



# Mergeable Summaries

Pankaj K. Agarwal<sup>\*</sup>  
Duke University  
pankaj@cs.duke.edu

Graham Cormode  
AT&T Labs–Research  
graham@research.att.com

Zengfeng Huang  
HKUST  
huangzsf@cse.ust.hk

Jeff M. Phillips  
University of Utah  
jeffp@cs.utah.edu

Zhewei Wei  
HKUST  
wzxac@cse.ust.hk

Ke Yi<sup>†</sup>  
HKUST  
yike@cse.ust.hk

## ABSTRACT

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 union of the two data sets, while preserving the error and size guarantees. This property means that the summaries can be merged in a way like 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  $\varepsilon$ -approximate heavy hitters, there is a deterministic mergeable summary of size  $O(1/\varepsilon)$ ; for  $\varepsilon$ -approximate quantiles, there is a deterministic summary of size  $O(\frac{1}{\varepsilon} \log(\varepsilon n))$  that has a restricted form of mergeability, and a randomized one of size  $O(\frac{1}{\varepsilon} \log^{3/2} \frac{1}{\varepsilon})$  with full mergeability. We also extend our results to geometric summaries such as  $\varepsilon$ -approximations and  $\varepsilon$ -kernels.

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

<sup>\*</sup>Supported by NSF under grants CNS-05-40347, IIS-07-13498, CCF-09-40671, and CCF-1012254, by ARO grants W911NF-07-1-0376 and W911NF-08-1-0452, by an NIH grant 1P50-GM-08183-01, and by a grant from the U.S.–Israel Binational Science Foundation.

<sup>†</sup>Supported by an RPC grant from HKUST and a Google Faculty Research Award.

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

PODS'12, May 21–23, 2012, Scottsdale, Arizona, USA.

Copyright 2012 ACM 978-1-4503-1248-6/12/05 ...\$10.00.

## Categories and Subject Descriptors

F.2.2 [Analysis of algorithms and problem complexity]: Nonnumerical algorithms and problems

## General Terms

Algorithms, theory

## Keywords

Summaries, streaming algorithms

## 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  $\varepsilon$ -approximations and  $\varepsilon$ -kernels, and to graph summaries like distance oracles. Note that the error parameter  $\varepsilon$  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  $\varepsilon$ -summaries of two (separate) data sets to obtain an  $\varepsilon$ -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 commuta-

tive 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 [18], 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 [18] 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 [18] 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 [29]: 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 [13, 22, 30, 31]. To obtain any overall guarantee, it is necessary to bound the number of rounds of merging operations so that the error parameter  $\varepsilon$  can be scaled 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  $\varepsilon$ ), 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  $D$  and an error parameter  $\varepsilon$ ,  $S()$  may have many valid outputs (e.g., depending on the order in which it processes  $D$ , it may return different valid  $\varepsilon$ -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  $\varepsilon$  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  $S(D_1 \uplus D_2, \varepsilon)$ <sup>1</sup> 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 additional 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.

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  $D$  are trivially mergeable. These include the  $F_2$  AMS sketch [4], the Count-Min sketch [15], the  $\ell_1$  sketch [17, 37], 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 [6, 26]. However, several fundamental problems have summaries that are based on other techniques, and are not known to be mergeable (or have unsatisfactory bounds). This paper focuses on summaries for several key problems, which are widely applicable. We develop both randomized and deterministic 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  $O(\log \frac{1}{\delta})$  independent summaries.

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  $[u] = \{0, \dots, u - 1\}$  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  $\varepsilon$ -approximate frequency estimation summary of  $D$  can be used to estimate  $f(x)$  for any  $x$  within an additive error of  $\varepsilon n$ . A heavy hitters summary allows one to extract all frequent items approximately, i.e., for a user-specified  $\phi$ , it returns all items  $x$  with

<sup>1</sup> $\uplus$  denotes multiset addition.

$f(x) > \phi n$ , 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 [15] of size  $O(\frac{1}{\varepsilon} \log u)$ , 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  $O(\frac{1}{\varepsilon} \log u \log(\frac{\log u}{\varepsilon}))$  from the extra sketches with adjusted parameters. The Count-Min sketch is randomized; while there is also a deterministic linear sketch for the problem [19], its size is  $O(\frac{1}{\varepsilon^2} \log^2 u \log n)$ . 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 [36] and the SpaceSaving summary [35], have been reported [14] 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  $O(\frac{1}{\varepsilon})$ . 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 [8]. Some merging algorithms for these summaries have been previously proposed, but the error increases after each merging step [30, 31].

**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  $\phi$ -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  $\varepsilon$ -approximate  $\phi$ -quantile is an element with rank between  $(\phi - \varepsilon)n$  and  $(\phi + \varepsilon)n$ , and a quantile summary allows us to extract an  $\varepsilon$ -approximate  $\phi$ -quantile for any  $0 < \phi < 1$ . It is well known [14] that the frequency estimation problem can be reduced to an  $\varepsilon'$ -approximate quantile problem for some  $\varepsilon' = \Theta(\varepsilon)$ , by identifying elements that are quantiles for multiples of  $\varepsilon'$  after tie breaking. Therefore, a quantile summary is automatically a frequency estimation summary (ignoring a constant-factor difference in  $\varepsilon$ ), but not vice versa.

Quite a number of quantile summaries have been designed [15, 20–22, 32, 39], 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  $O(\frac{1}{\varepsilon} \log^2 u \log(\frac{\log u}{\varepsilon}))$  after adjusting parameters [15]. The q-digest [39] has size  $O(\frac{1}{\varepsilon} \log u)$ ; 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 [21], which guarantees a size of  $O(\frac{1}{\varepsilon} \log(\varepsilon n))$ . A merging algorithm has been previously designed, but the error could increase to  $2\varepsilon$  when two  $\varepsilon$ -summaries are merged [22].

**$\varepsilon$ -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  $\varepsilon$ -approximation of

$(D, \mathcal{R})$  if

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

where  $\text{abs}(x)$  denotes the absolute value of  $x$ . Over the last two decades,  $\varepsilon$ -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<sup>2</sup>  $\nu$ , a random sample of  $O(1/\varepsilon^2(\nu + \log(1/\delta)))$  points from  $D$  is an  $\varepsilon$ -approximation with probability at least  $1 - \delta$  [28, 42]. 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  $\varepsilon$ -approximation of size  $O((1/\varepsilon) \log^{d+1/2}(1/\varepsilon))$  [27], and an  $\varepsilon$ -approximation of size  $O((1/\varepsilon) \log^{2d}(1/\varepsilon))$  [38] can be computed efficiently. More generally, an  $\varepsilon$ -approximation of size  $O(1/\varepsilon^{2\nu/(\nu+1)})$  exists for a range space of VC-dimension  $\nu$  [34]. Furthermore, such an  $\varepsilon$ -approximation can be constructed using Bansal’s algorithm [5]; see also [11, 34].

These algorithms for constructing  $\varepsilon$ -approximations are not known to be mergeable. Although they proceed by partitioning  $D$  into small subsets, constructing  $\varepsilon$ -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 Matousek [34] and Chazelle [12]) has gone into bounding the discrepancy, which governs the increase in error at each step. We are unaware of any mergeable  $\varepsilon$ -approximations of  $o(1/\varepsilon^2)$  size.

**$\varepsilon$ -kernels.** Finally, we consider  $\varepsilon$ -kernels [1] which are summaries for approximating the convex shape of a point set  $P$ . Specifically, they are a specific type of coresset that approximates the width of  $P$  within a relative  $(1 + \varepsilon)$ -factor in any direction. These summaries have been extensively studied in computational geometry [2, 9, 10, 43] 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  $\mathbb{R}^d$ ,  $\varepsilon$ -kernels of size  $O(1/\varepsilon^{(d-1)/2})$  [9, 43] can always be constructed, which is optimal. In the streaming setting, several algorithms have been developed [1, 3, 9] ultimately yielding an algorithm using  $O((1/\varepsilon^{(d-1)/2}) \log(1/\varepsilon))$  space [44].

However,  $\varepsilon$ -kernels, including those maintained by streaming algorithms, are not mergeable. Combining two  $\varepsilon$ -kernels will in general double the error or double the size.

### 1.3 Our results

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

- We warm-up by showing 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  $\varepsilon$ . 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.

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

problem	offline	streaming	mergeable
heavy hitters	$1/\varepsilon$	$1/\varepsilon$ [35, 36]	$1/\varepsilon$ (§2)
quantiles (deterministic)	$1/\varepsilon$	$(1/\varepsilon) \log(\varepsilon n)$ [21]	$(1/\varepsilon) \log u$ [39] $(1/\varepsilon) \log(\varepsilon n)$ (§3.1, restricted merging)
quantiles (randomized)	$1/\varepsilon$	$1/\varepsilon \cdot \log^{3/2}(1/\varepsilon)$ (§3.3)	
$\varepsilon$ -approximations (rectangles)	$(1/\varepsilon) \log^{2d}(1/\varepsilon)$	$(1/\varepsilon) \log^{2d+1}(1/\varepsilon)$ [40]	$(1/\varepsilon) \log^{2d+3/2}(1/\varepsilon)$ (§4)
$\varepsilon$ -approximations (range spaces) (VC-dim $\nu$ )	$1/\varepsilon^{\frac{2\nu}{\nu+1}}$	$1/\varepsilon^{\frac{2\nu}{\nu+1}} \log^{\nu+1}(1/\varepsilon)$ [40]	$1/\varepsilon^{\frac{2\nu}{\nu+1}} \log^{3/2}(1/\varepsilon)$ (§4)
$\varepsilon$ -kernels	$1/\varepsilon^{\frac{d-1}{2}}$	$1/\varepsilon^{\frac{d-1}{2}} \log(1/\varepsilon)$ [44]	$1/\varepsilon^{\frac{d-1}{2}}$ (§5, w/assumptions on data)

**Table 1: Best constructive summary size upper bounds under different models; the generality of model increases from left to right.**

- In Section 3 we first show a limited result, that the (deterministic) GK summary for  $\varepsilon$ -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  $O(\frac{1}{\varepsilon} \log^{3/2} \frac{1}{\varepsilon})$  that is mergeable. This in fact even improves on the previous best randomized streaming algorithm for quantiles, which had size  $O(\frac{1}{\varepsilon} \log^3 \frac{1}{\varepsilon})$  [40].
- In Section 4 we present mergeable  $\varepsilon$ -approximations of range spaces of near-optimal sizes. This generalizes quantile summaries (for intervals) to more general range spaces. Specifically, for  $d$ -dimensional axis-aligned rectangles, our mergeable  $\varepsilon$ -approximation has size  $O((1/\varepsilon) \log^{2d+3/2}(1/\varepsilon))$ ; for range spaces of VC-dimension  $\nu$  (e.g., ranges induced by halfspaces in  $\mathbb{R}^\nu$ ), the size is  $O(1/\varepsilon^{2\nu/(\nu+1)} \cdot \log^{3/2}(1/\varepsilon))$ .
- In Section 5 we provide a mergeable  $\varepsilon$ -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  $\varepsilon$ -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.

We summarize the current best summary sizes for these problems under various models in Table 1. The running times of our merging algorithms are polynomial (in many cases near-linear) in the summary size.

## 2. HEAVY HITTERS

The MG summary [36] and the SpaceSaving summary [35] 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. Thus they solve the frequency estimation problem with additive error  $\varepsilon n$  with space  $O(1/\varepsilon)$ , 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  $\varepsilon n$ .

We show that both MG and SpaceSaving summaries are mergeable. We first prove the mergeability of MG summaries by presenting a merging algorithm that preserves 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 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  $\hat{n}$ .  $\square$

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 [8]. Since each counter stored by the algorithm corresponds to (a subset of) actual arrivals of the corresponding item, we have that  $\hat{n} \leq n - F_1^{res(k)}$ . But we need the error bound in the lemma above in order to show mergeability.

We present an algorithm that, given two MG summaries with the property stated in Lemma 1, produces 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.  $(n_2 - \hat{n}_2)/(k+1)$ ). Our

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, say  $C_{k+1}$ , 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  $O(k)$ .

**Theorem 1** *The MG summaries are mergeable with the above merging algorithm. They have size  $O(1/\varepsilon)$ .*

PROOF. Setting  $k + 1 = \lceil 1/\varepsilon \rceil$ , the size is  $O(1/\varepsilon)$  and the claimed error is  $(n - \hat{n})/(k + 1) \leq n\varepsilon$ . That the size remains the same on a merge follows trivially from the algorithm. If we store the (items, counter) pairs in a hash table, the merging algorithm can be implemented to run in time linear in the total number of counters. So it only remains to argue that the error is preserved, i.e., the merged summary 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.

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  $(n_1 - \hat{n}_1 + n_2 - \hat{n}_2)/(k + 1)$ .

The prune operation incurs an additional error of  $C_{k+1}$ , so if we can show that

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

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 (1) follows.  $\square$

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** *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.  $\square$

Due to this correspondence, we can immediately state:

**Corollary 1** *The SpaceSaving summaries are mergeable.*

### 3. QUANTILES

We first describe a result of a weaker form of mergeability for a deterministic summary, the GK algorithm [21]. 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 [8] showed similar one-way mergeability of heavy hitter algorithms.

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  $\varepsilon$ -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  $\varepsilon$ -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  $\varepsilon n$ . 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  $\varepsilon n$ . Since  $|D \cap I|$  is the rank of  $x$  in  $D$ , we can then do a binary search<sup>3</sup> to find an  $\varepsilon$ -approximate  $\phi$ -quantile for any given  $\phi$ . We will first develop a randomized mergeable  $\varepsilon$ -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  $\varepsilon$ -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 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 wlog 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 every time insert a batch of elements, summarized by a summary in  $S_2$ . As noted in Section 1.2, prior work showed one-way mergeability of heavy hitter algorithms.

**Theorem 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  $\varepsilon n$ . 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  $\infty$ -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  $\hat{F}$ .

Given summaries  $S_1$  and  $S_2$ , which summarize  $n_1$  and  $n_2$  items respectively with error  $\varepsilon_1$  and  $\varepsilon_2$ , 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

$$\begin{aligned} & \|\hat{F}' - (F_1 + F_2)\|_\infty \\ & \leq \|\hat{F}' - (\hat{F}_2 + F_1)\|_\infty + \|(\hat{F}_2 + F_1) - (F_1 + F_2)\|_\infty \\ & \leq \varepsilon_1(n_1 + n_2) + \|\hat{F}_2 - F_2\|_\infty \\ & = \varepsilon_1(n_1 + n_2) + \varepsilon_2 n_2. \end{aligned}$$

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

By the same argument, if we merge in a third summary  $S_3$  of  $n_3$  items with error  $\varepsilon_3$ , 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  $S_1, S_2, S_3 \dots S_s$ , then the resulting summary has error at most

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

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

An immediate observation is that the GK algorithm [21] (along with other deterministic techniques for streaming computation of quantiles which require more space [32]) 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 2** *The GK algorithm is one-way mergeable, with a summary size of  $O(\frac{1}{\varepsilon} \log(\varepsilon n))$ .*

### 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 the merge-reduce algorithm [13, 33] for constructing deterministic  $\varepsilon$ -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 ( $\varepsilon$ -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.* [40], 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** *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  $k_\varepsilon$ , 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  $n/k_\varepsilon$  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  $\varepsilon$ -summary after any number of merges, so it is important that the merging algorithm is oblivious to  $m$  (hence  $n$ ). In fact, our algorithm only has one parameter  $k_\varepsilon$ . We first analyze the correctness of  $S$  for any *one* query.

**Lemma 4** *If we set  $k_\varepsilon = O((1/\varepsilon)\sqrt{\log(1/\delta)})$ , then for any interval  $I \in \mathcal{I}$  with probability at least  $1 - \delta$ ,*

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

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  $\varepsilon n$  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  $\text{abs}(X_{1,j}) \leq 1 = \Delta_1$ , by Lemma 3. 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  $\{X_{1,1}, \dots, X_{1,r_1}\}$ . 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  $2|\bigcup_j S^{(j)} \cap I|$ .

So 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  $k_\varepsilon$ , 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  $\text{abs}(X_{i,j}) \leq 2^{i-1} = \Delta_i$ , by Lemma 3. Again we have  $r_i$  such merges in this level, and  $r_i$  independent random variables  $\{X_{i,1}, \dots, X_{i,r_i}\}$ . 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  $M_{i-1}, \dots, M_1$ , 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  $\{Y_1, \dots, Y_t\}$  of independent random variables such that  $\text{abs}(Y_j - E[Y_j]) \leq \Upsilon_j$ , then for  $T = \sum_{j=1}^t Y_j$  we can bound  $\Pr[\text{abs}(T - \sum_{j=1}^t E[Y_j]) > \alpha] \leq 2e^{-2\alpha^2 / (\sum_{j=1}^t (2\Upsilon_j)^2)}$ . 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  $\text{abs}(X_{i,j} - E[X_{i,j}]) = \text{abs}(X_{i,j}) \leq \Delta_i = 2^{i-1}$ . There are  $m$  such sets for  $i \in \{1, \dots, m\}$ . Setting  $\alpha = h2^m$  for some

parameter  $h$ , we can write

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

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

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

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

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  $\varepsilon n$  error.

**Theorem 3** *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  $\varepsilon$ -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 [22].

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  $k_\varepsilon$  points. In the 0th layer, each summary point has weight 1, and in the  $i$ th layer, each summary point has weight  $2^i$ . We assume  $n/k_\varepsilon$  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  $k_\varepsilon$  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  $k_\varepsilon$  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  $\varepsilon$  across layers: Since each layer is produced by only same-weight merges, it is an  $\varepsilon$ -approximation of the set of points represented by this layer, namely the error is  $\varepsilon n_i$  for layer  $i$  where  $n_i$  is the number of points being represented. Summing over all layers yields a total error of  $\varepsilon n$ . Again it should be clear that this algorithm works without the *a priori* knowledge of the number of merges.

**Theorem 4** *There is a mergeable summary of size  $O((1/\varepsilon) \cdot \sqrt{\log(1/\varepsilon\delta)} \log(\varepsilon n))$  which is a one-dimensional  $\varepsilon$ -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 [41, 42] shows that a random sample of  $k_\varepsilon = O((1/\varepsilon^2) \log(1/\delta))$  points from  $D$  is an  $\varepsilon$ -approximation with probability  $1 - \delta$ . So an  $\varepsilon$ -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  $k_\varepsilon$  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  $k_\varepsilon$  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  $\varepsilon$ -approximation with probability at least  $1 - \delta$ .*

We next 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  $\varepsilon n$  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  $(1/\varepsilon) \text{poly}(\log(1/\varepsilon))$ .

**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  $k_\varepsilon = (4/\varepsilon) \sqrt{\ln(4/\varepsilon)}$ . Let  $g_\varepsilon = (64/\varepsilon^2) \ln(16/\varepsilon)$ . For each level  $l$  between  $i = j - \log_2(g_\varepsilon)$  and  $j - 1$  we either maintain  $k_\varepsilon$  points, or no points. Each point at the  $l$ th 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  $k_\varepsilon$  points (or only  $m$  if  $m < k_\varepsilon$ ). Each point in the sample has weight  $m/k_\varepsilon$  (or 1 if  $m < k_\varepsilon$ ). Note the total size is  $O(k_\varepsilon \log(g_\varepsilon)) = O((1/\varepsilon) \log^{1.5}(1/\varepsilon))$ .

**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  $k_\varepsilon$  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  $k_\varepsilon$  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  $k_\varepsilon$  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.

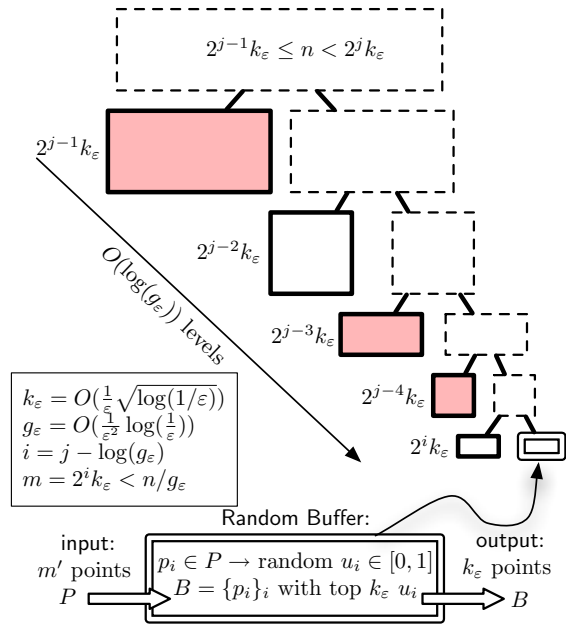
**Analysis.** First we formalize the upward movement of points.

**Lemma 5** *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  $l \leq i - 1$ , 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.





**Figure 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 only  $k_\epsilon$  summary points.

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.  $\square$

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  $\text{abs}(X) \leq \epsilon n/2$  with probability  $1 - \epsilon/4$ . Then applying a union bound will ensure the summary is correct for all  $1/\epsilon$  intervals in  $\mathcal{I}_\epsilon$  with probability at least  $3/4$ . This will imply that for all intervals  $I \in \mathcal{I}$  the summary has error at most  $\epsilon n$ .

The total over-count error can be decomposed into two parts. First, we invoke Theorem 4 to show that the effect of all same-weight merges has error at most  $\epsilon n/4$  with probability at least  $1 - \epsilon/8$ . 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 4, 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  $\epsilon n/4$  with probability at least  $1 - \epsilon/8$ . 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  $\epsilon n/2$  with probability at least  $1 - \epsilon/4$ .

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  $k_\epsilon$  points of weight  $2^l$  each. Then we quantify the total error attributed to the random buffer output at level  $l$ .

**Lemma 6** *A summary of size  $n$ , for  $2^{j-1}k_\epsilon \leq n < 2^j k_\epsilon$ , has experienced  $h_l \leq 2^{j-l} = 2^{i-l}g_\epsilon$  random buffer promotions to level  $l$  within its entire merge history.*

**PROOF.** By Lemma 5, 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  $n < 2^j k_\epsilon$ . Since each promotion outputs points with a total weight of  $2^l k_\epsilon$ , this can happen at most  $h_l < 2^j k_\epsilon / 2^l k_\epsilon = 2^{j-l}$  times. The proof concludes using  $g_\epsilon = 2^{j-i}$ .  $\square$

**Lemma 7** *When the random buffer promotes a set  $B$  of  $k_\epsilon$  points representing a set  $P$  of  $m'$  points (where  $m/2 < m' < 2m$ ), for any interval  $I \in \mathcal{I}$  the over-count*

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

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

**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  $|B| = k_\epsilon$ , then  $m/k_\epsilon$  is the correct scaling constant in expectation.

To bound  $\text{abs}(X)$ , 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  $\text{abs}(X) < 2m$ .  $\square$

Since  $m \leq n/g_\epsilon$  at level  $i$ , then  $m \leq 2^{l-i}n/g_\epsilon$  at level  $l$ , and we can bound the over-count error as  $\Delta_l = \text{abs}(X) \leq 2m \leq 2^{l-i+1}n/g_\epsilon$ . 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  $\text{abs}(X_{l,s}) \leq \Delta_l$ . These events are independent so we can apply another Chernoff-Hoeffding bound on these  $\sum_{l=0}^i h_l$  events. Recall that  $g_\epsilon = (64/\epsilon^2) \ln(16/\epsilon)$  and let  $\hat{T} = \sum_{i=0}^i \sum_{s=1}^{h_i} X_{l,s}$ , which has expected value 0. Then

$$\begin{aligned} \Pr[\text{abs}(\hat{T}) \geq \epsilon n/4] &= 2 \exp\left(-2 \frac{(\epsilon n/4)^2}{\sum_{i=0}^i h_i \Delta_i^2}\right) \\ &\leq 2 \exp\left(-2 \frac{(\epsilon n/4)^2}{\sum_{i=0}^i (2^{i-l}g_\epsilon)^2 (2^{l-i+1}n/g_\epsilon)^2}\right) \\ &\leq 2 \exp\left(-g_\epsilon \frac{\epsilon^2}{8} \frac{1}{\sum_{l=0}^i 2^{i-l}2^{2(l-i)+2}}\right) \end{aligned}$$

$$\begin{aligned}
&= 2 \exp \left( -g_\varepsilon \frac{\varepsilon^2}{32} \frac{1}{\sum_{l=0}^i 2^{l-i}} \right) \\
&= 2 \exp \left( -2 \ln(16/\varepsilon) \frac{1}{\sum_{l=0}^i 2^{-l}} \right) \\
&\leq 2 \exp(-\ln(16/\varepsilon)) \\
&= 2(\varepsilon/16) = \varepsilon/8.
\end{aligned}$$

**Theorem 5** *The above scheme maintains a fully mergeable one-dimensional  $\varepsilon$ -approximation of size  $O(\frac{1}{\varepsilon} \log^{1.5}(1/\varepsilon))$ , with probability at least  $3/4$ .*

#### 4. $\varepsilon$ -APPROXIMATIONS OF RANGE SPACES

In this section, we generalize the approach of the previous section to  $\varepsilon$ -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  $\nu$  (see Section 1.2 for the definition). We will 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 [5], we compute a low-discrepancy coloring  $\chi : S' \rightarrow \{-1, +1\}$  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 as follows.

**Lemma 8** *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 [38]. Hence the generalization of Lemma 3 is as follows.

**Lemma 9** *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 4 and its proof generalize in a straightforward way, the only change being that now  $\Delta_i = 2^{i-1} \Lambda_\nu$ , hence implying  $h > O(\Lambda_\nu \log^{1/2}(1/\varepsilon\delta))$  and

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

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

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

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

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

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

**Lemma 10** *Using the above framework, on same-weight merges, we can maintain an  $\varepsilon$ -approximation of  $(P, \mathcal{A})$  with constant VC-dimension  $\nu$  of size  $O((1/\varepsilon)^{2\nu/(\nu+1)} \log^{\nu/(\nu+1)}(1/\varepsilon\delta))$ , with probability at least  $1 - \delta$ . For the range space  $(P, \mathcal{R}_d)$ , the size is  $O((1/\varepsilon) \log^{2d}(\log(1/\delta)/\varepsilon) \cdot \sqrt{\log(1/\varepsilon\delta)})$ .*

Then this extends to different-weight merges with an extra  $\log(n\varepsilon)$  factor, as with intervals. Also, the random buffer can maintain a random sample of the same asymptotic size  $O((1/\varepsilon^2) \log(1/\delta))$  and 0 expected over-count error. Substituting the increased  $k_\varepsilon$  value, the generalizations of Lemma 6 and Lemma 7 follow. We can also move the  $\log(1/\delta)$  term to the outside, by setting  $\delta = 1/2$  in these expressions, and then repeating the processes  $O(\log(1/\delta'))$  times independently to drive the probability of failure down to  $\delta'$ .

**Theorem 6** *A mergeable  $\varepsilon$ -approximation of a range space  $(D, \mathcal{R})$  of VC-dimension  $\nu$  of size  $O(1/\varepsilon^{\frac{2\nu}{\nu+1}} \log^{\frac{2\nu+1}{\nu+1}} \frac{1}{\varepsilon} \log \frac{1}{\delta})$  can be maintained with probability at least  $1 - \delta$ . If  $\mathcal{R} = \mathcal{R}_d$ , then the size is  $O\left(\frac{1}{\varepsilon} \log^{2d+3/2} \left(\frac{1}{\varepsilon}\right) \log \frac{1}{\delta}\right)$ .*

#### 5. $\varepsilon$ -KERNELS

A unit vector  $u$  in  $\mathbb{R}^d$  defines a direction, 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  $E[P, u] = \max_{p \in P} \langle u, p \rangle$ , and the width  $\text{wid}[P, u] = E[P, u] + E[P, -u]$ . An  $\varepsilon$ -kernel is a subset  $K \subset P$  such that over all directions  $u$ ,

$$\max_u E[P, u] - E[K, u] \leq \varepsilon \text{wid}[P, u]$$

and

$$\max_u \frac{\text{wid}[P, u] - \text{wid}[K, u]}{\text{wid}[P, u]} \leq 2\varepsilon.$$

An  $\varepsilon$ -kernel of size  $O(1/\varepsilon^{(d-1)/2})$  can be computed in time  $O(|P| + 1/\varepsilon^{d-3/2})$  [9, 43]. It is also known that the union of two  $\varepsilon$ -kernels is an  $\varepsilon$ -kernel [9], but this observation alone is not sufficient to have mergeable  $\varepsilon$ -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  $\varepsilon$ -kernel basics.** We say a point set  $P$  is  $\beta$ -fat if over all directions  $u, v$  we can bound the width ratio  $\max_{u,v} (\text{wid}(P, u)/\text{wid}(P, v)) \leq \beta$ . Given a box  $B \supset P$ ,  $P$  is  $\beta$ -fat with respect to  $B$  if

$$\max_{u,v} (\text{wid}(B, u)/\text{wid}(B, v)) \leq \beta.$$

If  $\beta$  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 axis, as well as a relative scale along those axis. That is, the  $d$  orthogonal directions the of box's face normals  $\{b_1, \dots, b_d\}$  define coordinate axis, 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  $\varepsilon$ -kernels use the following observations.

- Let  $A$  be an affine transform. If  $K$  is an  $\varepsilon$ -kernel of  $P$ , then  $A(K)$  is an  $\varepsilon$ -kernel of  $A(P)$  [1].
- Let  $I = [-1, 1]^d$  and  $\beta_d = 2^d d^{5/2} d!$ . There exists an  $O(d^2|P|)$  size algorithm to construct an affine transform  $A$  such that  $A(P) \subset I$  and  $A(P)$  is  $\beta_d$ -fat with respect to  $I$  [7, 25].
- Place a grid  $G_\varepsilon$  on  $I$  so that each grid cell has width  $\varepsilon/\beta_d$ . 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  $\varepsilon$ -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  $O(1/\varepsilon^{d-1})$  [1].

Because of the first two observations, for static  $\varepsilon$ -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  $\varepsilon$ -kernel algorithms is maintaining such a reference frame  $I$ .

The size of  $\varepsilon$ -kernels can be reduced to  $O(1/\varepsilon^{(d-1)/2})$  using an additional trick [1], given that we have an  $(\varepsilon/3)$ -kernel  $K_1$  of  $P$  that is  $\beta$ -fat with respect to  $I$ . Consider a sphere  $S$  of radius  $\sqrt{d} + 1$  centered at the origin, and place a set  $Q_\varepsilon$  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  $\varepsilon$ -kernel of  $P$  (of size  $O(1/\varepsilon^{(d-1)/2})$ ).

#### Mergeable $\varepsilon$ -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 [1] implies that we can assume  $B = [-1, +1]^d$ . We can now create mergeable  $\varepsilon$ -kernels for  $P_1$  and  $P_2$ . More precisely, we create  $\varepsilon$ -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  $\varepsilon$ -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  $\varepsilon$ -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  $\varepsilon$ -kernels as desired, without increasing the  $\varepsilon$  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 7** *A mergeable  $\varepsilon$ -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  $\varepsilon$ -kernel that does not restrict the points to a fixed reference frame.

## 6. CONCLUDING REMARKS

We have formalized the notion of mergeable summaries, and demonstrated fully mergeable summaries for the cen-

tral problems of heavy hitters, quantiles,  $\varepsilon$ -approximations and  $\varepsilon$ -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 the problem of  $k$ -median clustering. Guha *et al.* [24] 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  $O(\frac{k}{\varepsilon} \log 1/\varepsilon)$  that provides a  $2 + \varepsilon$  guarantee [23].

In the graph setting, a simple technique for finding a  $t$ -spanner is to eject any edges which complete a cycle of length  $t$ . Given two such spanners, we can merge by simply applying the same rule as we add one graph to the other, edge by edge. This results in a mergeable summary of size  $O(n^{1+2/(t+1)})$  [16]. 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.

## 7. REFERENCES

- [1] P. K. Agarwal, S. Har-Peled, and K. R. Varadarajan. Approximating extent measure of points. *Journal of the ACM*, 51(4):660–635, 2004.
- [2] P. K. Agarwal, J. M. Phillips, and H. Yu. Stability of  $\varepsilon$ -kernels. In *Proc. European Symposium on Algorithms*, 2010.
- [3] P. K. Agarwal and H. Yu. A space-optimal data-stream algorithm for coresets in the plane. In *Proc. Annual Symposium on Computational Geometry*, 2007.
- [4] N. Alon, Y. Matias, and M. Szegedy. The space complexity of approximating the frequency moments. *Journal of Computer and System Sciences*, 58(1):137–147, 1999.
- [5] N. Bansal. Constructive algorithms for discrepancy minimization. In *Proc. IEEE Symposium on Foundations of Computer Science*, pages 407–414, 2010.
- [6] Z. Bar-Yossef, T. S. Jayram, R. Kumar, D. Sivakumar, and L. Trevisan. Counting distinct elements in a data stream. In *RANDOM*, 2002.
- [7] G. Barequet and S. Har-Peled. Efficiently approximating the minimum-volume bounding box of a point set in three dimensions. *Journal of Algorithms*, 38:91–109, 2001.
- [8] R. Berinde, G. Cormode, P. Indyk, and M. Strauss. Space-optimal heavy hitters with strong error bounds. *ACM Transactions on Database Systems*, 35(4), 2010.
- [9] T. Chan. Faster core-set constructions and data-stream algorithms in fixed dimensions. *Computational Geometry: Theory and Applications*, 35:20–35, 2006.
- [10] T. Chan. Dynamic coresets. In *Proc. Annual Symposium on Computational Geometry*, 2008.
- [11] M. Charikar, A. Newman, and A. Nikolov. Tight

- hardness results for minimizing discrepancy. In *Proc. ACM-SIAM Symposium on Discrete Algorithms*, 2011.
- [12] B. Chazelle. *The Discrepancy Method*. Cambridge, 2000.
- [13] B. Chazelle and J. Matousek. On linear-time deterministic algorithms for optimization problems in fixed dimensions. *Journal of Algorithms*, 21:579–597, 1996.
- [14] G. Cormode and M. Hadjieleftheriou. Finding frequent items in data streams. *Proc. VLDB Endowment*, 1(2):1530–1541, 2008.
- [15] G. Cormode and S. Muthukrishnan. An improved data stream summary: The count-min sketch and its applications. *Journal of Algorithms*, 55(1):58–75, 2005.
- [16] J. Feigenbaum, S. Kannan, A. McGregor, S. Suri, and J. Zhang. Graph distances in the streaming model: The value of space. In *ACM-SIAM Symposium on Discrete Algorithms*, 2005.
- [17] J. Feigenbaum, S. Kannan, M. J. Strauss, and M. Viswanathan. An approximate L1-difference algorithm for massive data streams. *SIAM Journal on Computing*, 32(1):131–151, 2003.
- [18] J. Feldman, S. Muthukrishnan, A. Sidiropoulos, C. Stein, and Z. Svitkina. On distributing symmetric streaming computations. In *Proc. ACM-SIAM Symposium on Discrete Algorithms*, 2008.
- [19] S. Ganguly and A. Majumder. CR-precis: A deterministic summary structure for update data streams. In *ESCAPE*, 2007.
- [20] A. C. Gilbert, Y. Kotidis, S. Muthukrishnan, and M. J. Strauss. How to summarize the universe: Dynamic maintenance of quantiles. In *Proc. International Conference on Very Large Data Bases*, 2002.
- [21] M. Greenwald and S. Khanna. Space-efficient online computation of quantile summaries. In *Proc. ACM SIGMOD International Conference on Management of Data*, 2001.
- [22] M. Greenwald and S. Khanna. Power conserving computation of order-statistics over sensor networks. In *Proc. ACM Symposium on Principles of Database Systems*, 2004.
- [23] S. Guha. Tight results for clustering and summarizing data streams. In *Proc. International Conference on Database Theory*, 2009.
- [24] S. Guha, N. Mishra, R. Motwani, and L. O’Callaghan. Clustering data streams. In *Proc. IEEE Conference on Foundations of Computer Science*, 2000.
- [25] S. Har-Peled. *Approximation Algorithm in Geometry (Chapter 21)*. <http://valis.cs.uiuc.edu/~sariel/teach/notes/aprx/>, 2010.
- [26] D. M. Kane, J. Nelson, and D. P. Woodruff. An optimal algorithm for the distinct elements problem. In *Proc. ACM Symposium on Principles of Database Systems*, 2010.
- [27] K. G. Larsen. On range searching in the group model and combinatorial discrepancy. under submission, 2011.
- [28] Y. Li, P. M. Long, and A. Srinivasan. Improved bounds on the sample complexity of learning. *Journal of Computer and System Sciences*, 62:516–527, 2001.
- [29] S. Madden, M. J. Franklin, J. M. Hellerstein, and W. Hong. TAG: a tiny aggregation service for ad-hoc sensor networks. In *Proc. Symposium on Operating Systems Design and Implementation*, 2002.
- [30] A. Manjhi, S. Nath, and P. B. Gibbons. Tributaries and deltas: efficient and robust aggregation in sensor network streams. In *Proc. ACM SIGMOD International Conference on Management of Data*, 2005.
- [31] A. Manjhi, V. Shkapenyuk, K. Dhamdhere, and C. Olston. Finding (recently) frequent items in distributed data streams. In *Proc. IEEE International Conference on Data Engineering*, 2005.
- [32] G. S. Manku, S. Rajagopalan, and B. G. Lindsay. Approximate medians and other quantiles in one pass and with limited memory. In *Proc. ACM SIGMOD International Conference on Management of Data*, 1998.
- [33] J. Matousek. Approximations and optimal geometric divide-and-conquer. In *Proc. ACM Symposium on Theory of Computing*, 1991.
- [34] J. Matousek. *Geometric Discrepancy; An Illustrated Guide*. Springer, 1999.
- [35] A. Metwally, D. Agrawal, and A. Abbadi. An integrated efficient solution for computing frequent and top-k elements in data streams. *ACM Transactions on Database Systems*, 31(3):1095–1133, 2006.
- [36] J. Misra and D. Gries. Finding repeated elements. *Science of Computer Programming*, 2:143–152, 1982.
- [37] J. Nelson and D. P. Woodruff. Fast manhattan sketches in data streams. In *Proc. ACM Symposium on Principles of Database Systems*, 2010.
- [38] J. M. Phillips. Algorithms for  $\epsilon$ -approximations of terrains. In *Proc. ICALP*, 2008.
- [39] N. Shrivastava, C. Buragohain, D. Agrawal, and S. Suri. Medians and beyond: New aggregation techniques for sensor networks. In *Proc. ACM SenSys*, 2004.
- [40] S. Suri, C. Toth, and Y. Zhou. Range counting over multidimensional data streams. *Discrete and Computational Geometry*, 36(4):633–655, 2006.
- [41] M. Talagrand. Sharper bounds for Gaussian and empirical processes. *Annals of Probability*, 22:76, 1994.
- [42] V. Vapnik and A. Chervonenkis. On the uniform convergence of relative frequencies of events to their probabilities. *Theory of Probability and its Applications*, 16:264–280, 1971.
- [43] H. Yu, P. K. Agarwal, R. Poredy, and K. R. Varadarajan. Practical methods for shape fitting and kinetic data structures using coresets. In *Proc. Annual Symposium on Computational Geometry*, 2004.
- [44] H. Zarrabi-Zadeh. An almost space-optimal streaming algorithm for coresets in fixed dimensions. In *Proc. European Symposium on Algorithms*, 2008.