graph data were proposed [Ahn et al. 2012] for problems such as connectivity, k-connectivity, and bipartiteness. However, there are many other problems in the domain of high-dimensional data, geometric data and graph data for which no mergeable summary is known or for which bounds are not tight.

REFERENCES

AGARWAL, P. K., CORMODE, G., HUANG, Z., PHILLIPS, J. M., WEI, Z., AND YI, K. 2012. Mergeable summaries. In *Proceedings 31st ACM Symposium on Principals of Database Systems*. 23–34.

AGARWAL, P. K., HAR-PELED, S., AND VARADARAJAN, K. R. 2004. Approximating extent measures of points. *Journal of the ACM 51,* 4, 606–635.

AGARWAL, P. K., PHILLIPS, J. M., AND YU, H. 2010. Stability of ε-kernels. In *Proc. European Symposium on Algorithms*. Springer-Verlag, 487–499.

AGARWAL, P. K. AND YU, H. 2007. A space-optimal data-stream algorithm for coresets in the plane. In *Proc. Annual Symposium on Computational Geometry*. ACM, 1–10.

AHN, K. J., GUHA, S., AND MCGREGOR, A. 2012. Analyzing graph structure via linear measurements. In *Proc. ACM-SIAM Symposium on Discrete Algorithms*.

ALON, N., MATIAS, Y., AND SZEGEDY, M. 1999. The space complexity of approximating the frequency moments. *Journal of Computer and System Sciences 58,* 1, 137–147.

BANSAL, N. 2010. Constructive algorithms for discrepancy minimization. In *Proc. IEEE Symposium on Foundations of Computer Science*. IEEE, 3–10.

BANSAL, N. 2012. Semidefinite optimization in discrepancy theory. *Math. Program. 134,* 1, 5–22.

BAR-YOSSEF, Z., JAYRAM, T. S., KUMAR, R., SIVAKUMAR, D., AND TREVISAN, L. 2002. Counting distinct elements in a data stream. In *RANDOM*.

BAREQUET, G. AND HAR-PELED, S. 2001. Efficiently approximating the minimum-volume bounding box of a point set in three dimensions. *Journal of Algorithms 38,* 1, 91–109.

BERINDE, R., CORMODE, G., INDYK, P., AND STRAUSS, M. 2010. Space-optimal heavy hitters with strong error bounds. *ACM Transactions on Database Systems 35,* 4.

CHAN, T. 2006. Faster core-set constructions and data-stream algorithms in fixed dimensions. *Computational Geometry 35,* 1, 20–35.

CHAN, T. 2009. Dynamic coresets. *Discrete and Computational Geometry 42,* 3, 469–488.

CHAZELLE, B. 2000. The discrepancy method: randomness and complexity.

CHAZELLE, B. AND MATOUSEK, J. 1996. On linear-time deterministic algorithms for optimization problems in fixed dimension. *Journal of Algorithms 21,* 3, 579–597.

CORMODE, G. AND HADJIELEFTHERIOU, M. 2008a. Finding frequent items in data streams. *Proceedings of the VLDB Endowment 1,* 2, 1530–1541.

CORMODE, G. AND HADJIELEFTHERIOU, M. 2008b. Finding frequent items in data streams. In *Proc. International Conference on Very Large Data Bases*.

CORMODE, G. AND MUTHUKRISHNAN, S. 2005. An improved data stream summary: The count-min sketch and its applications. *Journal of Algorithms 55,* 1, 58–75.

FEIGENBAUM, J., KANNAN, S., STRAUSS, M. J., AND VISWANATHAN, M. 2003. An approximate L1-difference algorithm for massive data streams. *SIAM J. Comput. 32,* 1, 131–151.

FELDMAN, J., MUTHUKRISHNAN, S., SIDIROPOULOS, A., STEIN, C., AND SVITKINA, Z. 2008. On distributing symmetric streaming computations. In *Proc. ACM-SIAM Symposium on Discrete Algorithms*.

GANGULY, S. AND MAJUMDER, A. 2007. CR-precis: A deterministic summary structure for update data streams. In *ESCAPE*.

GILBERT, A. C., KOTIDIS, Y., MUTHUKRISHNAN, S., AND STRAUSS, M. J. 2002. How to summarize the universe: Dynamic maintenance of quantiles. In *Proc. International Conference on Very Large Data Bases*.

GREENWALD, M. AND KHANNA, S. 2001. Space-efficient online computation of quantile summaries. In *Proc. ACM SIGMOD International Conference on Management of Data*.

GREENWALD, M. AND KHANNA, S. 2004. Power conserving computation of order-statistics over sensor networks. In *Proc. ACM Symposium on Principles of Database Systems*.

GUHA, S. 2009. Tight results for clustering and summarizing data streams. In *Proc. International Conference on Database Theory*. ACM, 268–275.

GUHA, S., MISHRA, N., MOTWANI, R., AND O'CALLAGHAN, L. 2000. Clustering data streams. In *Proc. IEEE Symposium on Foundations of Computer Science*. IEEE, 359–366.

HAR-PELED, S. 2010. Approximation algorithm in geometry(chapter 22).

HUANG, Z., WANG, L., YI, K., AND LIU, Y. 2011. Sampling based algorithms for quantile computation in sensor networks. In *Proc. ACM SIGMOD International Conference on Management of Data*.

KANE, D. M., NELSON, J., AND WOODRUFF, D. P. 2010. An optimal algorithm for the distinct elements problem. In *Proc. ACM Symposium on Principles of Database Systems*.

LARSEN, K. 2011. On range searching in the group model and combinatorial discrepancy. In *Proc. IEEE Symposium on Foundations of Computer Science*. IEEE, 542–549.

LI, Y., LONG, P., AND SRINIVASAN, A. 2001. Improved bounds on the sample complexity of learning. *Journal of Computer and System Sciences 62,* 3, 516–527.

LOVETT, S. AND MEKA, R. 2012. Constructive discrepancy minimization by walking on the edges. In *Proceedings 53rd Annual IEEE Symposium on Foundations of Computer Science*.

MADDEN, S., FRANKLIN, M. J., HELLERSTEIN, J. M., AND HONG, W. 2002. TAG: a tiny aggregation service for ad-hoc sensor networks. In *Proc. Symposium on Operating Systems Design and Implementation*.

MANJHI, A., NATH, S., AND GIBBONS, P. B. 2005. Tributaries and deltas: efficient and robust aggregation in sensor network streams. In *Proc. ACM SIGMOD International Conference on Management of Data*.

MANJHI, A., SHKAPENYUK, V., DHAMDHERE, K., AND OLSTON, C. 2005. Finding (recently) frequent items in distributed data streams. In *Proc. IEEE International Conference on Data Engineering*.

MANKU, G. S., RAJAGOPALAN, S., AND LINDSAY, B. G. 1998. Approximate medians and other quantiles in one pass and with limited memory. In *Proc. ACM SIGMOD International Conference on Management of Data*.

MATOUŠEK, J. 1991. Approximations and optimal geometric divide-and-conquer. In *Proc. ACM Symposium on Theory of Computing*. ACM, 505–511.

MATOUŠEK, J. 2010. *Geometric discrepancy: An illustrated guide*. Vol. 18. Springer Verlag.

METWALLY, A., AGRAWAL, D., AND ABBADI, A. 2006. An integrated efficient solution for computing frequent and top-k elements in data streams. *ACM Transactions on Database Systems 31,* 3, 1095–1133.

MISRA, J. AND GRIES, D. 1982. Finding repeated elements. *Sc. Comp. Prog. 2*, 143–152.

NELSON, J. AND WOODRUFF, D. P. 2010. Fast manhattan sketches in data streams. In *Proc. ACM Symposium on Principles of Database Systems*.

PHILLIPS, J. 2008. Algorithms for ε-approximations of terrains. *Automata, Languages and Programming*, 447–458.

SHRIVASTAVA, N., BURAGOHAIN, C., AGRAWAL, D., AND SURI, S. 2004. Medians and beyond: New aggregation techniques for sensor networks. In *Proc. ACM SenSys*.

SURI, S., TÓTH, C., AND ZHOU, Y. 2006. Range counting over multidimensional data streams. *Discrete and Computational Geometry 36,* 4, 633–655.

TALAGRAND, M. 1994. Sharper bounds for gaussian and empirical processes. *The Annals of Probability 22,* 1, 28–76.

VAPNIK, V. AND CHERVONENKIS, A. 1971. On the uniform convergence of relative frequencies of events to their probabilities. *Theory of Probability and its Applications 16*, 264–280.

YU, H., AGARWAL, P. K., POREDDY, R., AND VARADARAJAN, K. R. 2004. Practical methods for shape fitting and kinetic data structures using core sets. In *Proc. Annual Symposium on Computational Geometry*. ACM, 263–272.

ZARRABI-ZADEH, H. 2011. An almost space-optimal streaming algorithm for coresets in fixed dimensions. *Algorithmica 60,* 1, 46–59.