UCLA

UCLA Electronic Theses and Dissertations

Title

Order statistics and variability in data streams

Permalink

https://escholarship.org/uc/item/5nd2399r

Author

Felber, David

Publication Date

2015

Peer reviewed|Thesis/dissertation

eScholarship.org Powered by the California Digital Library
University of California

University of California

Los Angeles

Order statistics and variability in data streams

A dissertation submitted in partial satisfaction

of the requirements for the degree

Doctor of Philosophy in Computer Science

by

David Victor Felber

2015

© Copyright by

David Victor Felber

2015

Abstract of the Dissertation

Order statistics and variability in data streams

by

David Victor Felber

Doctor of Philosophy in Computer Science

University of California, Los Angeles, 2015

Professor Rafail Ostrovsky, Chair

High-volume data streams are too large and grow too quickly to store entirely in working memory, introducing new challenges to what might otherwise be simple calculations. For one, nearly all interesting statistics of a stream can only be estimated without recalling expired data, which can be costly or impossible. For another, even just estimating a statistic could be infeasible if the space required to do so does not grow slowly with the size of the stream.

One such set of fundamental statistics are the quantiles (order statistics) of a dataset defined by an aggregating stream of items. We develop an approximate quantile summary with probabilistic guarantees that yields a new upper bound on the space needed for such a summary when the items are subject only to comparison operations.

The difficulties of data streams are compounded when streams may arrive distributed across several locations, as would occur in sensor networks and telecommunication networks; in addition to handling the high volume of these streams we must also coordinate among the sites to ensure consistency for any statistics we wish to track, ideally with a minimum of communication.

We introduce a new, natural parameter for data streams, the "variability" v, that permits us to easily extend existing algorithms for the aggregating streaming model to one in which streams are composed of insertion and deletion transactions. For this second model, our definition refines existing worst-case communication bounds from $O(n)$ to $\tilde{O}(v)$ for a host of problems. We further show that the variabilities for many streams of interest grow slowly with respect to the size of the streams.

The dissertation of David Victor Felber is approved.

Suhas Diggavi

Raghu Meka

Alexander Sherstov

Rafail Ostrovsky, Committee Chair

University of California, Los Angeles

2015

Table of Contents

List of Figures

Acknowledgments

I offer my sincere thanks and appreciation to Rafail Ostrovsky, my advisor. Rafi was always willing to make time to speak with me, to point out interesting problems and directions for research, to share his thoughts and ideas, and to discuss and listen to mine. Anything that I've accomplished here, would not have been, without his guidance.

I'm grateful also to Suhas Diggavi, Raghu Meka, and Alexander Sherstov, for serving on my dissertation committee; to Adam Meyerson, for introducing me to computer science theory research; and to Aaron Coté, Ran Gelles, Yuan He, Alan Roytman, Michael Shindler, Brian Tagiku, Corey Worrell, and Arman Yousefi, past and present colleagues, who contributed to making my time as a graduate student pleasant and enjoyable.

This dissertation is based on co-authored work. Chapter 1 is a version of [FO15a]. Chapter 2 is a version of [FO15b]. Versions of both have been submitted to conferences for publication.

On a personal note, I thank my family for supporting me in all my endeavors.

Vita

2008	B.S. (Computer Science) and B.S. (Mathematics), University of California, Los Angeles.
2010	M.S. (Computer Science), University of California, Los Angeles.
2011–2015	Teaching Assistant, University of California, Los Angeles.

CHAPTER 1

A randomized quantile summary

A quantile summary is a data structure that approximates to ε-relative error the order statistics of a much larger underlying dataset.

In this chapter we develop a randomized online quantile summary for the cash register data input model and comparison data domain model that uses $O(\frac{1}{\varepsilon}\log\frac{1}{\varepsilon})$ words of memory. This improves upon a previous upper bound of $O(\frac{1}{\varepsilon}\log^{3/2}\frac{1}{\varepsilon})$ by Agarwal et. al. (PODS 2012). Further, by a lower bound of Hung and Ting (FAW 2010) no deterministic summary for the comparison model can outperform our randomized summary in terms of space complexity. Lastly, our summary has the nice property that $O(\frac{1}{\varepsilon}\log\frac{1}{\varepsilon})$ words suffice to ensure that the success probability is $1 - e^{-\text{poly}(1/\varepsilon)}$.

1.1 Introduction

A quantile summary S is a fundamental data structure that summarizes an underlying dataset X of size n, in space much less than n. Given a query ϕ, S returns a sample y of X such that the rank of y in X is (probably) approximately ϕn. Quantile summaries are used in sensor networks to aggregate data in an energy-efficient manner and in database query optimizers to generate query execution plans.

Quantile summaries have been developed for a variety of different models and metrics. The data input model we consider is the standard online cash register streaming model, in which a new item is added to the dataset at each new timestep, and the total number of items is not known until the end. The data domain model we consider is the comparison model, in which stream items come from an arbitrary ordered domain (and specifically, not necessarily from the integers).

Formally, our quantile summary problem is defined over a totally ordered domain $\mathcal{D}$ and by an error parameter $\varepsilon \leq 1/2$. There is a dataset X that is initially empty. Time occurs in discrete steps. In timestep t, stream item x_t arrives and is then processed, and then any quantile queries ϕ in that step are received and processed. To be definite, we pick the first timestep to be 1. We write X_t or $X(t)$ for the t-item prefix stream $x_1 \ldots x_t$ of X. The goal is to maintain at all times t a summary S_t of the dataset X_t that, given any query ϕ in $(0, 1]$, can return a sample $y = y(\phi)$ so that $|R(y, X_t) - \phi t| \leq \varepsilon t$, where $R(a, Z)$ is the *rank of item a in set Z*, defined as $|\{z \in Z : a \leq z\}|$. For randomized summaries, we only require that $\forall t \forall \phi,\ P(|R(y, X_t) - \phi t| \leq \varepsilon t) \geq 2/3$; that is, y's rank is only probably close to ϕt, not definitely close. In fact, it will be easier to deal with the rank directly, so we define $\rho = \phi t$ and use that in what follows.

1.1.1 Previous work

The two most directly relevant pieces of prior work ([ACH12, ACH13] and [MRL99]) are randomized online quantile summaries for the cash register/comparison model. Aside from oblivious sampling algorithms (which require storing $\Omega(1/\varepsilon^2)$ samples) the only other such work of which we are aware is an approach by Wang, Luo, Yi, and Cormode [WLY13] that combines the methods of [ACH12, ACH13] and [MRL99] into a hybrid with the same space bound as [ACH12, ACH13].

The newer of the two is that of Agarwal, Cormode, Huang, Phillips, Wei, and Yi [ACH12, ACH13]. Among other results, Agarwal et. al. develop a randomized online quantile summary for the cash register/comparison model that uses $O(\frac{1}{\varepsilon}\log^{3/2}\frac{1}{\varepsilon})$ words of memory. This summary has the nice property that any two such summaries can be combined to form a summary of the combined underlying dataset without loss of accuracy or increase in size.

The earlier such summary is that of Manku, Rajagopalan, and Lindsay [MRL99], which uses $O(\frac{1}{\varepsilon}\log^2\frac{1}{\varepsilon})$ space. At a high level, their algorithm downsamples the input stream in a non-uniform way and feeds the downsampled stream into a deterministic summary, while periodically adjusting the downsampling rate.

For the comparison model, the best deterministic online summary to date is the (GK) summary of Greenwald and Khanna [GK01], which uses $O(\frac{1}{\varepsilon}\log\varepsilon n)$ space. This improved upon a deterministic (MRL) summary of Manku, Rajagopalan, and Lindsay [MRL98] and a summary implied by Munro and Paterson [MP78], which use $O(\frac{1}{\varepsilon}\log^2\varepsilon n)$ space.

A more restrictive domain model than the comparison model is the bounded universe model, in which elements are drawn from the integers $\{1,\ldots,u\}$. For this model there is a deterministic online summary by Shrivastava, Buragohain, Agrawal, and Suri [SBA04] that uses $O(\frac{\log u}{\varepsilon})$ space.

Not much exists in the way of lower bounds for this problem. There is a simple lower bound of $\Omega(1/\varepsilon)$ which intuitively comes from the fact that no one sample can satisfy more than $2\varepsilon n$ different rank queries. For the comparison model, Hung and Ting [HT10] developed a deterministic $\Omega(\frac{1}{\varepsilon}\log\frac{1}{\varepsilon})$ lower bound.

1.1.2 Our results

In the next section we describe a simple $O(\frac{1}{\varepsilon}\log\frac{1}{\varepsilon})$ streaming summary that is online except that it requires n to be given up front and that it is unable to process queries until it has seen a constant fraction of the input stream. In section 1.3 we develop this simple summary into a fully online summary that can answer queries at any point in time. We close in section 1.4 by examining the similarities and differences between our summary and previous work and discuss a design approach for similar streaming problems.

1.2 A simple streaming summary

Before we describe our algorithm we must first describe its two main components in a bit more detail than was used in the chapter introduction. The two components are Bernoulli sampling and the GK summary [GK01].

1.2.1 Bernoulli sampling

Bernoulli sampling downsamples a stream X of size n to a sample stream S by choosing to include each next item into S with independent probability m/n. (As stated this requires knowing the size of X in advance.) At the end of processing X, the expected size of S is m, and the expected rank of any sample y in S is $E(R(y,S)) = \frac{m}{n}R(y,X)$. In fact, for any times $t \leq n$ and partial streams X_t and S_t, where S_t is the sample stream of X_t, we have $E(|S_t|) = mt/n$ and $E(R(y,S_t)) = \frac{m}{n}R(y,X_t)$. To generate an estimate for $R(y,X_t)$ from S_t we use $\hat{R}(y,X_t) = \frac{n}{m}R(y,S_t)$. The following theorem bounds the probability that S is very large or that $\hat{R}(y,X_t)$ is very far from $R(y,X_t)$ (for any given time $t \geq n/64$, but not for all times $t = n/64 \ldots n$ combined). Wang et. al. [WLY13] credit the result to Vapnik and Chervonenkis [VC71].

Theorem 1.2.1. *For all times $t \geq n/64$, $P(|S_t| > 2tm/n) < \exp(-m/192)$.*

Further, for all times $t \geq n/64$ and items y,

$$P(|\hat{R}(y,X_t) - R(y,X_t)| > \varepsilon t/8) < 2\exp(-\varepsilon^2 m/12288)$$

Proof. For the first part, $P(|S_t| > 2tm/n) < \exp(-tm/3n) < \exp(-m/192)$

(since $t \geq n/64$).

For the second part,

$$\begin{aligned} & P(|\hat{R}(y, X_t) - R(y, X_t)| > \varepsilon t/8) \\ = \; & P(|R(y, S_t) - E(R(y, S_t))| > \varepsilon t m/8n) \end{aligned}$$

The Chernoff bound is

$$P(|R(y, S_t) - E(R(y, S_t))| > \delta E(R(y, S_t))) \; < \; 2\exp(-\min\{\delta, \delta^2\} E(R(y, S_t))/3)$$

Here, $\delta = \varepsilon t/8R(y, S_t)$, so

$$P \; < \; 2\exp(-\varepsilon^2 t^2 m/192nE(R(y, S_t))) \; \leq \; 2\exp(-\varepsilon^2 m/12288)$$

yielding the theorem. □

This means that, given any $1 \leq \rho \leq t$, if we return the sample $y \in S_t$ with $R(y, S_t) = \rho m/n$, then $R(y, X_t)$ is likely to be close to ρ.

1.2.2 GK summary

The GK summary is a deterministic summary that can answer queries to relative error, over any portion of the received stream. If G_t is the summary after inserting the first t items X_t from stream X into G then, given any $1 \leq \rho \leq t$, G_t can return a sample $y \in X_t$ so that $|R(y, X_t) - \rho| \leq \varepsilon t/8$. Greenwald and Khanna guarantee in [GK01] that G_t uses $O(\frac{1}{\varepsilon}\log(\varepsilon t))$ words. We call this the *GK guarantee.*

1.2.3 Our summary

We combine Bernoulli sampling with the GK summary by downsampling the input data stream X to a sample stream S and then feeding S into a GK summary G.

It looks like this:

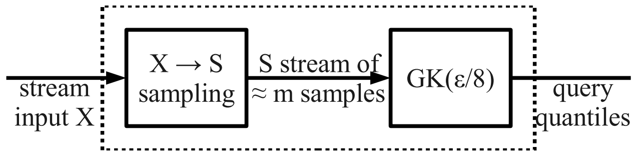

Figure 1.1: The big picture.

The key reason this gives us a small summary is that we never need to store S; each time we sample an item into S we immediately feed it into G. Therefore, we only use as much space as $G(S(X_t))$ uses. In particular, as long as we have $m = O(\text{poly}(1/\varepsilon))$, we use only $O(\frac{1}{\varepsilon}\log\frac{1}{\varepsilon})$ words.

To answer a query ρ for X_t we ask G_t the query $\rho m/n$ and return the resulting sample y. There is a slight issue in that $\rho m/n$ may be larger than $|S|$; but if the approximation guarantee holds for the largest item in X_t then we have $\rho m/n < (t+\varepsilon t/8)m/n$, so using $\min\{\rho m/n, |S|\}$ instead will not cause more than $\varepsilon/8$ relative error in the approximation.

The probability that our sample stream S_t is not too big (uses more than $2tm/n$ samples) is at least $1 - \exp(-m/192)$. If this happens to be the case then the probability that all of its samples y have $|R(y, S_t) - E(R(y, S_t))| \leq \varepsilon tm/8n$ (that is, are good) is at least $1 - 4m\exp(-\varepsilon^2 m/12288)$ by theorem 1.2.1 and the union bound. Choosing $m \geq \frac{300000 \ln 1/\varepsilon}{\varepsilon^2}$ suffices to guarantee that both events occur with total probability at least $2/3$.

Further, if both S_t events occur then the total error introduced by both S_t and G_t is at most $\varepsilon t/2$. Suppose that G_t returns y when given $\rho m/n$. This means that $|R(y, S_t) - \rho m/n| \leq \varepsilon|S_t| \leq \varepsilon(2tm/n)/8$ by the GK guarantee. Since both events for S_t occur, we also have $|R(y, S_t) - \frac{m}{n}R(y, X_t)| \leq \varepsilon tm/4n$ (and only $\varepsilon tm/8n$ in the case that we don't truncate $\rho m/n$ to $|S|$). Thus, $|\frac{m}{n}R(y, X_t) - \rho m/n| \leq \varepsilon tm/2n$. Equivalently, $|R(y, X_t) - \rho| \leq \varepsilon t/2$.

1.2.4 Caveats

There are two serious issues with this summary. The first is that it requires us to know the value of n in advance to perform the sampling. Also, as a byproduct of the sampling, we can only obtain approximation guarantees after we have seen at least 1/64 (or at least some constant fraction) of the items. This means that while the algorithm is sufficient for approximating order statistics over streams stored on disk, more is needed to get it to work for online streaming applications, in which (1) the stream size n is not known in advance, and (2) queries can be answered approximately at all times $t \leq n$ and not just when $t \geq n/64$.

An important point to note is that the GK summary only works in the cash register data input model; it does not permit items to be deleted from the stream. A naive approach to avoiding the aforementioned problems would be to restrict the number of items sampled using a method like reservoir sampling [Vit85]. Aside from the issue of implementing such a sampling method in $O(\frac{1}{\varepsilon} \log \frac{1}{\varepsilon})$ words, a major problem with this approach is that it effectively deletes items from the sample stream that is fed into the GK summary. So to use our basic streaming summary idea of figure 1.1 online really requires an approach that permits committing to sampled items.

Thus, adapting our basic streaming summary idea to work online constitutes the next section and the bulk of this chapter. We start with a high-level overview of our online summary algorithm. In section 1.3.1 we formally define an initial version of our algorithm whose expected size at any given time (but not necessarily at all times) is $O(\frac{1}{\varepsilon} \log \frac{1}{\varepsilon})$ words. In section 1.3.2 we show that our algorithm gurantees that $\forall n \forall \rho,\ P(|R(y, X_n) - \rho| \leq \varepsilon n) \geq 1 - \exp(-1/\varepsilon)$. In section 1.3.3 we discuss the slight modifications necessary to get a deterministic $O(\frac{1}{\varepsilon} \log \frac{1}{\varepsilon})$ space complexity, and also perform a time complexity analysis.

1.3 An online summary

Our algorithm works in *rows*, which are illustrated in figure 1.2. Row r is a summary of the first $2^r 32m$ stream items. Since we don't know how many items will actually be in the stream, we can't start all of these rows running at the outset. Therefore, we start each row $r \geq 1$ once we have seen $1/64$ of its total items. However, since we can't save these items for every row we start, we need to construct an approximation of this fraction of the stream, which we do by using the summary of the previous row, and join this approximating stream with the new items that arrive while the row is live. We then wait until the row has seen a full half of its items before we permit it to start answering queries; this dilutes the influence of approximating the $1/64$ of its input that we couldn't store.

Operation within a row is very much like the operation of our fixed-n streaming summary. We feed the joint approximate prefix + new item stream through a Bernoulli sampler to get a sample stream, which is then fed into a GK summary (which is stored). After row r has seen half of its items, its GK summary becomes the one used to answer quantile queries. When row $r+1$ has seen $1/64$ of *its* total items, row r generates an approximation of those items from its GK summary and feeds them as a stream into row $r+1$.

Row 0 is slightly different in order to bootstrap the algorithm. There is no join step since there is no previous row to join. Also, row 0 is active from the start. Lastly, we get rid of the sampling step so that we can answer queries over timesteps $1 \dots m/2$.

After the first $32m$ items, row 0 is no longer needed, so we can clean up the space used by its GK summary. Similarly, after the first $2^r 32m$ items, row r is no longer needed. The upshot of this is that we never need storage for more than six rows at a time. Since each GK summary uses $O(\frac{1}{\varepsilon} \log \frac{1}{\varepsilon})$ words, the six live GK summaries use only a constant factor more.

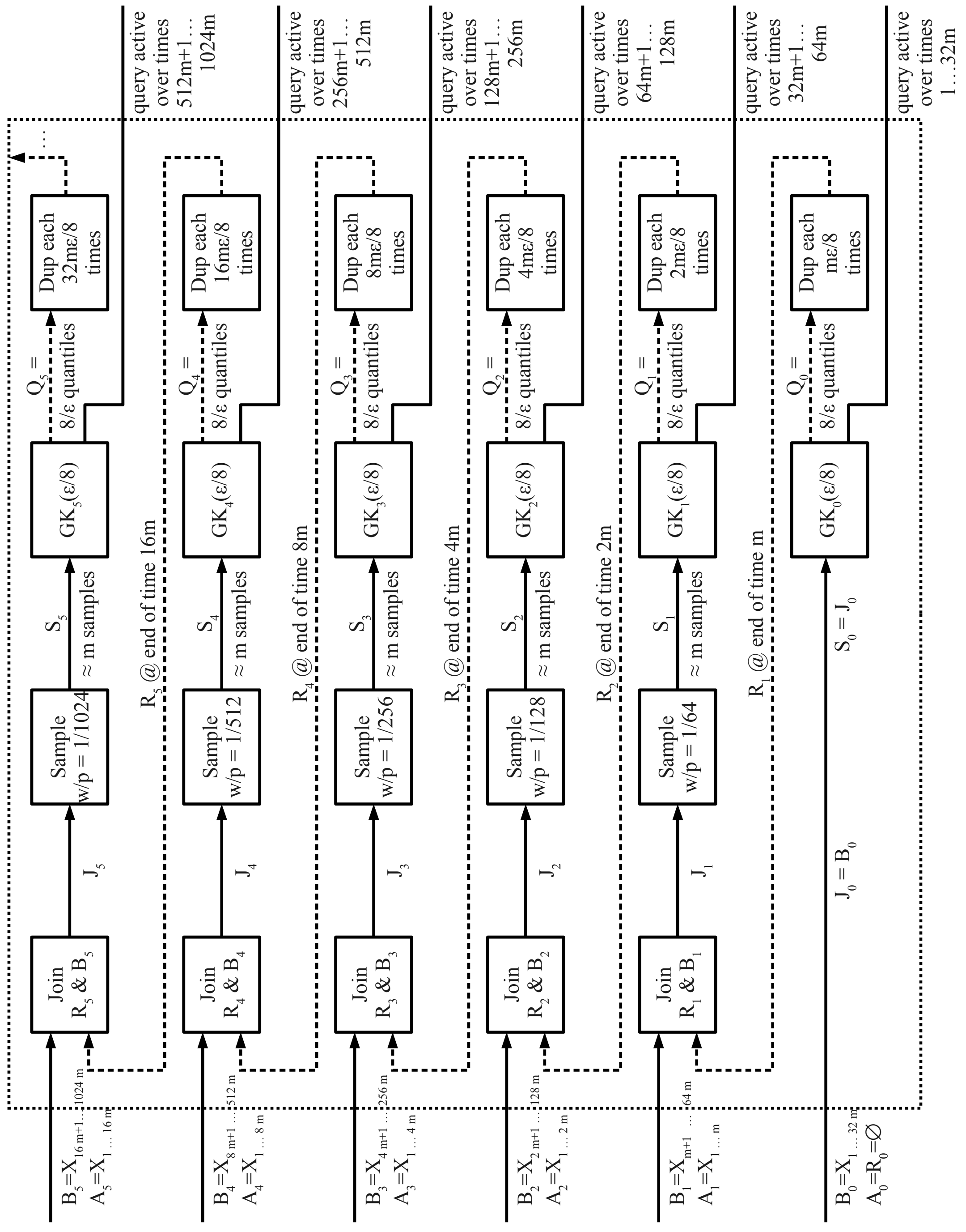

Figure 1.2: Each row r has its own copy G_r of the GK algorithm that approximates its input to $\varepsilon/8$ relative error. A_r is the prefix stream of row r, B_r is its suffix stream, R_r is its prefix stream replacement (generated by the previous row), J_r is the joint stream R_r followed by B_r, S_r is its sample stream, and Q_r is a one-time stream generated from G_r at time $2^r m$ to get the replacement prefix R_{r+1}.

Our error analysis, on the other hand, will require us to look back as many as $\Theta(\log 1/\varepsilon)$ rows to ensure our approximation guarantee. We stress that we will not need to actually *store* these $\Theta(\log 1/\varepsilon)$ rows for our guarantee to hold; we will only need that they didn't have any bad events (as will be defined) when they *were* alive.

1.3.1 Algorithm description

Our algorithm works in rows. Each row r has its own copy G_r of the GK algorithm that approximates its input to $\varepsilon/8$ relative error. For each row r we define several streams: A_r is the prefix stream of row r, B_r is its suffix stream, R_r is its prefix stream replacement (generated by the previous row), J_r is the joint stream R_r followed by B_r, S_r is its sample stream, and Q_r is a one-time stream generated from G_r by querying it with ranks $\rho_1 \dots \rho_{8/\varepsilon}$, where $\rho_q = q(\varepsilon/8)(m/32)$ for $r \geq 1$ and $\rho_q = q\varepsilon m/8$ for $r = 0$.

The prefix stream $A_r = X(2^{r-1}m)$ for row $r \geq 1$, importantly, is not directly received by row r. Instead, at the end of timestep $2^{r-1}m$, row $r-1$ generates Q_{r-1} and duplicates each of those $8/\varepsilon$ items $2^{r-1}\varepsilon m/8$ times to get the replacement prefix R_r, which is then immediately fed into row r before timestep $2^{r-1}m+1$ begins.

Each row can be *live* or not and *active* or not. Row 0 is live in timesteps $1 \dots 32m$ and row $r \geq 1$ is live in timesteps $2^{r-1}m+1 \dots 2^r 32m$. Live rows require space; once a row is no longer live we can free up the space it used. Row 0 is active in timesteps $1 \dots 32m$ and row $r \geq 1$ is active in timesteps $2^r 16m+1 \dots 2^r 32m$. This definition means that exactly one row $r(t)$ is active in any given timestep t. Any queries that are asked in timestep t are answered by $G_{r(t)}$. Given query ρ, we ask $G_{r(t)}$ for $\rho/2^{r(t)}32$ (if $r \geq 1$) or for ρ (if $r = 0$) and return the result.

At each timestep t, when item x_t arrives, it is fed as the next item in the suffix stream B_r for each live row r. B_r joined with R_r defines the joined input stream J_r. For $r \geq 1$, J_r is downsampled to the sample stream S_r by sampling each item independently with probability $1/2^r 32$. For row 0, no downsampling is performed, so $S_0 = J_0$. Lastly, S_r is fed into G_r.

Figure 1.2 shows the operation of and the communication between the first six rows. Solid arrows indicate continuous streams and dashed arrows indicate one-time messages. Algorithm 1 is a pseudocode listing of the algorithm.

```
Initially, allocate space for $G_0$. Mark row 0 as live and active.
**for** $t = 1, 2, \ldots$ **do**
    **foreach** *live row* $r \geq 0$ **do**
        **with probability** $1/2^r 32$ **do**
            Insert $x_t$ into $G_r$.
    **if** $t = 2^{r-1} m$ *for some* $r \geq 1$ **then**
        Allocate space for $G_r$. Mark row $r$ as live.
        Query $G_{r-1}$ with $\rho_1 \ldots \rho_{8/\varepsilon}$ to get $y_1 \ldots y_{8/\varepsilon}$.
        **for** $q = 1 \ldots 8/\varepsilon$ **do**
            **for** $1 \ldots 2^{r-1} \varepsilon m/8$ **do**
                **with probability** $1/2^r 32$ **do**
                    Insert $y_q$ into $G_r$.
    **if** $t = 2^r 16m$ *for some* $r \geq 1$ **then**
        Mark row $r$ as active. Unmark row $r-1$ as active.
    **if** $t = 2^r 32m$ *for some* $r \geq 0$ **then**
        Unmark row $r$ as live. Free space for $G_r$.
**on** *query* $\rho$ **do**
    Let $r = r(t)$ be the active row.
    Query $G_r$ for rank $\rho/2^r 32$ (if $r \geq 1$) or for rank $\rho$ (if $r = 0$).
    Return the result.
```

Algorithm 1: Procedural listing of the algorithm in section 1.3.1.

1.3.2 Error analysis

Define $C_r = x(2^r 32m+1), x(2^r 32m+2), \ldots$ and Y_r to be R_r followed by B_r and then C_r. That is, Y_r is just the continuation of J_r for the entire length of the input stream.

Fix some time t. All of our claims will be relative to time t; that is, if we write S_r we mean $S_r(t)$. Our error analysis proceeds as follows. We start by proving that $R(y, Y_r)$ is a good approximation of $R(y, Y_{r-1})$ when certain conditions hold for S_{r-1}. By induction, this means $R(y, Y_r)$ is a good approximation of $R(y, X = Y_0)$ when the conditions hold for all of $S_0 \dots S_{r-1}$, and actually it's enough for the conditions to hold for just $S_{r-\log 1/\varepsilon} \dots S_{r-1}$ to get a good approximation. Having proven this claim, we then prove that the result $y = y(\rho)$ of a query to our summary has $R(y, X)$ close to ρ. Lastly, we show that $m = O(\text{poly}(1/\varepsilon))$ suffices to ensure that the conditions hold for $S_{r-\log 1/\varepsilon} \dots S_{r-1}$ with very high probability $(1 - e^{-1/\varepsilon})$.

Lemma 1.3.1. *Let α_r be the event that $|S_r| > 2m$ and let β_r be the event that any of the first $\leq 2m$ samples z in S_r has $|2^r 32 R(z, S_r) - R(z, Y_r)| > \varepsilon t/8$. Say that S_r is* good *if neither α_r nor β_r occur (or if $r = 0$).*

For all rows $r \geq 1$ such that $t \geq t_r = 2^{r-1}m$, and all for all items y, if S_{r-1} is good then we have that $|R(y, Y_r) - R(y, Y_{r-1})| \leq 2^r \varepsilon m$.

Proof. At the end of time t_r we have $Y_r(t_r) = R_r(t_r)$, which is each item $y(\rho_q)$ in Q_{r-1} duplicated $\varepsilon t_r/8$ times. If $S_{r-1}(t_r)$ is good then theorem 1.2.1 and the GK guarantee imply $|R(y(\rho_q), Y_{r-1}(t_r)) - 2^{r-1}32\rho_q| \leq \varepsilon t_r/2$.

Fix q so that $y(\rho_q) \leq y < y(\rho_{q+1})$, where $y(\rho_0)$ and $y(\rho_{1+8/\varepsilon})$ are defined to be $\inf \mathcal{D}$ and $\sup \mathcal{D}$ for completeness. By fixing q this way we ensure that $R(y, Y_r(t_r)) = 2^{r-1}32\rho_q$. By the above bound on $R(y(\rho_q), Y_{r-1}(t_r))$ we also have that $2^{r-1}32\rho_q - \varepsilon t_r/2 \leq R(y, Y_{r-1}(t_r)) < 2^{r-1}32\rho_{q+1} + \varepsilon t_r/2$.

Recalling that $\rho_q = q\varepsilon m/256$, these bounds imply that

$$|R(y, Y_r(t_r)) - R(y, Y_{r-1}(t_r))| \leq 2^r \varepsilon m$$

For each time t after t_r, the new item x_t changes the rank of y in both streams Y_r

and Y_{r-1} by the same additive offset, so

$$|R(y, Y_r) - R(y, Y_{r-1})| \;=\; |R(y, Y_r(t_r)) - R(y, Y_{r-1}(t_r))| \;\leq\; 2^r \varepsilon m$$

yielding the lemma. □

By applying this lemma inductively we can bound the difference between Y_r and $X = Y_0$:

Corollary 1.3.2. *For all $r \geq 1$ such that $t \geq t_r = 2^{r-1}m$, if all of $S_0(t_1), S_1(t_2), \ldots, S_{r-1}(t_r)$ are good, then $|R(y, Y_r) - R(y, X)| \leq 2 \cdot 2^r \varepsilon m$.*

To ensure that all of these S_i are good would require m to grow with n, which would be bad. Happily, it is enough to require only the last $\log_2 1/\varepsilon$ sample summaries to be good, since the other items we disregard constitute only a small fraction of the total stream.

Corollary 1.3.3. *Let $d = \log_2 1/\varepsilon$. For all $r \geq 1$ such that $t \geq t_r = 2^{r-1}m$, if all of $S_{r-1}(t_r), \ldots, S_{r-d}(t_{r-d+1})$ are good, then $|R(y, Y_r) - R(y, X)| \leq 2^{r+2}\varepsilon m$.*

Proof. By lemma 1.3.1 we have $|R(y, Y_r) - R(y, Y_{r-d})| \leq 2^{r+1}\varepsilon m$. At time $t \geq t_{r-d}$, Y_{r-d} and X share all except possibly the first $2^{(r-d)-1}m = 2^{r-1}m/2^d = 2^{r-1}\varepsilon m$ items. Thus

$$\begin{aligned} |R(y, Y_r) - R(y, X)| \;&\leq\; |R(y, Y_r) - R(y, Y_{r-d})| + |R(y, Y_{r-d}) - R(y, X)| \\ &\leq\; 2^{r+1}\varepsilon m + 2^r \varepsilon m \end{aligned}$$

proving the corollary. □

We now prove that if the last several sample streams were good then querying our summary will give us a good result.

Lemma 1.3.4. *Let $d = \log_2 \frac{1}{\varepsilon}$ and $r = r(t)$. If all $S_r(t), S_{r-1}(t_r), \ldots, S_{r-d}(t_{r-d+1})$ are good, then querying our summary with rank ρ (= querying the active GK summary G_r with $\rho/2^r 32$ if $r \geq 1$, or with ρ if $r = 0$) returns $y = y(\rho)$ such that $|R(y, X) - \rho| \leq \varepsilon t$.*

Proof. For $r \geq 1$ we have that $|R(y, Y_r) - R(y, X)| \leq 2^{r+2}\varepsilon m \leq \varepsilon t/2$ by corollary 1.3.3. By theorem 1.2.1 and the GK guarantee, $|R(y, Y_r) - \rho| \leq \varepsilon t/2$.

For $r = 0$, the GK guarantee alone proves the lemma. □

Lastly, we prove that $m = O(\text{poly}(1/\varepsilon))$ suffices to ensure that all of $S_r(t)$, $S_{r-1}(t_r), \ldots, S_{r-d}(t_{r-d+1})$ are good with probability at least $1 - e^{-1/\varepsilon}$.

Lemma 1.3.5. *Let $d = \log_2 1/\varepsilon$ and $r = r(t)$. If $m \geq \frac{400000 \ln 1/\varepsilon}{\varepsilon^2}$ then all of $S_r(t), S_{r-1}(t_r), \ldots, S_{r-d}(t_{r-d+1})$ are good with probability at least $1 - e^{-1/\varepsilon}$.*

Proof. There are at most $1 + \log_2 1/\varepsilon \leq 4 \ln 1/\varepsilon$ of these summary streams total. Theorem 1.2.1 and the union bound give us $P(\text{some } \alpha_r \text{ occurs}) \leq 4 \ln \frac{1}{\varepsilon} \exp(-\frac{m}{192})$ and $P(\text{some } \beta_r \text{ occurs}) \leq 16m \ln \frac{1}{\varepsilon} \exp(-\frac{\varepsilon^2 m}{12288})$.

Together, $P = P(\text{some } S_r \text{ is not good}) \leq 20m \ln \frac{1}{\varepsilon} \exp(-\varepsilon^2 m/12288)$. It suffices to choose $m \geq \frac{400000 \ln 1/\varepsilon}{\varepsilon^2}$ to obtain $P \leq e^{-1/\varepsilon}$. □

1.3.3 Space and time complexity

A minor issue with the algorithm is that, as written in section 1.3.1, we do not actually have a bound on the worst-case space complexity of the algorithm; we only have a bound on the space needed at any given point in time. This issue is due to the fact that there are low probability events in which $|S_r|$ can get arbitrarily large and the fact that over n items there are a total of $\Theta(\log n)$ sample streams. The space complexity of the algorithm is $O(\max |S_r|)$, and to bound this value with constant probability using the Chernoff bound appears to require that $\max |S_r| = \Omega(\log \log n)$, which is too big.

Fortunately, fixing this problem is simple. Instead of feeding every sample of S_r into the GK summary G_r, we only feed each next sample if G_r has seen $< 2m$ samples so far. That is, we deterministically restrict G_r to receiving only $2m$ samples. Lemmas 1.3.1 through 1.3.4 condition on the goodness of the sample streams S_r, which ensures that the G_r receive at most $2m$ samples each, and the claim of lemma 1.3.5 is independent of the operation of G_r. Therefore, by restricting each G_r to receive at most $2m$ inputs we can ensure that the space complexity is deterministically $O(\frac{1}{\varepsilon} \log \frac{1}{\varepsilon})$ without breaking our error guarantees.

From a practical perspective, the assumption in the streaming setting is that new items arrive over the input stream X at a high rate, so both the worst-case per-item processing time as well as the amortized time to process n items are important. For our per-item time complexity, the limiting factor is the duplication step that occurs at the end of each time $t_r = 2^{r-1}m$, which makes the worst-case per-item processing time as large as $\Theta(n)$. Instead, at time t_r we could generate Q_{r-1} and store it in $O(1/\varepsilon)$ words, and then on each arrival $t = 2^{r-1}m+1 \ldots 2^r m$ we could insert both x_t and also the next item in R_r. By the time $t_{r+1} = 2t_r$ that we generate Q_r, all items in R_r will have been inserted into J_r. Thus the worst-case per-item time complexity is $O(\frac{1}{\varepsilon} T_{\text{GK}}^{\max})$, where $T_{\text{GK}}^{\max}$ is the worst-case per-item time to query or insert into one of our GK summaries. Over $2^r 32m$ items there are at most $2m$ insertions into any one GK summary, so the amortized time over n items in either case is $O(\frac{m \log(n/m)}{n} T_{\text{GK}})$, where T_{GK} is the amortized per-item time to query or insert into one of our GK summaries. Algorithm 2 includes the changes of this section.

1.4 Discussion

Our starting point is a very natural idea that appears to have been first used in Manku et. al. [MRL99]. This key idea is to downsample the input stream and feed

```
Initially, allocate space for $G_0$. Mark row 0 as live and active.
for $t = 1, 2, \ldots$ do
    foreach live row $r \geq 0$ do
        with probability $1/2^r 32$ do
            Insert $x_t$ into $G_r$ if $G_r$ has seen $< 2m$ insertions.
        if $r \geq 1$ and $2^{r-1}m < t \leq 2^r m$ and $G_r$ has seen $< 2m$ insertions
        then
            with probability $1/2^r 32$ do
                Also insert item $t - 2^{r-1}m$ of $R_r$ into $G_r$.
    if $t = 2^{r-1}m$ for some $r \geq 1$ then
        Allocate space for $G_r$. Mark row $r$ as live.
        Query $G_{r-1}$ with $\rho_1 \ldots \rho_{8/\varepsilon}$ to get $Q_{r-1} = y_1 \ldots y_{8/\varepsilon}$.
        Store $Q_{r-1}$, to implicitly define $R_r$.
    if $t = 2^r 16m$ for some $r \geq 1$ then
        Mark row $r$ as active. Unmark row $r-1$ as active.
    if $t = 2^r 32m$ for some $r \geq 0$ then
        Unmark row $r$ as live. Free space for $G_r$.
on query $\rho$ do
    Let $r = r(t)$ be the active row.
    Query $G_r$ for rank $\rho/2^r 32$ (if $r \geq 1$) or for rank $\rho$ (if $r = 0$).
    Return the result.
```

Algorithm 2: Procedural listing of the algorithm in section 1.3.3. The changes between sections 1.3.1 and 1.3.3 are that G_r never has more than $2m$ insertions and that stream R_r is paired with items in B_r.

the resulting sample stream into a deterministic summary data structure (compare our figure 1.1 with figure 1 on page 254 of [MRL99]). At a very high level, we are simply replacing their deterministic $O(\frac{1}{\varepsilon}\log^2 \varepsilon n)$ MRL summary [MRL98] with the deterministic $O(\frac{1}{\varepsilon}\log \varepsilon n)$ GK summary [GK01].

Our implementation of this idea, however, is conceptually different from the implementation of Manku et. al. in two respects. First, we use the GK algorithm strictly as a black box, whereas Manku et. al. peek into the internals of their MRL algorithm, using its algorithm-specific interface (New, Collapse, Output) rather than the more generic interface (Insert, Query). At an equivalent level, dealing with the GK algorithm is already unpleasant—the space complexity analysis in [GK01] is quite involved. Using the generic interface, our implementation could just as easily replace the GK boxes in the diagram in figure 1.2 with MRL boxes; or, for the bounded universe model, with boxes running the q-digest summary of Shrivastava et. al. [SBA04].

The second respect in which our algorithm differs critically from that of Manku et. al. is that we operate on *streams* rather than on stream *items*. We use this approach in our proof strategy too; the key step in our error analysis, lemma 1.3.1, is a statement about (what to us are) static objects, so we can trade out the complexity of dealing with time-varying data structures for a simple induction. This idea of developing streaming algorithms with analyses that hinge on analyzing streams rather than just stream items may be a generally useful design approach. The way we implemented this idea in our method for reducing a deterministic summary to a randomized summary was:

1. For a fixed n, downsample the input stream, feed the resulting sample stream into the deterministic summary, and prove a probabilistic bound.
2. Run an infinite number of copies of step 1, for exponentially growing values of n.
3. Replace a constant fraction prefix of each copy with an approximation generated by the previous copy, and prove using step 1 that this approximation probably doesn't cause too much error.
4. Use step 3 inductively to prove a probabilistic bound for the entire stream.

CHAPTER 2

Variability in data streams

We consider the problem of tracking with small relative error an integer function $f(n)$ defined by a distributed update stream $f'(n)$. Existing streaming algorithms with worst-case guarantees for this problem assume $f(n)$ to be monotone; there are very large lower bounds on the space requirements for summarizing a distributed non-monotonic stream, often linear in the size n of the stream.

Input streams that give rise to large space requirements are highly variable, making relatively large jumps from one timestep to the next. However, in practice the impact on $f(n)$ of any single update $f'(n)$ is usually small. In this chapter we propose a framework for non-monotonic streams that admits algorithms whose worst-case performance is as good as existing algorithms for monotone streams and degrades gracefully for non-monotonic streams as those streams vary more quickly. We introduce a new stream parameter, the "variability" v, deriving its definition in a way that shows it to be a natural parameter to consider for non-monotonic streams. It is also a useful parameter. From a theoretical perspective, we can adapt existing algorithms for monotone streams to work for non-monotonic streams, with only minor modifications, in such a way that they reduce to the monotone case when the stream happens to be monotone, and in such a way that we can refine the worst-case communication bounds from $\Theta(n)$ to $\tilde{O}(v)$. From a practical perspective, we demonstrate that v can be small in practice by proving that v is $O(\log f(n))$ for monotone streams and $o(n)$ for streams that are "nearly" monotone or that are generated by random walks.

2.1 Introduction

In the distributed monitoring model, there is a single central monitor and several (k) observers. The observers receive data and communicate with the monitor, and the goal is to maintain at the monitor a summary of the data received at the observers while minimizing the communication between them.

This model was introduced by Cormode, Muthukrishnan, and Yi [CMY08, CMY11] with the motivating application of minimizing radio energy usage in sensor networks, but can be applied to other distributed applications like determining network traffic patterns. Since the monitor can retain all messages received, algorithms in the model can be used to answer historical queries too, making the model useful for auditing changes to and verifying the integrity of time-varying datasets.

The distributed monitoring model has also yielded several theoretical results. These include algorithms and lower bounds for tracking total count [CMY08, CMY11, LRV11, LRV12], frequency moments [CMY08, CMY11, HYZ12, WZ11, WZ12], item frequencies [HYZ12, WZ11, WZ12, YZ09, YZ13], quantiles [HYZ12, WZ11, WZ12, YZ09, YZ13], and entropy [ABC09, WZ11, WZ12] to small relative error.

However, nearly all of the upper bounds assumed that data is only inserted and never deleted. This is unfortunate because in the standard turnstile streaming model, all of these problems have similar algorithms that permit both insertions and deletions. In general, this unfortunate situation is unavoidable; existing lower bounds for the distributed model [ABC09] demonstrate that it is not possible to track even the total item count in small space when data is permitted to be deleted.

That said, when restrictions are placed on the types of allowable input, the lower bounds evaporate, and very nice upper bounds exist. Tao, Yi, Sheng, Pei, and Li [TYS10] developed algorithms for the problem of summarizing the order statistics history of a dataset D over an insertion/deletion stream of size n, which has an $\Omega(n)$-bit lower bound in general; however, they performed an interesting analysis that yielded online and offline upper bounds proportional to $\sum_{t=1}^{n} 1/|D(t)|$, with a nearly matching lower bound. A year or two later, Liu, Radunović, and Vojnović [LRV11, LRV12] considered the problem of tracking $|D|$ under random inputs; for general inputs, there is an $\Omega(n)$-bit lower bound, but Liu et. al. obtained (among other results) expected communication costs proportional to $\sqrt{n}\log n$ when the insertion/deletion pattern is the result of fair coin flips.

In fact, the pessimistic lower bounds for the general case can occur only when the input stream is such that the quantity being tracked is forced to vary quickly. In the problems considered by Tao et. al. and Liu et. al., this occurs when $|D|$ is usually small. These two groups avoid this problem in two different ways: Tao et. al. provide an analysis that yields a worst-case upper bound that is small when $|D|$ is usually large, and Liu et. al. consider input classes for which $|D|$ is usually large in expectation.

Our contributions In this paper we propose a framework that extends the analysis of Tao et. al. to the distributed monitoring model and that permits worst-case analysis that can be specialized for random input classes considered by Liu et. al. In so doing, we explain the intuition behind the factor of $\sum_{t=1}^{n} 1/|D(t)|$ in the bounds of Tao et. al. and how we can separate the different sources of randomness that appear in the algorithms of Liu et. al. to obtain worst-case bounds for the random input classes we also consider.

In the next section we derive a stream parameter, the variability v. We prove that v is $O(\log f(n))$ for monotone streams and $o(n)$ for streams that are "nearly" monotone or that are generated by random walks, and find that the bounds of Tao et. al. and Liu et. al. are stated nicely in terms of v. In section 2.3 we combine ideas from the upper bounds of Tao et. al. [TYS10] with the existing distributed counting algorithms of Cormode et. al. [CMY08, CMY11] and Huang, Yi, and Zhang [HYZ12] to obtain upper bounds for distributed counting that are proportional to v. In section 2.4 we show that our dependence on v is essentially necessary by developing deterministic and randomized space+communication lower bounds that hold even when v is small. We round out the piece in section 2.5 with a discussion of the suitability of variability as a general framework, in which we extend the ideas of section 2.3 to the problems of distributed tracking of item frequencies and of tracking general aggregates when $k = 1$.

But before we jump into the derivation of variability, we define our problem formally and abstract away unessential details.

Problem definition The problem is that of tracking at the coordinator an integer function $f(n)$ defined by an update stream $f'(n)$ that arrives online at the sites. Time occurs in discrete steps; to be definite, the first timestep is 1, and we define $f(0) = 0$ unless stated otherwise. At each new current time n the value $f'(n) = f(n) - f(n-1)$ appears at a single site $i(n)$.

There is an error parameter ε that is specified at the start. The requirement is that, after each timestep n, the coordinator must have an estimate $\hat{f}(n)$ for $f(n)$ that is usually good. In particular, for deterministic algorithms we require that $\forall n,\ |f(n) - \hat{f}(n)| \leq \varepsilon f(n)$, and for randomized algorithms we require that $\forall n,\ P(|f(n) - \hat{f}(n)| \leq \varepsilon f(n)) \geq 2/3$.

2.2 Variability

In the original distributed monitoring paper [CMY08], Cormode et. al. define a general thresholded problem $(k, f, \tau, \varepsilon)$. A dataset D arrives as a distributed stream across k sites. At any given point in time, the coordinator should be able to determine whether $f(D) \geq \tau$ or $f(D) \leq (1-\varepsilon)\tau$.

In continuous tracking problems, there is no single threshold, and so $f(n)$ is tracked to within an additive $\varepsilon\tau(n)$, where $\tau(n)$ also changes with the dataset $D(n)$. Since τ is now a function, it needs to be defined; the usual choice is f itself, except for tracking item frequencies and order statistics, for which (following the standard streaming model) τ is chosen to be $|D|$. That is, the continuous monitoring problem (k, f, ε) is, at all times n maintain at the coordinator an estimate $\hat{f}(n)$ of $f(n)$ so that $|f(n) - \hat{f}(n)| \leq \varepsilon f(n)$.

The intuition for the way we define variability is from looking at the situation as though item arrivals and communication occur continuously. That is, over $n = [0.1, 0.2]$ we receive the second tenth of the first item, for example. At any time t at which f changes by $\pm\varepsilon f$, we would need to communicate at least one message to keep the coordinator in sync; so if f changes by $f'(t)\,dt$ then we should communicate $|\frac{f'(t)\,dt}{\varepsilon f(t+dt)}|$ messages.

With discrete arrivals, $dt = 1$. Otherwise, the idea remains the same, so we would expect the total number of messages to look like $\sum_{t=1}^{n} |\frac{f'(t)}{\varepsilon f(t)}|$, where here we define $f'(t) = f(t) - f(t-1)$ to simplify the expressions. In sections 2.3 and 2.4 we find that, modulo the number k of sites and constant factors, this is indeed the case.

Being a parameter of the problem rather than of the stream, we can move the $1/\varepsilon$ factor out of our definition of variability and bring it back in along with the appropriate functions of k when we state upper and lower bounds for our problem. This permits us to treat the stream parameter v independently of the problem. We also need to handle the case $f = 0$ specially, which we can do by communicating at each timestep that case occurs.

Keeping all of this in mind, we define the *f-variability* of a stream to be

$v(n) = \sum_{t=1}^{n} \min\{1, |\frac{f'(t)}{f(t)}|\}$. We also write $v'(t) = \min\{1, |\frac{f'(t)}{f(t)}|\}$ to be the increase in variability at time t. We say "variability" for f-variability in the remainder of this paper.

From a practical perspective, we believe low variability streams to be common. In many database applications the database is interesting primarily because it tends to grow more than it shrinks, so it is common for the size of the dataset to have low variability; as more items are inserted, the rate of change of $|D|$ shrinks relative to itself, and about as many deletions as insertions would be required to keep the ratio constant. In the following subsection, we prove that monotone and nearly monotone functions have low variability and that random walks have low variability in expectation, lending evidence to our belief.

From a theoretical perspective, variability is a way to analyze algorithms for ε relative error in the face of non-monotonicity and generate provable worst-case bounds that degrade gracefully as our assumptions about the input become increasingly pessimistic. For our counting problem, it allows us to adapt the existing distributed counting algorithms of Cormode et. al. [CMY08, CMY11] and Huang et. al. [HYZ12] with only minor modifications, and the resulting analyses show that the dependence on k and ε remains unchanged.

2.2.1 Interesting cases with small variability

We start with functions that are nearly monotone in the sense that they are eventually mostly nondecreasing. We make this precise in the theorem statement.

Theorem 2.2.1. *Let $f^-(n) = \sum_{t:f'(t)<0} |f'(t)|$ and $f^+(n) = \sum_{t:f'(t)>0} f'(t)$. If there is a monotone nondecreasing function $\beta(t) \geq 1$ and a constant t_0 such that for all $n \geq t_0$ we have $f^-(n) \leq \beta(n)f(n)$, then the variability $\sum_{t=1}^{n} |f'(t)/f(t)|$ is $O(\beta(n)\log(\beta(n)f(n)))$.*

The proof partitions time into intervals over which $f^+(t)$ doubles and shows the variability in each interval to be $O(\beta(n))$.

Proof. For $i = 1, \ldots, k$, define t_i to be the earliest time t such that we have $f^+(t_i) > 2f^+(t_{i-1})$, where k is the smallest index such that $t_k > n$. (If k is undefined, define $k = n + 1$.)

The cost $\sum_{t=1}^{t_0-1} |f'(t)/f(t)|$ is constant. We bound the cost $\sum_{t=t_0}^{n} |f'(t)|/f(t)$ as follows. We partition the interval $[t_0, t_k)$ into subintervals $[t_0, t_1), \ldots, [t_{k-1}, t_k)$ and sum over the times t in each one. There are at most $1 + \log f^+(n)$ of these subintervals.

$$\begin{aligned}
\sum_{t=t_0}^{n} \frac{|f'(t)|}{f(t)} &\leq \sum_{i=1}^{k} \sum_{t=t_{i-1}}^{t_i-1} \frac{|f'(t)|}{f(t)} \leq \sum_{i=1}^{k} \frac{1+\beta(n)}{f^+(t_{i-1})} \sum_{t=t_{i-1}}^{t_i-1} |f'(t)| \\
&\leq \sum_{i=1}^{k} (1+\beta(n)) \frac{f^+(t_i-1) + f^-(t_i-1)}{f(t_{i-1})} \\
&\leq 4(1+\beta(n))(1+\log f^+(t_i-1)) \\
&\leq 4(1+\beta(n))(1+\log(2(1+\beta(n))f(n)))
\end{aligned}$$

because the condition $f^-(t) \leq \beta(t)f(t)$ implies $f(t) \geq f^+(t)/(1+\beta(t))$ and $f^-(t) \leq f^+(t)$. □

When $f(n)$ is strictly monotone, $\beta(n) = 1$ suffices, and the theorem reduces to the result claimed in the abstract. As we will see in section 2.3, our upper bounds will simplify in the monotone case to those of Cormode et. al. [CMY08, CMY11] and Huang et. al. [HYZ12].

Next, we compute the variability for two random input classes considered by Liu et. al. [LRV11, LRV12]. This will permit us to decouple the randomness of their algorithms from the randomness of their inputs. This means, for example, that even our deterministic algorithm of section 2.3 has $o(n)$ cost in expectation for these input classes. The first random input class we consider is the symmetric random walk.

Theorem 2.2.2. *If $f'(t)$ is a stream of i.i.d. ± 1 coin flips then the expected variability $E(v(n)) = O(\sqrt{n}\log n)$.*

Proof. The update sequence defines a random walk for $f(t)$, and the expected variability is

$$\sum_{t=1}^{n} P(f(t)=0) \; + \; \sum_{t=1}^{n}\sum_{s=1}^{t} 2P(f(t)=s)/s$$

We use the following fact, mentioned and justified in Liu et. al. [LRV11]:

Fact 2.2.3. *For any $t \geq 1$ and $s \in [-t, t]$ we have $P(f(t)=s) \leq c_1/\sqrt{t}$, where c_1 is some constant.*

Together, these show the expected cost to be at most

$$c_1 \sum_{t=1}^{n} (1 + 2H_t)/\sqrt{t} \;\leq\; c_2 \log(n) \sum_{t=1}^{n} 1/\sqrt{t} \;\leq\; c_3 \log(n)\sqrt{n}$$

since $(1 + 2H_n) \leq \frac{c_2}{c_1}\log(n)$ and $\sum_{t=1}^{n} 1/\sqrt{t} \leq \frac{c_3}{2c_2}\int_1^n 1/\sqrt{t}\,dt$. $\square$

The second random input class we consider is i.i.d. increments with a common drift rate of $\mu > 0$. The case $\mu < 0$ is symmetric. We assume that μ is constant with respect to n. The proof is a simple application of Chernoff bounds.

Theorem 2.2.4. *If $f'(t)$ is a sequence of i.i.d. ± 1 random variables having $P(f'(t)=1) = (1+\mu)/2$ then $E(v(n)) = O(\frac{\log n}{\mu})$.*

Proof. We show that, with high probability, $f(t) \geq \mu t/2$ for times $t \geq t_0 = t_0(n)$ when n is large enough with respect to μ.

We write $f(t) = -t + 2Y_t$, where $Y_t = \sum_{s=1}^{t} y_s$, and y_s is a Bernoulli variable with mean $\frac{1+\mu}{2}$. We have that $P(f(t) \leq \mu t/2) = P(Y_t \leq \frac{2+\mu}{4}t)$ and also that $E(Y_t) = \frac{1+\mu}{2}t$. Further, $P(Y_t \leq \frac{2+\mu}{4}t) \leq \exp(-\mu t/16)$ using a Chernoff bound. Let A be the event $\exists t \geq t_0\,(f(t) \leq \mu t/2)$. Then $P(A) \leq \sum_{t=t_0}^{n} e^{-\mu t/16}$ by the union bound. We can upper bound this sum by

$$\sum_{t=t_0}^{n} e^{-\mu t/16} \;\le\; e^{-\mu t_0/16} + \int_{t_0}^{n} e^{-\mu t/16}\, dt \;\le\; 17 e^{-\mu t_0/16}/\mu$$

Taking $t_0 = (16/\mu)\ln(17n/\mu)$ gives us $P(A) \le 1/n$. Thus

$$E\left(\sum_{t=1}^{n} \min\left\{1, \left|\frac{f'(t)}{f(t)}\right|\right\}\right) \;\le\; t_0 + \left(\frac{1}{n}\right) n + \left(1 - \frac{1}{n}\right)\sum_{t=t_0}^{n} \frac{2}{\mu t} \;=\; O\left(\frac{\log n}{\mu}\right)$$

yielding the theorem. □

Remarks We can restate the results of Liu et. al. [LRV11, LRV12] and Tao et. al. [TYS10] in terms of variability. For unbiased coin flips, Liu et. al. obtain an algorithm that uses $O(\frac{\sqrt{k}}{\varepsilon}\sqrt{n}\log n)$ messages (of size $O(\log n)$ bits each) in expectation, and for biased coin flips with constant μ, an algorithm that uses $O(\frac{\sqrt{k}}{\varepsilon}\frac{1}{|\mu|}(\log n)^{1+c})$ messages in expectation. If we rewrite these bounds in terms of expected variability, they become $O(\frac{\sqrt{k}}{\varepsilon}E(v(n)))$ and $O(\frac{\sqrt{k}}{\varepsilon}(\log n)^c E(v(n)))$, respectively. In the next section, we obtain (when $k = O(1/\varepsilon^2)$) a randomized bound of $O(\frac{\sqrt{k}}{\varepsilon}v(n))$. In marked contrast to the bounds of Liu et. al., our bound is a worst-case lower bound that is a function of $v(n)$; if the input happens to be generated by fair coin flips, then our expected cost happens to be $O(\frac{\sqrt{k}}{\varepsilon}\sqrt{n}\log n)$.

The results of Tao et. al. are for a different problem, but they can still be stated nicely in terms of the $|D|$-variability $v(n)$: for the problem of tracking the historical record of order statistics, they obtain a lower bound of $\Omega(\frac{1}{\varepsilon}v(n))$ and offline and online upper bounds of $O((\frac{1}{\varepsilon}\log^2\frac{1}{\varepsilon})v(n))$ and $O(\frac{1}{\varepsilon^2}v(n))$, respectively. We adapt ideas from both their upper and lower bounds in sections 2.3 and 2.4.

2.3 Upper bounds

In this section we develop deterministic and randomized algorithms for maintaining at the coordinator an estimate $\hat{f}(n)$ for $f(n)$ that is usually good. In particular, for deterministic algorithms we require that $\forall n,\ |f(n) - \hat{f}(n)| \leq \varepsilon f(n)$, and for randomized algorithms that $\forall n,\ P(|f(n) - \hat{f}(n)| \leq \varepsilon f(n)) \geq 2/3$. We obtain deterministic and randomized upper bounds of $O(\frac{k}{\varepsilon} v(n))$ and $O((k + \frac{\sqrt{k}}{\varepsilon}) v(n))$ messages, respectively. For comparison, the analogous algorithms of Cormode et. al. [CMY08, CMY11] and Huang et. al. [HYZ12] for the monotone case use $O(\frac{k}{\varepsilon} \log n)$ and $O((k + \frac{\sqrt{k}}{\varepsilon}) \log n)$ messages, respectively.

For our upper bounds we assume that $f'(n) = \pm 1$ always. If $|f'(n)| > 1$ we could simulate it with $|f'(n)|$ arrivals of ± 1 updates with $O(\log \max f'(n))$ overhead; in the following theorem and proof, we define $1/f(n) = 1$ when $f(n) = 0$ and assume that $f(n) \geq 0$ always, to simplify notation.

Theorem 2.3.1. *For $f'(n) > 1$ we have $\sum_{t=1}^{f'(n)} \frac{1}{f(n-1)+t} \leq \frac{f'(n)}{f(n)}(1 + H(f'(n)))$ and for $f'(n) < -1$ we have $\sum_{t=0}^{1-f'(n)} \frac{t}{f(n)+t} \leq \frac{3f'(n)}{f(n)}$, where $H(x)$ is the xth harmonic number.*

Proof. For $f'(n) > 1$, we have

$$\sum_{t=1}^{f'(n)} \frac{1}{f(n-1)+t} = \frac{f'(n)}{f(n)} + \frac{1}{f(n)} \sum_{t=1}^{f'(n)} \frac{f'(n) - t}{f(n-1)+t} \leq \frac{f'(n)}{f(n)} + \frac{f'(n)}{f(n)} \sum_{t=1}^{f'(n)} \frac{1}{t}$$

If $f'(n) < -1$ and $f(n) \geq 1$, then

$$\sum_{t=0}^{1-f'(n)} \frac{1}{f(n)+t} \leq \frac{1}{f(n)} + \ln\left(\frac{f(n-1)}{f(n)}\right) = \frac{1}{f(n)} + \ln\left(1 + \frac{|f'(n)|}{f(n)}\right) \leq \frac{2|f'(n)|}{f(n)}$$

and if $f(n) = 0$, add another $|f'(n)|/f(n)$. ☐

2.3.1 Partitioning time

We use an idea from Tao et. al. [TYS10] to first divide time into manageable blocks. At the end of each block we know the values n and $f(n)$ exactly. Within each block, we know these values only approximately. The division into blocks is deterministic and the same for both our deterministic and randomized algorithms. Our division ensures that the change in $v(n)$ over each block is at least $1/5$, which simplifies our analysis. Specifically, we prove

Theorem 2.3.2. *There is an algorithm to divide time into blocks $B_0, B_1, \ldots$, where $B_j = [n_j + 1, n_{j+1}]$, such that $n_0 = 0$, the change $v_j = v(n_{j+1}) - v(n_j)$ in variability over each block B_j is at least $1/5$, and the partitioning algorithm uses at most $25kv + 3k$ messages of size $O(\log n)$ bits each.*

The algorithm is:

- The coordinator requests the sites' values c_i and f_i at times $n_0 = 0, n_1, n_2, \ldots$ and then broadcasts a value r. These values will be defined momentarily.
- Each site i maintains a variable c_i that counts the number of stream updates $f'(n)$ it received since the last time it sent c_i to the coordinator. It also maintains f_i that counts the change in f it received since the last broadcast n_j. Whenever $c_i = \lceil 2^{r-1} \rceil$, site i sends c_i to the coordinator. This is in addition to replying to requests from the coordinator.
- The coordinator maintains a variable $\hat{t}$. After broadcasting r, $\hat{t}$ is reset to zero. Whenever site i sends c_i, the coordinator updates $\hat{t} = \hat{t} + c_i$.
- The coordinator also maintains variables $\hat{f}$, j, and t_j. At the first time $n_j > n_{j-1}$ at which $\hat{t} \geq t_j$, the coordinator requests the c_i and f_i values, updates $\hat{f}$ and r, sets $t_{j+1} = \lceil 2^{r-1} \rceil k$, broadcasts r, and increments j.
- When r is updated at the end of time n_j, it is set to r if $2^r 2k \leq |f(n_j)| < 2^r 4k$ and zero if $|f(n_j)| < 4k$.

Proof. Algebra tells us some facts:

- $\lceil 2^{r-1} \rceil k \ \leq \ n_{j+1} - n_j \ \leq \ 2^r k$.
- $|f(n) - f(n_j)| \leq 2^r 5k$ for all n in B_j.
- If $r \geq 1$ then $|f(n) - f(n_j)| \geq 2^r k$ for all n in B_j.

The total number of messages sent in block B_j is at most $5k$: we have at most $2k$ updates from sites, k requests from the coordinator, k replies from each site, and k broadcast at n_{j+1}.

The change in variability v_j over block B_j is

$$v_j \ = \ \sum_{t=n_j+1}^{n_{j+1}} \frac{1}{\min\{1, |f(t)|\}} \ \geq \ 2^r k / 2^r 5k \ \geq \ 1/5$$

And therefore the total number of messages is bounded by $25kv + 3k$. □

2.3.2 Estimation inside blocks

What remains is to estimate $f(n)$ within a given block. Since we have partitioned time into constant-variability blocks, we can use the algorithms of Cormode et. al. [CMY08, CMY11] and Huang et. al. [HYZ12] almost directly. Both of our algorithms use the following template, changing only CONDITION, MESSAGE, and UPDATE:

- Site i maintains a variable d_i that tracks the drift at site i, defined as the sum of $f'(n)$ updates received at site i during the block. I.e., $f(n) - f(n_j) = \sum_i d_i$.
- Site i also maintains a variable δ_i that tracks the change in d_i since the last time site i sent a MESSAGE. δ_i is initially zero.
- The coordinator maintains an estimate $\hat{d}_i$ for each value d_i. These are initially zero. It also defines two estimates based on these $\hat{d}_i$:
  - For the global drift: $\hat{d} = \sum_i \hat{d}_i$.

    - For $f(n)$: $\hat{f}(n) = f(n_j) + \hat{d}(n)$.

- When site i receives stream update $f'(n)$, it updates d_i. It then checks its CONDITION. If true, it sends a MESSAGE to the coordinator and resets $\delta_i = 0$.
- When the coordinator receives a MESSAGE from a site i it UPDATEs its estimates.

2.3.3 The deterministic algorithm

Our method guarantees that at all times n we have $|f(n) - \hat{f}(n)| \leq \varepsilon|f(n)|$. It uses $O(kv/\varepsilon)$ messages in total.

- CONDITION: true if $|\delta_i| = 1$ and $r = 0$, or if $|\delta_i| \geq \varepsilon 2^r$. Otherwise, false.
- MESSAGE: the new value of d_i.
- UPDATE: set $\hat{d}_i = d_i$.

Proof. Let $\delta = \sum_i \delta_i$ be the error with which $\hat{d}$ estimates $d = \sum_i d_i$. The error in $\hat{f}$ is

$$|f(n) - \hat{f}(n)| \;=\; |(f(n_j) + d(n)) \;-\; (\hat{f}(n_j) + d(n) + \delta(n))| \;=\; |\delta(n)|$$

When $r \geq 1$ we have $|B_j| \leq 2^r k$, and we always have that $\delta \leq |B_j|$. Since we constrain $\delta_i < \varepsilon 2^r$ at the end of each timestep, we maintain at the end of each timestep that $|f(n) - \hat{f}(n)| \;<\; \varepsilon 2^r k \;\leq\; \varepsilon|f(n)|$.

We also use at most $2k/\varepsilon$ messages for the block. If $r = 0$ then the number of messages is at most k. If $r \geq 1$, then since a site must receive $\varepsilon 2^r$ new stream updates to send a new message, and since there are at most $2^r k$ stream updates in the block, there are at most k/ε messages.

In each block the change in v is at least $1/5$, so the total number of messages is at most $5kv/\varepsilon$. □

2.3.4 The randomized algorithm

Our method uses $O(\sqrt{k}v/\varepsilon)$ messages (plus the time partitioning) and guarantees that at all times n we have $P(|f(n) - \hat{f}(n)| > \varepsilon|f(n)|) < 1/3$.

The idea is to estimate the sums d_i^+ and d_i^- of $+1$ and -1 updates separately. The estimators for those values are independent and monotone, so we can use the method of Huang et. al. [HYZ12] to estimate the two and then combine them.

Specifically, the coordinator and each site run two independent copies A^+ and A^- of the algorithm. Whenever $f'(n) = +1$ arrives at site i, a $+1$ is fed into algorithm A^+ at site i. Whenever $f'(n) = -1$ arrives at site i, a $+1$ is fed into algorithm A^- at site i. So the drifts d_i^+ and d_i^- at every site will always be non-negative. At the coordinator, the estimates $\hat{d}_i^\pm$ and $\hat{d}^\pm$ are tracked independently also. However, the coordinator also defines $\hat{d} = \hat{d}^+ - \hat{d}^-$ and $\hat{f}(n) = f(n_j) + \hat{d}(n)$. The definitions for algorithm $A^\pm$ are

- Condition: true with probability $p = \min\{1, 3/\varepsilon 2^r k^{1/2}\}$.
- Message: the new value of $d_i^\pm$.
- Update: set $\hat{d}_i^\pm = d_i^\pm - 1 + 1/p$.

Proof. The following fact 2.3.3 is lemma 2.1 of Huang et. al. [HYZ12]. Our algorithm effectively divides the stream $f'(B_j)$ into two streams $|f'(B_j^\pm)|$. Since these streams consist of $+1$ increments only we run the algorithm of Huang et. al. separately on each of them. At any time n, stream $|f'(B_j^\pm)|$ has seen $d_i^\pm(n)$ increments at site i, and lemma 2.1 of Huang et. al. guarantees that the estimates $\hat{d}_i^\pm(n)$ for the counts $d_i^\pm(n)$ are good.

Fact 2.3.3. *$E(\hat{d}_i^\pm) = d_i^\pm$ and $Var(\hat{d}_i^\pm) \leq 1/p^2$.*

This means that $E(\hat{d}^\pm) = \sum_i E(\hat{d}_i^\pm) = \sum_i d_i^\pm$, and therefore it also means that $E(\hat{d}) = \sum_i E(d_i^+ - d_i^-) = \sum_i d_i$. Since the estimators $\hat{d}_i^\pm$ are independent, the variance of the global drift is at most $2k/p^2$. By Chebyshev's inequality,

$$P(|\delta(n)| > \varepsilon 2^r k) \ \le \ \frac{2k/p^2}{(\varepsilon 2^r k)^2} \ < \ 1/3$$

Further, the expected cost of block B_j is at most

$$p|B_j| \le (3/\varepsilon 2^r k^{1/2})(2^r 2k) \le 30k^{1/2} v_j/\varepsilon$$

ensuring that only $O(\sqrt{k}v/\varepsilon)$ messages are used in total. □

2.4 Lower bounds

In this section we show that the dependence on v is essentially necessary by developing deterministic and randomized lower bounds on space+communication that hold even when v is small. Admittedly, this is not as pleasing as a pure communication lower bound would be. On the other hand, a distributed monitoring algorithm with high space complexity would be impractical for monitoring sensor data, network traffic patterns, and other applications of the model. Note that in terms of space+communication, our deterministic lower bound is tight up to factors of k, and our randomized lower bound is within a factor of $\log(n)$ of that.

For these lower bounds we use a slightly different problem. We call this problem the tracing problem. The streaming model for the tracing problem is the standard turnstile streaming model with updates $f'(n)$ arriving online. The problem is to maintain in small space a summary of the sequence f so that, at any current time n, if we are given an earlier time t as a query, we can return an estimate $\hat{f}(t)$ so that $P(|f(t) - \hat{f}(t)| \le \varepsilon f(t))$ is large (one in the deterministic case, 2/3 in the randomized case). We call this the tracing problem because our summary traces f through time, so that we can look up earlier values.

The reason for introducing this problem is that a space lower bound for the tracing problem implies a space+communication lower bound for the distributed tracking problem:

Lemma 2.4.1. *Fix some ε. Suppose that the tracing problem has an $\Omega(L_\varepsilon(n))$-bit space deterministic lower bound. Also suppose that there is a deterministic algorithm A for the distributed tracking problem that uses $C_\varepsilon(n)$ bits of communication and $S_\varepsilon(n)$ bits of space at the site and coordinator combined. Then we must have $C + S = \Omega(L)$.*

Further, if we replace "deterministic" with "randomized" in the preceding paragraph, the claim still holds.

Proof. Toward a contradiction, suppose that for all constants $c < 1$ and all n_0 there is an $n > n_0$ such that $C(n)+S(n) < cL(n)$. Then we can write an algorithm B for the tracing problem that uses $L'(n) < cL(n)$ bits of space: simulate A, recording all communication, and on a query t, play back the communication that occurred through time t.

At no point did we use the fact that A guarantees $P(|f(t)-\hat{f}(t)| \leq \varepsilon f(t)) = 1$, so the claim still holds if we change the correctness requirement to $P \geq 2/3$. $\square$

2.4.1 The deterministic bound

The deterministic lower bound that follows is similar in spirit to the lower bound of Tao et. al. [TYS10]. It uses a simple information-theoretic argument.

Theorem 2.4.2. *Let $\varepsilon = 1/m$ for some integer $m \geq 2$, let $n \geq 2m$, let $c < 1$ constant, and let $r \leq n^c$ and even. If a deterministic summary $S(f)$ guarantees, even only for sequences for which $v(n) = \frac{6m+9}{2m+6}\varepsilon r$, that $|f(t) - \hat{f}(t)| \leq \varepsilon f(t)$ for all $t \leq n$, then that summary must use $\Omega(\frac{\log n}{\varepsilon} v(n))$ bits of space.*

The full proof follows. At a high level, the sequences in the family take only values m or $m + 3$, and each sequence is defined by r of the n timesteps. If the new timestep t is one of the r chosen for our sequence, then we flip from m to $m+3$ or vice-versa. All of these sequences are unique and there are $2^{\Omega(r \log n)}$ of them.

Proof. We construct a family of input sequences of length n and variability $\frac{6m+9}{2m+6}\varepsilon r$. Choose sets of r different indices $1 \ldots n$ so that there are choose(n, r) such sets.

For each set S we define an input sequence f_S. We define $f_S(0) = m$ and the rest of f_S recursively: $f_S(t) = f_S(t–1)$ if t is not in S, and $f_S(t) = (2m+3) - f_S(t–1)$ if t is in S. (That is, switch between m and $m+3$.)

If A and B are two different sets, then $f_A \neq f_B$: let i be the smallest index that is in one and not the other; say i is in A. Then $f_A(1 \ldots (i-1)) = f_B(1 \ldots (i-1))$, but $f_A(i) \neq f_A(i-1) = f_B(i-1) = f_B(i)$.

The variability of any f_S is $\frac{6m+9}{2m+6}\varepsilon r$: There are $r/2$ changes from m to $m+3$ and another $r/2$ from $m+3$ to m. When we switch from m to $m+3$, we get a change in variability of $|f'(t)/f(t)| = 3/(m+3)$, and when we switch from $m+3$ to m, we get $|f'(t)/f(t)| = 3/m$. Thus $\sum_t |\frac{f'(t)}{f(t)}| = \frac{r}{2}\frac{6m+9}{m(m+3)} = \frac{6m+9}{2m+6}\varepsilon r$.

There are choose$(n, r) \geq (n/r)^r$ input sequences in our family, so to distinguish between any two input sequences we need at least $r \log(n/r) = \Omega(r \log n)$ bits. Any summary that can determine for each t the value $f(t)$ to within $\pm\varepsilon f(t)$, must also distinguish between $f(t) = m$ and $f(t) = m+3$, since there is no value within εm of m and also within $\varepsilon(m+3)$ of $m+3$. Since this summary must distinguish between $f(t) = m$ and $f(t) = m+3$ for all t, it must distinguish between any two input sequences in the family, and therefore needs $\Omega(r \log n)$ bits. □

2.4.2 The randomized bound

We use a construction similar to the one in our deterministic lower bound to produce a randomized lower bound. In order to make the analysis simple we forego a single variability value for all sequences in our constructed family, but still maintain that they all have low variability. C is a universal constant to be defined later.

Theorem 2.4.3. *Choose $\varepsilon \le 1/2$, $v \ge 32400\varepsilon \ln C$, and $n > 3v/\varepsilon$. If a summary $S(f)$ guarantees that $P(|f(t) - \hat{f}(t)| \le \varepsilon f(t)) \ge 99/100$ for all $t \le n$, even only for sequences for which $v(n) \le v$, then that summary must use $\Omega(v/\varepsilon)$ bits of space.*

We prove this theorem in two lemmas. In the first lemma, we reduce the claim to a claim about the existence of a hard family of sequences. In the second lemma we show the existence of such a family.

First a couple of definitions. For any two sequences f and g define the *number of overlaps between f and g* to be the number of positions $1 \le t \le n$ for which $|f(t) - g(t)| \le \varepsilon \max\{f(t), g(t)\}$. Say that f and g *match* if they have at least $\frac{6}{10}n$ overlaps.

Lemma 2.4.4. *Let $\mathcal{F}$ be a family of sequences of length n and variabilities $\le v$ such that no two sequences in $\mathcal{F}$ match. If a summary $S(f)$ guarantees for all f in $\mathcal{F}$ that $P(|f(t) - \hat{f}(t)| \le \varepsilon f(t)) \ge 99/100$ for all $t \le n$, then that summary must use $\Omega(\log |\mathcal{F}|)$ bits of space.*

The full proof follows. At a high level, if $S(f)$ is the summary for a sequence f, we can use it to generate an approximation $\hat{f}$ that at least 90% of the time overlaps with f in at least $\frac{9}{10}n$ positions. Since no two sequences in $\mathcal{F}$ overlap in more than $\frac{6}{10}n$ positions, at least 90% of the time we can determine f given $\hat{f}$. We then solve the one-way Index_N problem by deterministically generating $\mathcal{F}$ and sending a summary $S(f(x))$, where x is Alice's input string of size $N = \log_2 |\mathcal{F}|$, and $f(x)$ is the xth sequence in $\mathcal{F}$.

Proof. Let $S(f)$ be the summary for a sequence f, and sample $\hat{f}(1) \dots \hat{f}(n)$ once each using $S(f)$ to get $\hat{f}$. We want $\hat{f}(t)$ to be a good approximation for most of the timesteps. Let A be the event that $|\{t : |f(t)-\hat{f}(t)| \leq \varepsilon f(t)\}| \geq \frac{90}{100}n$. By Markov's inequality and the guarantee in the premise, we must have $P(A) \geq 9/10$.

Let ω define the random bits used in constructing $S(f)$ and in sampling $\hat{f}$. For any choice ω in A we have that $\hat{f}$ overlaps with f in at least $\frac{9}{10}n$ positions, which means that $\hat{f}$ overlaps with any other $g \in \mathcal{F}$ in at most $\frac{7}{10}n$ positions: at most the $\frac{6}{10}n$ in which f and g could overlap, plus the $\frac{1}{10}n$ in which $\hat{f}$ and f might not overlap.

Define $F \subseteq \mathcal{F}$ to be the sequences g that overlap with $\hat{f}$ in at least $\frac{9}{10}n$ positions. This means that when $\omega \in A$ we have $|F| = 1$, and therefore with probability at least 9/10 we can identify which sequence f had been used to construct $S(f)$.

We now prove our claim by reducing the Index_N problem to the problem of tracing the history of a sequence f. The following statement of Index_N is roughly as in Kushilevitz and Nisan [KN97]. There are two parties, Alice and Bob. Alice has an input string x of length $N = \log_2 |\mathcal{F}|$ and Bob has an input string i of length $\log_2 N$ that is interpreted as an index into x. Alice sends a message to Bob, and then Bob must output x_i correctly with probability at least 9/10.

Consider the following algorithm for solving Index_N. Alice deterministically generates a family $\mathcal{F}$ of sequences of length n and variabilities $\leq v$ such that no two match, by iterating over all possible sequences and choosing each next one that doesn't match any already chosen. Her $\log_2 |\mathcal{F}|$ bits of input x index a sequence f in $\mathcal{F}$. Alice computes a summary $S(f)$ and sends it to Bob. After receiving $S(f)$, Bob computes $\hat{f}(t)$ for every $t = 1 \ldots n$, to get a sequence $\hat{f}$. He then generates $\mathcal{F}$ himself and creates a set F of all sequences in $\mathcal{F}$ that overlap with $\hat{f}$ in at least $\frac{9}{10}n$ positions. If $F = \{f\}$, which it is with probability at least 9/10, then Bob can infer every bit of x.

Since the Index$_N$ problem is known to have a one-way communication complexity of $\Omega(N)$, it must be that $|S(f)| = \Omega(\log |\mathcal{F}|)$. $\square$

Lemma 2.4.5. *For all $\varepsilon \leq 1/2$, $v \geq 32400\varepsilon \ln C$, and $n > 3v/\varepsilon$, there is a family $\mathcal{F}$ of size $e^{\Omega(v/\varepsilon)}$ of sequences of size n such that no two sequences match and every sequence has variability at most v.*

The full proof follows. At a high level, sequences again switch between $m = 1/\varepsilon$ and $m{+}3$, except that these switches are chosen independently. We model the overlap with a Markov chain; the overlap between any two sequences is the sum over times t of a function y applied to the states of a chain modeling their interaction. We then apply a result of Chung, Lam, Liu, and Mitzenmacher [CLL12] to show that the probability that any two sequences match is low. Lastly, we show that not too many sequences have variability more than v, by proving that they usually don't switch between m and $m{+}3$ many times.

Proof. We construct $\mathcal{F}$ so that each of the two items holds (separately) with probability at least 4/5. Let $m = 1/\varepsilon$. To construct one sequence in $\mathcal{F}$, first define $f(0) = m$ with probability 1/2, else $f(0) = m{+}3$. Then, for $t = 1 \ldots n$: define $f(t) = (2m{+}3) - f(t{-}1)$ with probability $p = v/6\varepsilon n$, else $f(t) = f(t{-}1)$. That is, switch from m to $m{+}3$ (or vice-versa) with probability $p = v/6\varepsilon n$.

We first prove that the probability is at most 1/5 that any two sequences f and g match. We have that $P(f(0){=}g(0)) = 1/2$. If at any point in time we have $f(t) = g(t)$, then

$$P(f(t{+}1){=}g(t{+}1)) = \alpha = 1 - 2p(1{-}p)$$
$$\text{and } P(f(t{+}1){\neq}g(t{+}1)) = 1 - \alpha = 2p(1{-}p)$$

Similarly, if $f(t) \neq g(t)$, then

$$P(f(t{+}1){=}g(t{+}1)) = 1 - \alpha$$
$$\text{and } P(f(t{+}1){\neq}g(t{+}1)) = \alpha$$

The overlap between f and g is the number of times t that $f(t) = g(t)$. We model this situation with a Markov chain M with two states, c for "same" (that is, $f = g$) and d for "different" ($f \neq g$). Let s_t be the state after t steps, and let $p_t = (p_t(c), p_t(d))$ be the probabilities that M is in state c and d after step t. The stationary distribution $\pi = (1/2, 1/2)$, which also happens to be our initial distribution. We can model the overlap between f and g by defining a function $y(s_t) = 1$ if $s_t = c$ and $y(s_t) = 0$ otherwise; then $Y = \sum_{t=1}^{n} y(s_t)$ is the overlap between f and g. The expected value $E(y(\pi))$ of y evaluated on π is $1/2$.

The $(1/8)$-mixing time T is defined as the smallest time T such that we have $\frac{1}{2}||M^t r_0 - \pi||_1 \leq 1/8$ over all initial distributions r_0. Let r_0 be any initial distribution and $r_t = M^t r_0$. If we define $\Delta_t = r_t(c) - \pi(c)$, then $\Delta_t = (2\alpha{-}1)^t \Delta_0$. We can similarly bound $|r_t(d) - \pi(d)|$, so we can bound

$$T \;\leq\; \frac{\ln(8)}{\ln(1/(2\alpha{-}1))} \;\leq\; \frac{3}{(1-(2\alpha{-}1))} \;\leq\; \frac{3}{2p(1{-}p)} \;\leq\; \frac{3}{2p} \;=\; \frac{9\varepsilon n}{v}$$

since $1 - p \geq 1/2$ and since $1/\ln(1/x) \leq 1/(1{-}x)$ for x in $(0, 1)$. With this information we can now apply a sledgehammer of a result by Chung, Lam, Liu, and Mitzenmacher [CLL12]. Our fact 2.4.6 is their theorem 3.1, specialized a bit to our situation:

Fact 2.4.6. *Let M be an ergodic Markov chain with state space S. Let T be its $(1/8)$-mixing time. Let $(s_1, \ldots, s_n)$ denote an n-step random walk on M starting from its stationary distribution π. Let y be a weight function having $E(y(\pi)) = \mu$. Define the total weight of the walk by $Y = \sum_{t=1}^{n} y(s_t)$. Then there exists some* *universal constant C such that $P(Y \geq (1+\delta)\mu n) \leq C\exp(-\delta^2\mu n/72T)$ when $0 < \delta < 1$.*

Specifically, this means that $P(Y \geq \frac{6}{10}n) \leq C\exp(-v/(25 \cdot 72 \cdot 9 \cdot \varepsilon))$. Since v is large enough, we can also write $P \leq \exp(-v/32400\varepsilon)$. If we choose $|\mathcal{F}| = \frac{1}{5}\exp(v/(2 \cdot 32400\varepsilon))$, then by the union bound, with probability at least 4/5, no pair of sequences f, g matches.

We also must prove that there are enough sequences with variability at most v. The change in variability due to a single switch from m to $m+3$ (or vice-versa) is at most $3/m = 3\varepsilon$. For any sequence f, let $U_t = 1$ if f switched at time t, else $U_t = 0$. The expected number of switches is $v/6\varepsilon$; using a standard Chernoff bound, $P(\sum_t U_t \geq 2v/6\varepsilon) \leq \exp(-v/18\varepsilon) \leq 1/10$. Suppose we sample N sequences and B of them have more than $2v/6\varepsilon$ switches. In expectation there are at most $E(B) \leq \frac{1}{10}N$ that have too many switches. By Markov's inequality, $P(B \geq N/2) \leq 1/5$, so we can toss out the $\leq N/2$ bad sequences. This gives us a final size of $\mathcal{F}$ of $\frac{1}{10}\exp(v/(2 \cdot 32400\varepsilon))$. $\square$

2.5 Variability as a framework

In section 2.2 we proposed the f-variability $\sum_{t=1}^{n} \min\{1, |\frac{f'(t)}{f(t)}|\}$ as a way to analyze algorithms for the continuous monitoring problem (k, f, ε) over general update streams. However, our discussion so far has focused on distributed counting. In this final section we revisit the suitability of our definition by mentioning extensions to tracking other functions of a dataset defined by a distributed update stream.

2.5.1 Tracking item frequencies

We can extend our deterministic algorithm of section 2.3 to the problem of tracking item frequencies, in a manner similar to that in which Yi and Zhang [YZ09, YZ13] extend the ideas of Cormode et. al. [CMY08] to this problem.

Problem definition The problem of tracking item frequencies is only slightly different than the counting problem we've considered so far. In this problem there is a universe U of items and we maintain a dataset $D(t)$ that changes over time. At each new timestep n, either some item ℓ from U is added to D, or some item ℓ from D is removed. This update is told to a single site i; that is, site $i(n)$ receives an update $f'_\ell(n) = \pm 1$.

The frequency $f_\ell(t)$ of item ℓ at time t is the number of copies of ℓ that appear in $D(t)$. The first frequency moment $F_1(t)$ at time t is the total number of items $|D(t)|$. The problem is to maintain estimates $\hat{f}_\ell(n)$ at the coordinator so that for all times n and all items ℓ we have that $P(|f_\ell(n) - \hat{f}_\ell(n)| \leq \varepsilon F_1(n))$ is large.

Since in this problem we are tracking each item frequency to $\varepsilon F_1(n)$, we use F_1-variability instead, defining $v'(t) = \min\{1, 1/F_1(t)\}$.

2.5.2 Item frequencies with low communication

We first partition time into blocks as in section 2.3.1, using $f = F_1$. That is, at the end of each block we know the values n and $F_1(n)$ deterministically, and also that either $r = 0$ holds or that $F_1(n_j)$ is within a factor of two of $F_1(n_{j-1})$.

For tracking during blocks we modify the deterministic algorithm so that each site i holds counters $d_{i\ell}$ and $\delta_{i\ell}$ for every item ℓ. It also holds counters $f_{i\ell}$ of the total number of copies of ℓ seen at site i across all blocks.

At the end of each block, each site i reports all $f_{i\ell} \geq \varepsilon 2^r/3$ (using the new value of r). If site i reports counter $f_{i\ell}$ then it starts the next block with $d_{i\ell} = \delta_{i\ell} = 0$;

otherwise, $d_{i\ell}$ is updated to $d_{i\ell} + \delta_{i\ell}$ and then $\delta_{i\ell}$ is reset to zero. Within a block $r \geq 1$, the CONDITION is true when $\delta_{i\ell} \geq \varepsilon 2^r/3$.

The coordinator maintains estimates $\hat{f}_{i\ell}$ of $f_{i\ell}$ for each site i and item ℓ. Whenever the coordinator receives an update $\delta_{i\ell}$ during a block it updates its estimate $\hat{f}_{i\ell} = \hat{f}_{i\ell} + \delta_{i\ell}$.

Estimation error The total error in the estimate $\hat{f}_{i\ell}(n)$ at any time n is the error due to $d_{i\ell}$ plus the error due to $\delta_{i\ell}$. In both cases these quantities are bounded by $\varepsilon 2^r/3 \leq \varepsilon F_1(n)/3$.

Communication The total communication for a block is the total communicated within and at the end of the block. Within a block, all $\delta_{i\ell}$ start at zero, and there are at most $2^r k$ updates, so the total number of messages sent is $3k/\varepsilon$. At the end of a block, $f_{i\ell} \geq \varepsilon 2^r/3$ is true for at most $12k/\varepsilon$ counters $f_{i\ell}$. Therefore the total number of messages $O(\frac{k}{\varepsilon}v(n))$.

2.5.3 Item frequencies in small space+communication

The algorithm so far uses $|U|$ counters per site, which is prohibitive in terms of space. In [CM05] Cormode and Muthukrishnan show that in order to track over a non-distributed update stream each $f_\ell(n)$ so that for all ℓ and all times n we have $P(|f_\ell(n) - \hat{f}_\ell(n)| \leq \varepsilon F_1(n)/3) \geq 8/9$, it suffices to randomly partition each item in U into one of $27/\varepsilon$ classes using a pairwise-independent hash function h, and to estimate $f_\ell(n)$ as $f_{h(\ell)}(n)$. The $27/\varepsilon$ counters and the hash function h together form their Count-Min Sketch [CM05].

Similarly, in [GM06, GM07] Ganguly and Majumder adapt a data structure of Gasieniec and Muthukrishnan [Mut05], which they call the CR-precis, to deterministically track each $f_\ell(n)$ to $\varepsilon F_1(n)/3$ error. This data structure uses $\frac{3}{\varepsilon}$ rows of $\frac{6 \log |U|}{\varepsilon \log 1/\varepsilon}$ counters, and estimates $f_\ell(n)$ as the average over rows r of $f_{h(r,\ell)}(n)$.

(Ganguly and Majumder actually take the minimum over the rows r, but the average works too and yields a linear sketch.)

In either case, we can first reduce our set of items ℓ to a small number of counters c, and instead of tracking $f_{i\ell}$ we track f_{ic} for each counter c. The coordinator can then linearly combine its estimates $\hat{f}_{ic}$ to obtain estimates $\hat{f}_{i\ell}$ for each item ℓ. This introduces another $\varepsilon F_1(n)/3$ error, yielding algorithms that guarantee

- $P(|f_{i\ell}(n) - \hat{f}_{i\ell}(n)| \leq \varepsilon F_1(n)) = 1$ in $O(\frac{k \log |U|}{\varepsilon^2 \log 1/\varepsilon} v(n) \log n)$ bits of space + communication, and
- $P(|f_{i\ell}(n) - \hat{f}_{i\ell}(n)| \leq \varepsilon F_1(n)) \geq 8/9$ in $O(k \log |U| + \frac{k}{\varepsilon} v(n) \log n)$ bits of space + communication.

2.5.4 Remarks

We obtain a randomized communication bound of $O(\frac{k}{\varepsilon} v(n))$ messages, but it might be possible to do better. In [HYZ12] Huang et. al. both develop a randomized counting algorithm ($O(\frac{\sqrt{k}}{\varepsilon} \log n)$ messages) and also extend it to the problem of tracking item frequencies to get the same communication bound. Unfortunately, their algorithm appears to require the total variance in their estimate at any time $t < n$ to be bounded by a constant factor of the variance at time n. This is only guaranteed to be true when item deletions are not permitted (and F_1 grows monotonically). We avoid this problem in section 2.3.4 for tracking $f = F_1$ by deterministically updating F_1 at the end of each block. For this problem, though, deterministically updating all of the large $\hat{f}_{i\ell}$ at the end of each block could incur $O(k/\varepsilon)$ messages.

2.5.5 Aggregate functions with one site

In this subsection we consider general single-integer-valued functions f of a dataset. When there is a single site, the site always knows the exact value of $f(n)$, and the only issue is updating the coordinator to have an approximation $\hat{f}(n)$ so that $|f(n) - \hat{f}(n)| \leq \varepsilon f(n)$ for all n. We can show that this problem of tracking f to ε relative error when $k = 1$ has an $O(\frac{1}{\varepsilon}v(n))$-word upper bound, where here $v(n)$ is the f-variability. The algorithm is: whenever $|f - \hat{f}| > \varepsilon f$, send f to the coordinator.

Proof. If $f(n) = 0$ then $v'(n) = 1$. Also, if $f(n)$ changes sign from $f(n-1)$, then $v'(n) = 1$. So consider intervals over which $f(n)$ is nonzero and doesn't change sign. Over such an interval, let $\Phi(n) = |\frac{f(n)-\hat{f}(n)}{f(n)}|$. If at time n we update $\hat{f}$ then $\Phi(n) = 0$. Otherwise,

$$\begin{aligned}
\Phi(n) &= \frac{|f(n-1) - \hat{f}(n-1) + f'(n)|}{|f(n)|} \leq \frac{|f(n-1) - \hat{f}(n-1)|}{|f(n)|} + \frac{|f'(n)|}{|f(n)|} \\
&= \frac{|f(n-1)|}{|f(n)|}\Phi(n-1) + \frac{|f'(n)|}{|f(n)|} \leq \frac{|f(n)| + |f'(n)|}{|f(n)|}\Phi(n-1) + \frac{|f'(n)|}{|f(n)|} \\
&\leq \Phi(n-1) + \frac{(1+\Phi(n-1))|f'(n)|}{|f(n)|}
\end{aligned}$$

Since $\Phi(n) \leq \varepsilon$ we have $|\Phi'(n)| \leq (1+\varepsilon)|\frac{f'(n)}{f(n)}|$. We only send a message each time that Φ would be more than ε, so the total number of messages sent is at most the total increase in Φ, which is $\sum_{t=1}^{n} \min\{1, |\frac{f'(t)}{f(t)}|\}$. □

Along with our lower bounds of section 2.4, this upper bound lends evidence to our claim that variability captures the difficulty of communicating changes in general functions f that are due to the non-monotonicity of the input stream.

References

[ABC09] Chrisil Arackaparambil, Joshua Brody, and Amit Chakrabarti. "Functional monitoring without monotonicity." In *Automata, Languages and Programming*, pp. 95–106. Springer, 2009.

[ACH12] Pankaj K. Agarwal, Graham Cormode, Zengfeng Huang, Jeff Phillips, Zhewei Wei, and Ke Yi. "Mergeable summaries." In *Proceedings of the 31st symposium on Principles of Database Systems*, PODS '12, pp. 23–34, New York, NY, USA, 2012. ACM.

[ACH13] Pankaj K Agarwal, Graham Cormode, Zengfeng Huang, Jeff M Phillips, Zhewei Wei, and Ke Yi. "Mergeable summaries." *ACM Transactions on Database Systems (TODS)*, **38**(4):26, 2013.

[CLL12] Kai-Min Chung, Henry Lam, Zhenming Liu, and Michael Mitzenmacher. "Chernoff-hoeffding bounds for markov chains: Generalized and simplified." *arXiv preprint arXiv:1201.0559*, 2012.

[CM05] Graham Cormode and S. Muthukrishnan. "An improved data stream summary: the count-min sketch and its applications." *Journal of Algorithms*, **55**(1):58 – 75, 2005.

[CMY08] Graham Cormode, S. Muthukrishnan, and Ke Yi. "Algorithms for Distributed Functional Monitoring." In *Proceedings of the Nineteenth Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA '08, pp. 1076–1085, Philadelphia, PA, USA, 2008. Society for Industrial and Applied Mathematics.

[CMY11] Graham Cormode, S Muthukrishnan, and Ke Yi. "Algorithms for distributed functional monitoring." *ACM Transactions on Algorithms (TALG)*, **7**(2):21, 2011.

[FO15a] David Felber and Rafail Ostrovsky. "A randomized online quantile summary in $O(\frac{1}{\varepsilon}\log\frac{1}{\varepsilon})$ words." *arXiv preprint arXiv:1503.01156*, 2015.

[FO15b] David Felber and Rafail Ostrovsky. "Variability in data streams." *arXiv preprint arXiv:1502.07027*, 2015.

[GK01] Michael Greenwald and Sanjeev Khanna. "Space-efficient online computation of quantile summaries." In *Proceedings of the 2001 ACM SIGMOD international conference on Management of data*, SIGMOD '01, pp. 58–66, New York, NY, USA, 2001. ACM.

[GM06] Sumit Ganguly and Anirban Majumder. "CR-precis: A deterministic summary structure for update data streams." *CoRR*, **abs/cs/0609032**, 2006.

[GM07] Sumit Ganguly and Anirban Majumder. "CR-precis: A Deterministic Summary Structure for Update Data Streams." In Bo Chen, Mike Paterson, and Guochuan Zhang, editors, *Combinatorics, Algorithms, Probabilistic and Experimental Methodologies*, volume 4614 of *Lecture Notes in Computer Science*, pp. 48–59. Springer Berlin Heidelberg, 2007.

[HT10] Regant Y.S. Hung and Hingfung F. Ting. "An $\Omega(\frac{1}{\varepsilon}\log\frac{1}{\varepsilon})$ Space Lower Bound for Finding ε-Approximate Quantiles in a Data Stream." In Der-Tsai Lee, DannyZ. Chen, and Shi Ying, editors, *Frontiers in Algorithmics*, volume 6213 of *Lecture Notes in Computer Science*, pp. 89–100. Springer Berlin Heidelberg, 2010.

[HYZ12] Zengfeng Huang, Ke Yi, and Qin Zhang. "Randomized algorithms for tracking distributed count, frequencies, and ranks." In *Proceedings of the 31st symposium on Principles of Database Systems*, pp. 295–306. ACM, 2012.

[KN97] Eyal Kushilevitz and Noam Nisan. *Communication Complexity*. Cambridge University Press, New York, NY, USA, 1997.

[LRV11] Zhenming Liu, Bozidar Radunović, and Milan Vojnović. "Continuous distributed counting for non-monotonic streams." In *Technical Report MSR-TR-2011-128*, 2011.

[LRV12] Zhenming Liu, Bozidar Radunović, and Milan Vojnović. "Continuous distributed counting for non-monotonic streams." In *Proceedings of the 31st symposium on Principles of Database Systems*, pp. 307–318. ACM, 2012.

[MP78] J. I. Munro and M. S. Paterson. "Selection and Sorting with Limited Storage." In *Proceedings of the 19th Annual Symposium on Foundations of Computer Science*, SFCS '78, pp. 253–258, Washington, DC, USA, 1978. IEEE Computer Society.

[MRL98] Gurmeet Singh Manku, Sridhar Rajagopalan, and Bruce G Lindsay. "Approximate medians and other quantiles in one pass and with limited memory." In *ACM SIGMOD Record*, volume 27, pp. 426–435. ACM, 1998.

[MRL99] Gurmeet Singh Manku, Sridhar Rajagopalan, and Bruce G Lindsay. "Random sampling techniques for space efficient online computation of order statistics of large datasets." In *ACM SIGMOD Record*, volume 28, pp. 251–262. ACM, 1999.

[Mut05] Shanmugavelayutham Muthukrishnan. *Data streams: Algorithms and applications*. Now Publishers Inc, 2005.

[SBA04] Nisheeth Shrivastava, Chiranjeeb Buragohain, Divyakant Agrawal, and Subhash Suri. "Medians and beyond: new aggregation techniques for sensor networks." In *Proceedings of the 2nd international conference on Embedded networked sensor systems*, SenSys '04, pp. 239–249, New York, NY, USA, 2004. ACM.

[TYS10] Yufei Tao, Ke Yi, Cheng Sheng, Jian Pei, and Feifei Li. "Logging every footstep: quantile summaries for the entire history." In *Proceedings of the 2010 ACM SIGMOD International Conference on Management of data*, SIGMOD '10, pp. 639–650, New York, NY, USA, 2010. ACM.

[VC71] Vladimir N Vapnik and A Ya Chervonenkis. "On the uniform convergence of relative frequencies of events to their probabilities." *Theory of Probability & Its Applications*, **16**(2):264–280, 1971.

[Vit85] Jeffrey S Vitter. "Random sampling with a reservoir." *ACM Transactions on Mathematical Software (TOMS)*, **11**(1):37–57, 1985.

[WLY13] Lu Wang, Ge Luo, Ke Yi, and Graham Cormode. "Quantiles over data streams: an experimental study." In *Proceedings of the 2013 ACM SIGMOD International Conference on Management of Data*, pp. 737–748. ACM, 2013.

[WZ11] David P. Woodruff and Qin Zhang. "Tight Bounds for Distributed Functional Monitoring." *CoRR*, **abs/1112.5153**, 2011.

[WZ12] David P Woodruff and Qin Zhang. "Tight bounds for distributed functional monitoring." In *Proceedings of the 44th symposium on Theory of Computing*, pp. 941–960. ACM, 2012.

[YZ09] Ke Yi and Qin Zhang. "Optimal Tracking of Distributed Heavy Hitters and Quantiles." In *Proceedings of the Twenty-eighth ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems*, PODS '09, pp. 167–174, New York, NY, USA, 2009. ACM.

[YZ13] Ke Yi and Qin Zhang. "Optimal Tracking of Distributed Heavy Hitters and Quantiles." *Algorithmica*, **65**(1):206–223, 2013.