Fast Moment Estimation in Data Streams in Optimal Space

Daniel M. Kane[*]
Harvard University,
Department of Mathematics.
dankane@math.harvard.edu

Jelani Nelson[†]
MIT Computer Science and Artificial Intelligence Laboratory
minilek@mit.edu

Ely Porat
Department of Computer Science, Bar-Ilan University
porately@macs.biu.ac.il

David P. Woodruff
IBM Research-Almaden
dpwoodru@us.ibm.com

ABSTRACT

We give a space-optimal streaming algorithm with update time $O(\log^2(1/\varepsilon)\log\log(1/\varepsilon))$ for approximating the pth frequency moment, $0 < p < 2$, of a length-n vector updated in a data stream up to a factor of $1 \pm \varepsilon$. This provides a nearly exponential improvement over the previous space-optimal algorithm of [Kane-Nelson-Woodruff, SODA 2010], which had update time $\Omega(1/\varepsilon^2)$.

When combined with the work of [Harvey-Nelson-Onak, FOCS 2008], we also obtain the first algorithm for entropy estimation in turnstile streams which simultaneously achieves near-optimal space and fast update time.

Categories and Subject Descriptors: F.2 Analysis of Algorithms and Problem Complexity: General

General Terms: Algorithms, Theory

Keywords: data stream algorithms, frequency moments

1. INTRODUCTION

The problem of estimating frequency moments in a data stream was first studied in the seminal work of Alon, Matias, and Szegedy [3] and has received much attention [6, 8, 29, 31, 36, 38, 41, 52, 53]. Estimation of the second moment has applications to estimating join and self-join sizes [2], numerical linear algebra [13, 47], and network anomaly detection [37, 49]. First moment estimation is useful in mining network traffic data [16], comparing empirical probability distributions [30], and several other applications (see [41] and the references therein). Estimating fractional moments in $(0, 2)$ has applications to entropy estimation [28, 54], mining tabular data [14], and image decomposition [22]. It was observed experimentally that using fractional moments in $(0, 1)$ can improve the effectiveness of standard clustering algorithms [1].

Of these applications of fractional moments, one of the most important is entropy. Entropy estimation is extensively studied in the streaming literature [7, 9, 10, 24, 26, 28, 39, 54]. It has become a useful tool in databases as a way of understanding database design, enabling data integration, and achieving data anonymization [48]. Estimating entropy on massive data sets is a key ingredient in performing this analysis (see the tutorial in [48] and the references therein). It is also useful for network anomaly detection [54].

Formally, in the moment estimation problem we are given a real number $p > 0$. There is an underlying n-dimensional vector x initialized to $\vec{0}$. What follows is a sequence of m updates of the form $(i_1, v_1), \ldots, (i_m, v_m) \in [n] \times \{-M, \ldots, M\}$ for some $M > 0$. An update (i, v) causes the change $x_i \leftarrow x_i + v$. We would like to compute $F_p \overset{\text{def}}{=} \|x\|_p^p \overset{\text{def}}{=} \sum_{i=1}^n |x_i|^p$, also called the p-th frequency moment of x. In many applications, it is required that the algorithm only use very limited space while processing the stream, e.g. in networking applications, where x may be indexed by source-destination IP pairs for which a router cannot afford to store the vector in memory, or in distributed settings where one wants a succinct "sketch" of a dataset which can be compared with other sketches for fast computation of similarity measures.

It is known [3] that linear space ($\Omega(\min\{n, m\})$ bits) is required unless one allows for (a) approximation, so that we are only guaranteed to output a value in $[(1-\varepsilon)F_p, (1+\varepsilon)F_p]$ for some $0 < \varepsilon < 1/2$, and (b) randomization, so that the output is correct only with some probability bounded away from 1, over the algorithm's randomness. Polynomial space is required for $p > 2$ [6, 11, 23, 32, 46], while the space complexity for $0 < p \leq 2$ is only $\Theta(\varepsilon^{-2}\log(mM)+\log\log(n))$ bits to achieve success probability $2/3$ [3, 36], which can be amplified to $1 - \delta$ by outputting the median of $O(\log(1/\delta))$ independent repetitions. This is space-optimal in terms of δ as well when n, M, and m are polynomially related [34]. In this work, we focus on this "feasible" regime for p, $0 < p \leq 2$, for which logarithmic space is achievable.

While there has been much previous work on minimizing the space consumption in streaming algorithms, recently researchers have begun to work toward minimizing *update time* [44, Question 1], i.e., the time taken to process a new update in the stream. For example, in network traffic monitoring applications each packet is an update, and so it is important that a streaming algorithm processing the packet stream be able to operate at network speeds (see for example the ap-

[*]Supported by an NSF Graduate Research Fellowship.

[†]Supported by a Xerox-MIT Fellowship, and in part by the Center for Massive Data Algorithmics (MADALGO) - a center of the Danish National Research Foundation.

plications in [37, 49]). Note that if an algorithm has update time say, $\Omega(1/\varepsilon^2)$, then achieving a small error parameter such as $\varepsilon = .01$ could be intractable since this time is multiplied by the length of the stream. This is true even if the space required of the algorithm is small enough to fit in memory.

For $p = 2$, optimal space and $O(1)$ update time are simultaneously achievable [12, 49], improving upon the original F_2 algorithm of [3]. For $p = 1$ near-optimal, but not quite optimal, space and $O(\log(n/\varepsilon))$ update time are achievable [41]. Optimal (or even near-optimal) space for other $p \in (0, 2]$ is only known to be achievable with $\mathrm{poly}(1/\varepsilon)$ update time [36].

Our Results: For all $0 < p < 2$ and $0 < \varepsilon < 1/2$ we give an algorithm for $(1\pm\varepsilon)$-approximating F_p with success probability at least 2/3 which uses an optimal $O(\varepsilon^{-2}\log(mM) + \log\log n)$ bits of space with $O(\log^2(1/\varepsilon)\log\log(1/\varepsilon))$ update time.[1] This is a nearly exponential improvement in the time complexity of the previous space-optimal algorithm for every such p.

Also, when combined with work in [28], our algorithm implies the first entropy estimation algorithm in the turnstile model that simultaneously has a fast $\mathrm{polylog}(\log(mM)/\varepsilon)$ update time and uses only $O(1/\varepsilon^2)\cdot\mathrm{polylog}(nmM/\varepsilon)$ space, both for additive and multiplicative approximation. This dependence on ε in our space bound is optimal up to $\log(1/\varepsilon)$ factors. We also obtain these results for additive approximation of a number of other statistical quantities, such as conditional entropy, mutual information, Jensen-Shannon divergence, and entropy norm, since they can be written as sums and differences of entropies and ℓ_1 norms.

1.1 Previous Work

The complexity of streaming algorithms for moment estimation has a long history; see Figure 1.

Alon, Matias, and Szegedy gave a space-optimal algorithm for $p = 2$. The update time was later brought down to an optimal $O(1)$ implicitly in [12] and explicitly in [49]. The work of [18] gave a space-optimal algorithm for $p = 1$, but under the restriction that each coordinate is updated at most twice, once positively and once negatively. Indyk [29] later removed this restriction, and also gave an algorithm handling all $0 < p < 2$, but at the expense of increasing the space by a $\log n$ factor. Li later [38] provided alternative estimators for all $0 < p < 2$, based on Indyk's sketches. The extra $\log n$ factor in the space of these algorithms was later removed in [36], yielding optimal space. The algorithms of [18, 29, 36, 38] all required $\mathrm{poly}(1/\varepsilon)$ update time. The work [42] gave an algorithm for $p = 1$ in the restricted setting where each coordinate is updated at most twice, as in [18], with space suboptimal by a $\log(1/\varepsilon)$ factor, and with update time $\log^2(mM)$. The work of [41] provides an algorithm for $p = 1$ with unrestricted updates which uses suboptimal space by a $\log n$ factor, but has update time only $O(\log(n/\varepsilon))$.

The empirical entropy of a data stream is

$$H(x) = -\sum_i p_i \log(p_i)$$

where $p_i = |x_i|/\|x\|_1$. In the insertion-only model (all updates (i, v) have $v > 0$), [9] gives an algorithm with space $O(\varepsilon^{-2}\log^2(m))$ and expected update time $O(\log((\log m)/\varepsilon))$ for sufficiently long streams when $M = 1$. The insertion-only model is restrictive, e.g., it does not allow for two sites to compute the empirical entropy of the difference of their datasets. In the turnstile model, [28] gives an algorithm with $O(1/\varepsilon^2)\cdot\mathrm{polylog}(nmM/\varepsilon)$ space and update time for both multiplicative-$(1 + \varepsilon)$ and additive-ε approximation. In the strict turnstile model ($\forall i\ x_i > 0$ at each point in the stream), Li [39] gives a moment estimation algorithm with small complexity for p near 1, implying a simpler entropy estimation algorithm with better $\log(mM/\varepsilon)$ factors in space complexity. Previously, the only algorithm known even in the strict turnstile model with fast update time required $\Omega(1/\varepsilon^3)$ space [7], and its update time was $\mathrm{polylog}(mM/\varepsilon)$. Several important statistics related to entropy are the entropy norm and Jensen-Shannon divergence. These statistics are useful in pattern matching and statistical learning. Previous algorithms [10, 25, 26] for these problems require $\Omega(\varepsilon^{-2})$ update time. We achieve $\mathrm{poly}(\log\log(mM), \log 1/\varepsilon)$ time, an almost exponential speedup for a wide range of ε.

On the lower bound front, a lower bound of

$$\Omega(\min\{n, m, \varepsilon^{-2}\log(\varepsilon^2 mM)\} + \log\log(nmM))$$

was shown in [36] for F_p-estimation, together with an upper bound of $O(\varepsilon^{-2}\log(mM)+\log\log n)$ bits. For nearly the full range of parameters these are tight, since if $\varepsilon \le 1/\sqrt{m}$ we can store the entire stream in memory in $O(m\log(nM)) = O(\varepsilon^{-2}\log(nM))$ bits of space (and we can ensure $n = O(m^2)$ via FKS hashing [19] with just an additive $O(\log\log n)$ bits increase in space), and if $\varepsilon \le 1/\sqrt{n}$ we can store the entire vector in memory in $O(n\log(mM)) = O(\varepsilon^{-2}\log(mM))$ bits. Thus, a gap exists only when ε is very near $1/\sqrt{\min\{n, m\}}$. This lower bound followed many previous lower bounds for this problem, given in [3, 5, 33, 52, 53]. For entropy estimation, there is a lower bound of $\Omega(\varepsilon^{-2}\log\min\{n, m\}/\log(1/\varepsilon))$ [9, 36]. These lower bounds for both moment and entropy estimation increase by a $\log(1/\delta)$ factor when success probability $1 - \delta$ is desired [34].

1.2 Overview of our approach

The starting point of our algorithm is an approach set forth by [41] for F_1-estimation. In that work, the coordinates $i \in \{1, \ldots, n\}$ were partitioned into *heavy hitters*, and *light* coordinates. A *ϕ-heavy hitter* with respect to F_p is a coordinate i such that $|x_i|^p \ge \phi\|x\|_p^p$. A list L of ε^2-heavy hitters with respect to F_1 were found by running the CountMin sketch of [15].

To estimate the contribution of the light elements to F_1, the work of [41] used $R = \Theta(1/\varepsilon^2)$ independent Cauchy sketches $D_1, \ldots, D_R$ (actually, D_j was a tuple of 3 independent Cauchy sketches). A *Cauchy sketch* of a vector x, introduced by Indyk [29], is the dot product of x with a random vector z with independent entries distributed according to the Cauchy distribution. This distribution has the property that $\langle z, x\rangle$ is itself a Cauchy random variable, scaled by $\|x\|_1$. Upon receiving an update to x_i in the stream, the update was fed to $D_{h(i)}$ for some hash function $h : [n] \to [R]$. At the end of the stream, the estimate of the contribution to F_1 from light elements was $(R/(R - |h(L)|)) \cdot \sum_{j\notin h(L)} \mathsf{EstLi}_1(D_j)$, where EstLi_p is Li's geometric mean estimator for F_p [38]. The analysis of [41] only used that Li's estimator is unbiased and has good variance.

Our algorithm LightEstimator for estimating the contri-

[1] Throughout this document we say $g = \tilde{O}(f)$ if $g = O(f \cdot \mathrm{polylog}(f))$. Similarly, $g = \tilde{\Omega}(f)$ if $g = \Omega(f/\mathrm{polylog}(f))$.

Paper	Space	Update Time	Model	Which p
[3]	$O(\varepsilon^{-2}\log(mM))$	$O(\varepsilon^{-2})$	unrestricted updates	$p=2$
[12, 49]	$O(\varepsilon^{-2}\log(mM))$	$O(1)$	unrestricted updates	$p=2$
[18]	$O(\varepsilon^{-2}\log(mM))$	$O(\varepsilon^{-2})$	at most 2 updates per coordinate	$p=1$
[29, 38]	$O(\varepsilon^{-2}\log(n)\log(mM))$	$O(\varepsilon^{-2})$	unrestricted updates	$p\in(0,2)$
[36]	$O(\varepsilon^{-2}\log(mM))$	$O(\varepsilon^{-2})$	unrestricted updates	$p\in(0,2)$
[42]	$O(\varepsilon^{-2}\log(mM)\log(1/\varepsilon))$	$O(\log^2(mM))$	at most 2 updates per coordinate	$p=1$
[41]	$O(\varepsilon^{-2}\log(n)\log(mM))$	$O(\log(n/\varepsilon))$	unrestricted updates	$p=1$
this work	$O(\varepsilon^{-2}\log(mM))$	$\tilde{O}(\log^2(1/\varepsilon))$	unrestricted updates	$p\in(0,2)$

Figure 1: Comparison of our contribution to previous works on F_p-estimation in data streams. All space bounds hide an additive $O(\log\log n)$ term.

bution to F_p from light coordinates for $p \neq 1$ follows the same approach. Our contribution here is to show that a variant of Li's geometric mean estimator has bounded variance and is approximately unbiased (to within relative error ε) even when the associated p-stable random variables are only k-wise independent for $k = \Omega(1/\varepsilon^p)$. This variant allows us to avoid Nisan's pseudorandom generator [43] and thus achieve optimal space. While the work of [36] also provided an estimator avoiding Nisan's pseudorandom generator, their estimator is not known to be approximately unbiased, which makes it less useful in applications involving the average of many such estimators. We evaluate the necessary k-wise independent hash function quickly by a combination of buffering and fast multipoint evaluation of a collection of pairwise independent polynomials. Our proof that bounded independence suffices uses the FT-mollification approach introduced in [36] and refined in [17], which is a method for showing that the expectation of some function is approximately preserved by bounded independence, via a smoothing operation (FT-mollification) and Taylor's theorem. While [17, 36] only dealt with FT-mollifying indicator functions of regions in Euclidean space, here we must FT-mollify functions of the form $f(x) = |x|^{1/t}$. We express $\mathbf{E}[f(x)] = \int_0^\infty f(x)\varphi_p(x)dx$ as $\int_0^\infty f'(x)(1-\Phi_p(x))dx$ via integration by parts, where φ_p is the density function of the absolute value of the p-stable distribution, and Φ_p is the corresponding cumulative distribution function. We then note $1-\Phi_p(x) = \mathbf{Pr}[|X| \geq x] = \mathbf{E}[I_{[x,\infty)\cup(-\infty,-x]}(X)]$ for X p-stable, where I_S is the indicator function of the set S, so that we can write $\mathbf{E}[f(x)]$ as a weighted integral of indicator functions. We then FT-mollify $I_{[x,\infty)\cup(-\infty,-x]}$, which *is* the indicator function of some set, at which point we can apply the methods of [17, 36].

In order to estimate the contribution to F_p from coordinates in L, we develop a novel data structure we refer to as HighEnd. Suppose L contains all the α-heavy hitters, and every index in L is an $(\alpha/2)$-heavy hitter. We would like to compute $\|x_L\|_p^p \pm O(\varepsilon)\cdot\|x\|_p^p$, where $\alpha = \Omega(\varepsilon^2)$. We maintain a matrix of counters $D_{j,k}$ for $(j,k) \in [t]\times[s]$ for $t = O(\log(1/\varepsilon))$ and $s = O(1/\alpha)$. For each $j \in [t]$ we have a hash function $h^j : [n] \to [s]$ and $g^j : [n] \to [r]$ for $r = O(\log(1/\varepsilon))$. The counter $D_{j,k}$ then stores $\sum_{h^j(v)=k} e^{2\pi i g^j(v)/r} x_v$ for $i = \sqrt{-1}$. That is, our data structure is similar to the CountSketch data structure of [12], but rather than taking the dot product with a random sign vector in each counter, we take the dot product with a vector whose entries are random complex roots of unity. Also, we require higher independence. At the end of the stream, our estimate of the F_p-contribution from the heavy hitters is the real part of

$$\sum_{w\in L}\left(\frac{3}{t}\sum_{k=1}^{t/3} e^{-2\pi i g^{j(w,k)}(w)/r}\cdot \mathrm{sign}(x_w)\cdot D_{j(w,k),h^{j(w,k)}(w)}\right)^p,$$

where $j(w,k)$ denotes the kth smallest value $b \in [t]$ such that h^b isolates w from the other $w' \in L$ (if fewer than $t/3$ such b exist, then we fail).

The choice to use sums of complex roots of unity is to ensure our estimator is approximately unbiased, since the real part of large powers of roots of unity is 0 in expectation. This is the advantage we achieve over sums of sign variables, which, although their tail bounds are similar, if used here would result in a constant factor bias in the expectation that we do not know how to remove. This is due to the non-linearity of the ℓ_p-norm for $p \neq 1$, which is why we moved away from the sign vectors of [41]. Our analysis also improves the previous space bound for $p = 1$.

Muthukrishnan had asked whether one could find streaming applications which make good use of complex random variables [40, Problem 5]. As far as we are aware, this is the first such algorithm. We believe the techniques we develop here to analyze complex random variables are of independent interest. For related problems, e.g., estimation of F_p for $p > 2$, using complex roots of unity leads to sub-optimal bounds [20]. Our intuition was that an algorithm using p-stable random variables would be necessary to estimate the contribution to F_p from the heavy hitters. However, such approaches generally suffer from large variance.

In parallel we must run an algorithm we develop to find the heavy hitters. Unfortunately, this algorithm, as well as HighEnd, uses suboptimal space. To overcome this, we actually use a list of ϵ^2-heavy hitters for $\epsilon = \varepsilon\cdot\log(1/\varepsilon)$. This then improves the space, at the expense of increasing the variance of LightEstimator. We then run $O((\epsilon/\varepsilon)^2)$ pairwise independent instantiations of LightEstimator in parallel and take the average estimate, to bring the variance down. This increases some part of the update time of LightEstimator by a $\log^2(1/\varepsilon)$ factor, but this term turns out to anyway be dominated by the time to evaluate various hash functions. Though, even in the extreme case of balancing with $\epsilon = 1$, our algorithm for finding the heavy hitters requires a sub-optimal $\Omega(\log(n)\log(mM))$ bits of space. In fact, this is a lower bound for finding the heavy hitters [4]. We bypass this lower bound by avoiding learning the actual identities of the heavy hitters. Rather, we perform a dimensionality reduction down to dimension poly$(1/\varepsilon)$ that injectively maps heavy hitters, while approximately preserving their weights, and does not introduce spurious ones. We work in this smaller universe. We do this with very limited independence, and standard analyses are not strong enough for the guarantees we need. We then apply HighEnd to estimate the contribution from heavy hitters in this new vector, and show the correctness of our overall algorithm is maintained.

For entropy estimation, [28] gave a reduction from additive estimation of entropy to multiplicative-error estimation of F_p for p near 1. They also gave a reduction from multiplicative estimation of entropy to multiplicative-error estimation of F_p and F_p^{res} for p near and equal to 1, where F_p^{res} is the pth moment of the vector x after removing the heaviest entry of x. The details of how our algorithm fits into their reduction are fairly straightforward, so we defer them to Section A.

1.3 Notation

For a positive integer r, we use $[r]$ to denote the set $\{1, \ldots, r\}$. All logarithms are base-2 unless otherwise noted. For a complex number z, $\mathbf{Re}[z]$ is the real part of z, $\mathbf{Im}[z]$ is the imaginary part of z, $\bar{z}$ is the complex conjugate of z, and $|z| \overset{\text{def}}{=} \sqrt{\bar{z}z}$. At times we consider random variables X taking on *complex* values. For such random variables, we use $\mathbf{Var}[X]$ to denote $\mathbf{E}[|X - \mathbf{E}[X]|^2]$. Note that the usual statement of Chebyshev's inequality still holds under this definition.

For $x \in \mathbb{R}^n$ and $S \subseteq [n]$, x_S denotes the n-dimensional vector whose ith coordinate is x_i for $i \in S$ and 0 otherwise. For a probabilistic event $\mathcal{E}$, we use $\mathbf{1}_{\mathcal{E}}$ to denote the indicator random variable for $\mathcal{E}$. We sometimes refer to a constant as *universal* if it does not depend on other parameters, such as n, m, ε, etc. All space bounds are measured in bits. When measuring time complexity, we assume a word RAM with machine word size $\Omega(\log(nmM))$ so that standard arithmetic and bitwise operations can be performed on words in constant time. We use *update time* to refer to the time taken to process an update in the data stream. We use *reporting time* to refer to the time taken to answer some query (e.g., "output an estimate of F_p").

Also, we can assume $n = O(m^2)$ by FKS hashing [19] with an additive $O(\log \log n)$ term in our final space bound; see Section A.1.1 of [36] for details. Henceforth any terms involving n appearing in space and time bounds may be assumed at most m^2. We also often assume that n, m, M, ε, and δ are powers of 2 (or sometimes 4), and that $1/\sqrt{n} < \varepsilon < \varepsilon_0$ for some universal constant $\varepsilon_0 > 0$. These assumptions are without loss of generality. We can assume $\varepsilon > 1/\sqrt{n}$ since otherwise we could store x explicitly in memory using $O(n \log(mM)) = O(\varepsilon^{-2} \log(mM))$ bits with constant update and reporting times. Finally, we assume $\|x\|_p^p \geq 1$. This is because x has integer entries, and so either $\|x\|_p^p \geq 1$ or it is 0. The case that it is 0 only occurs when x is the 0 vector, which can be detected in $O(\log(nmM))$ space by the AMS sketch [3].

1.4 Organization

In Section 2, we give an efficient subroutine HighEnd for estimating $\|x_L\|_p^p$ to within additive error $\varepsilon\|x\|_p^p$, where L is a list containing all α-heavy hitters for some $\alpha > 0$, with the promise that no $i \in L$ is not an $\alpha/2$-heavy hitter. In Section 3 we give a subroutine LightEstimator for estimating $\|x_{[n]\setminus L}\|_p^p$. Finally, in Section 4, we put everything together in a way that achieves optimal space and fast update time. We discuss how to compute L in the full version. We discuss entropy in Section A.

2. ESTIMATING THE CONTRIBUTION FROM HEAVY HITTERS

Before giving our algorithm HighEnd for estimating $\|x_L\|_p^p$, we first give a few necessary lemmas and theorems. The following theorem gives an algorithm for finding the ϕ-heavy hitters with respect to F_p. This algorithm uses the dyadic interval idea of [15] together with a black-box reduction of the problem of finding F_p heavy hitters to the problem of estimating F_p. Our proof is in the full version. We note that our data structure both improves and generalizes that of [21], which gave an algorithm with slightly worse bounds and only worked in the case $p = 1$.

THEOREM 1. *There is an algorithm* $\mathsf{F_pHH}$ *satisfying the following properties. Given $0 < \phi < 1$ and $0 < \delta < 1$, with probability at least $1 - \delta$,* $\mathsf{F_pHH}$ *produces a list L such that L contains all ϕ-heavy hitters and does not contain indices which are not $\phi/2$-heavy hitters. For each $i \in L$, the algorithm also outputs* $\text{sign}(x_i)$, *as well as an estimate $\tilde{x}_i$ of x_i satisfying $\tilde{x}_i^p \in [(6/7)|x_i|^p, (9/7)|x_i|^p]$. Its space usage is $O(\phi^{-1} \log(\phi n) \log(nmM) \log(\log(\phi n)/(\delta\phi)))$. Its update time is $O(\log(\phi n) \cdot \log(\log(\phi n)/(\delta\phi))$. Its reporting time is $O(\phi^{-1}(\log(\phi n) \cdot \log(\log(\phi n)/(\delta\phi))))$.*

The following moment bound can be derived from the Chernoff bound via integration, and is most likely standard though we do not know the earliest reference. A proof can be found in [35].

LEMMA 2. *Let $X_1, \ldots, X_n$ be such that X_i has expectation μ_i and variance σ_i^2, and $X_i \leq K$ almost surely. Then if the X_i are ℓ-wise independent for some even integer $\ell \geq 2$,*

$$\mathbf{E}\left[\left(\sum_{i=1}^n X_i - \mu\right)^\ell\right] \leq 2^{O(\ell)} \cdot \left(\left(\sigma\sqrt{\ell}\right)^\ell + (K\ell)^\ell\right),$$

where $\mu = \sum_i \mu_i$ and $\sigma^2 = \sum_i \sigma_i^2$. In particular,

$$\mathbf{Pr}\left[\left|\sum_{i=1}^n X_i - \mu\right| \geq \lambda\right] \leq 2^{O(\ell)} \cdot \left(\left(\sigma\sqrt{\ell}/\lambda\right)^\ell + (K\ell/\lambda)^\ell\right),$$

by Markov's inequality on the random variable $(\sum_i X_i - \mu)^\ell$.

LEMMA 3 (KHINTCHINE INEQUALITY [27]). *For $x \in \mathbb{R}^n$, $t \geq 2$, and uniformly random $z \in \{-1, 1\}^n$, $\mathbf{E}_z[|\langle x, z\rangle|^t] \leq \|x\|_2^t \cdot \sqrt{t}^t$.*

Henceforth in this section, i denotes $\sqrt{-1}$. The following lemma can be shown via a combination of Khintchine's inequality and Minkowski's inequality, and a proof can be found in the full version.

LEMMA 4. *Let $x \in \mathbb{R}^n$ be arbitrary. Let*

$$z \in \{e^{2\pi i/r}, e^{2\pi i \cdot 2/r}, e^{2\pi i \cdot 3/r}, \ldots, e^{2\pi i \cdot r/r}\}^n$$

be a random such vector for $r \geq 2$ an even integer. Then for $t \geq 2$ an even integer, $\mathbf{E}_z[|\langle x, z\rangle|^t] \leq \|x\|_2^t \cdot 2^{t/2}\sqrt{t}^t$.

2.1 The HighEnd data structure

In this section, we assume we know a subset $L \subseteq [n]$ of indices j so that

1. for all j for which $|x_j|^p \geq \alpha\|x\|_p^p$, $j \in L$,
2. if $j \in L$, then $|x_j|^p \geq (\alpha/2)\|x\|_p^p$,

3. for each $j \in L$, we know $\text{sign}(x_j)$.

for some $0 < \alpha < 1/2$ which we know. We also are given some $0 < \varepsilon < 1/2$, and we assume $1/\alpha = O(1/\varepsilon^2)$. We would like to output a value $\|x_L\|_p^p \pm O(\varepsilon)\|x\|_p^p$ with large constant probability

We first define the BasicHighEnd data structure. Put $s = \lceil 4/\alpha \rceil$. We choose a hash function $h : [n] \to [s]$ at random from an r_h-wise independent family for $r_h = \Theta(\log(1/\alpha))$. Also, let $r = \Theta(\log 1/\varepsilon)$ be a sufficiently large even integer. For each $j \in [n]$, we associate a random complex root of unity $e^{2\pi i g(j)/r}$, where $g : [n] \to [r]$ is drawn at random from an r_g-wise independent family for $r_g = r$. We initialize s counters $b_1, \ldots, b_s$ to 0. Given an update (j, v), add $e^{2\pi i g(j)/r} \cdot v$ to $b_{h(j)}$.

We now define the HighEnd data structure as follows. Define $T = \tau \cdot \max\{\log(1/\varepsilon), \log(2/\alpha)\}$ for a sufficiently large constant τ to be determined later. Define $t = 3T$ and instantiate t independent copies of the BasicHighEnd data structure. Given an update (j, v), perform the update described above to each of the copies of BasicHighEnd. We think of this data structure as a $t \times s$ matrix of counters $D_{j,k}$, $j \in [t]$ and $k \in [s]$. We let g^j be the hash function g in the jth independent instantiation of BasicHighEnd, and similarly define h^j. We sometimes use g to denote the tuple $(g^1, \ldots, g^t)$, and similarly for h.

We now define our estimator, but first we give some notation. For $w \in L$, let $j(w,1) < j(w,2) < \ldots < j(w, n_w)$ be the set of n_w indices $j \in [t]$ such that w is *isolated* by h^j from other indices in L; that is, indices $j \in [t]$ where no other $w' \in L$ collides with w under h^j.

Event $\mathcal{E}$. This is the event that $n_w \geq T$ for all $w \in L$.

If $\mathcal{E}$ does not hold, our estimator simply fails. Otherwise, define

$$x_w^* = \frac{1}{T} \cdot \sum_{k=1}^{T} e^{-2\pi i g^{j(w,k)}(w)/r} \cdot \text{sign}(x_w) \cdot D_{j(w,k), h^{j(w,k)}(w)}.$$

If $\mathbf{Re}[x_w^*] < 0$ for any $w \in L$, then we output fail. Otherwise, define $\Psi' = \sum_{w \in L} (x_w^*)^p$. Our estimator is then $\Psi = \mathbf{Re}[\Psi']$. Note x^* is a complex number. By z^p for complex z, we mean $|z|^p \cdot e^{ip \cdot \arg(z)}$, where $\arg(z) \in (-\pi, \pi]$ is the angle formed by the vector from the origin to z in the complex plane.

2.2 Analysis of HighEnd

For $w \in L$, we make the definitions $y_w \overset{\text{def}}{=} \frac{x_w^* - |x_w|}{|x_w|}$, $\Phi_w \overset{\text{def}}{=} |x_w|^p \cdot \left(\sum_{k=0}^{r/3} \binom{p}{k} \cdot y_w^k\right)$, and $\Phi \overset{\text{def}}{=} \sum_{w \in L} \Phi_w$. We assume $\mathcal{E}$ occurs so that the y_w and Φ_w (and hence Φ) are defined. Also, we use the definition $\binom{p}{k} = (\prod_{j=0}^{k-1}(p-j))/k!$ (note p may not be an integer).

Our overall goal is to show that $\Psi = \|x_L\|_p^p \pm O(\varepsilon) \cdot \|x\|_p^p$ with large constant probability. Our proof plan is to first show that $|\Phi - \|x_L\|_p^p| = O(\varepsilon) \cdot \|x\|_p^p$ with large constant probability, then to show that $|\Psi' - \Phi| = O(\varepsilon) \cdot \|x\|_p^p$ with large constant probability, at which point our claim follows by a union bound and the triangle inequality since $|\Psi - \|x_L\|_p^p| \leq |\Psi' - \|x_L\|_p^p|$ since $\|x_L\|_p^p$ is real.

Before analyzing Φ, we define the following event.

Event $\mathcal{D}$. Let $\mathcal{D}$ be the event that $\forall w \in L$,

$$\frac{1}{T^2} \sum_{k=1}^{T} \sum_{\substack{v \notin L \\ h^{j(w,k)}(v) = h^{j(w,k)}(w)}} x_v^2 < \frac{(\alpha \cdot \|x\|_p^p)^{2/p}}{r}.$$

We also define

$$V = \frac{1}{T^2} \sum_{w \in L} \sum_{j=1}^{t} \sum_{\substack{v \notin L \\ h^j(w) = h^j(v)}} |x_w|^{2p-2} \cdot |x_v|^2.$$

THEOREM 5. *Conditioned on h, $\mathbf{E}_g[\Phi] = \|x_L\|_p^p$ and $\mathbf{Var}_g[\Phi \mid \mathcal{D}] = O(V)$.*

Proof. By linearity of expectation,

$$\begin{aligned}
\mathbf{E}_g[\Phi] &= \sum_{w \in L} |x_w|^p \cdot \left[\sum_{k=0}^{r/3} \binom{p}{k} \mathbf{E}_g[y_w^k]\right] \\
&= \sum_{w \in L} |x_w|^p + \sum_{w \in L} |x_w|^p \cdot \sum_{k=1}^{r/3} \binom{p}{r} \mathbf{E}_g\left[y_w^k\right],
\end{aligned}$$

where we use that $\binom{p}{0} = 1$. Then $\mathbf{E}_g[y_w^k] = 0$ for $k > 0$ by using linearity of expectation and r_g-wise independence, since each summand involves at most $k < r$ rth roots of unity. Hence,

$$\mathbf{E}_g[\Phi] = \sum_{w \in L} |x_w|^p.$$

We now compute the variance. Note that if the g^j were each fully independent, then we would have $\mathbf{Var}_g[\Phi \mid \mathcal{D}] = \sum_{w \in L} \mathbf{Var}_g[\Phi_w \mid \mathcal{D}]$ since different Φ_w depend on evaluations of the g^j on disjoint $v \in [n]$. However, since $r_g > 2r/3$, $\mathbf{E}_g[|\Phi|^2]$ is identical as in the case of full independence of the g^j. We thus have $\mathbf{Var}_g[\Phi \mid \mathcal{D}] = \sum_{w \in L} \mathbf{Var}_g[\Phi_w \mid \mathcal{D}]$ and have reduced to computing $\mathbf{Var}_g[\Phi_w \mid \mathcal{D}]$.

$$\begin{aligned}
\mathbf{Var}_g[\Phi_w \mid \mathcal{D}] &= \mathbf{E}_g[|\Phi_w - \mathbf{E}_g[\Phi_w]|^2 \mid \mathcal{D}] \\
&= |x_w|^{2p} \cdot \mathbf{E}_g\left[\left|\sum_{k=1}^{r/3} \binom{p}{k} y_w^k\right|^2 \middle| \mathcal{D}\right] \\
&= |x_w|^{2p} \cdot (p^2 \cdot \mathbf{E}_g[|y_w|^2 \mid \mathcal{D}] + \sum_{k=2}^{r/3} O(\mathbf{E}_g[|y_w|^{2k} \mid \mathcal{D}]))
\end{aligned}$$

We have

$$\mathbf{E}_g[|y_w|^2 \mid \mathcal{D}] \overset{\text{def}}{=} u_w^2 = \frac{1}{T^2} \sum_{k=1}^{T} \sum_{v \notin L} \frac{x_v^2}{x_w^2} \cdot \mathbf{1}_{h^{j(w,k)}(v) = h^{j(w,k)}(w)}, \tag{1}$$

so that

$$\sum_{w \in L} p^2 \cdot \mathbf{E}_g[|y_w|^2 \mid \mathcal{D}] \leq p^2 V.$$

Eq. (1) follows since, conditioned on $\mathcal{E}$ so that y_w is defined,

$$\begin{aligned}
&\mathbf{E}_g[|y_w|^2] = \\
&\frac{1}{T^2 x_w^2} \sum_{k=1}^{T} \sum_{k'=1}^{T} \sum_{v \notin L} \sum_{v' \notin L} \mathbf{E}[e^{-2\pi i (g^{j(w,k)}(v) - g^{j(w,k')}(v'))/r}] \\
&\quad \times x_v x_{v'} \cdot \mathbf{1}_{h^{j(w,k)}(v) = h^{j(w,k)}(w)} \cdot \mathbf{1}_{h^{j(w,k')}(v') = h^{j(w,k')}(w)}.
\end{aligned}$$

When $j(w,k) \neq j(w,k')$ the above expectation is 0 since the g^j are independent across different j. When $j(w,k) = j(w,k')$ the above expectation is only non-zero for $v = v'$ since $r_g \geq 2$.

We also have for $k \geq 2$ that

$$\mathbf{E}_g[|y_w|^{2k} \mid \mathcal{D}] \leq 2^{O(k)} \cdot u_w^{2k} \cdot (2k)^k$$

by Lemma 4, so that

$$\sum_{k=2}^{r/3} \mathbf{E}_g[|y_w|^{2k} \mid \mathcal{D}] = O(u_w^2)$$

since $\mathcal{D}$ holds and so the sum is dominated by its first term. Thus, $\mathbf{Var}_g[\Phi \mid \mathcal{D}] = O(V)$. ■

Lemma 6. $\mathbf{E}_h[V] \leq 3\alpha \cdot \|x\|_p^{2p}/(4T)$.

Proof. For any $w \in L$, $v \notin L$, and $j \in [t]$, we have $\mathbf{Pr}_h[h^j(w) = h^j(v)] = 1/s \leq \alpha/4$ since $r_h \geq 2$. Thus,

$$\begin{aligned}\mathbf{E}_h[V] &\leq \frac{\alpha}{4T^2} \sum_{\substack{w\in L\\ v\notin L\\ j\in[t]}} |x_w|^{2p-2}|x_v|^2 \\ &= \frac{3\alpha}{4T}\left(\sum_{w\in L}|x_w|^p|x_w|^{p-2}\right)\left(\sum_{v\notin L}|x_v|^2\right) \\ &\leq \frac{3\alpha}{4T}\left(\sum_{w\in L}\|x\|_p^p(\alpha\cdot\|x\|_p^p)^{(p-2)/p}\right)\left(\frac{1}{\alpha}(\alpha\cdot\|x\|_p^p)^{2/p}\right) \qquad (2) \\ &= \frac{3}{4}\cdot\alpha\cdot\|x\|_p^{2p}/T.\end{aligned}$$

where Eq. (2) used that $\|x_{[n]\setminus L}\|_2^2$ is maximized when $[n]\setminus L$ contains exactly $1/\alpha$ coordinates v each with $|x_v|^p = \alpha\|x\|_p^p$, and that $|x_w|^{p-2} \leq (\alpha\cdot\|x\|_p^p)^{(p-2)/p}$ since $p \leq 2$. ■

Lemma 7. $\mathbf{Pr}_h[\mathcal{E}] \geq 1 - \varepsilon$.

Proof. For any $j \in [t]$, the probability that w is isolated by h^j is at least 1/2, since the expected number of collisions with w is at most 1/2 by pairwise independence of the h^j and the fact that $|L| \leq 2/\alpha$ so that $s \geq 2|L|$. If X is the expected number of buckets where w is isolated, the Chernoff bound gives $\mathbf{Pr}_h[X < (1-\epsilon)\mathbf{E}_h[X]] < \exp(-\epsilon^2\mathbf{E}_h[X]/2)$ for $0 < \epsilon < 1$. The claim follows for $\tau \geq 24$ by setting $\epsilon = 1/3$ then applying a union bound over $w \in L$. ■

Lemma 8. $\mathbf{Pr}_h[\mathcal{D}] \geq 63/64$.

Proof. We apply the bound of Lemma 2 for a single $w \in L$. Define $X_{j,v} = (x_v^2/T^2)\cdot\mathbf{1}_{h^j(v)=h^j(w)}$ and $X = \sum_{j=1}^t\sum_{v\notin L}X_{j,v}$. Note that X is an upper bound for the left hand side of the inequality defining $\mathcal{D}$, and thus it suffices to show a tail bound for X. In the notation of Lemma 2, we have $\sigma^2 \leq (3/(sT^3))\cdot\|x_{[n]\setminus L}\|_4^4$, $K = (\alpha\cdot\|x\|_p^p)^{2/p}/T^2$, and $\mu = (3/(sT))\cdot\|x_{[n]\setminus L}\|_2^2$. Since $\|x_{[n]\setminus L}\|_2^2$ and $\|x_{[n]\setminus L}\|_4^4$ are each maximized when there are exactly $1/\alpha$ coordinates $v \notin L$ with $|x_v|^p = \alpha\cdot\|x\|_p^p$,

$$\sigma^2 \leq \frac{3}{4T^3}\cdot(\alpha\cdot\|x\|_p^p)^{4/p}, \qquad \mu \leq \frac{3}{4T}\cdot(\alpha\cdot\|x\|_p^p)^{2/p}.$$

Setting $\lambda = (\alpha\cdot\|x\|_p^p)^{2/p}/(2r)$, noting that $\mu < \lambda$ for τ sufficiently large, and assuming $\ell \leq r_h$ is even, we apply Lemma 2 to obtain

$$\mathbf{Pr}[X \geq 2\lambda] \leq 2^{O(\ell)}\cdot\left(\left(\frac{\sqrt{3}r\cdot\sqrt{\ell}}{T^{3/2}}\right)^\ell + \left(\frac{2r\cdot\ell}{T^2}\right)^\ell\right).$$

By setting τ sufficiently large and $\ell = \log(2/\alpha)+6$, the above probability is at most $(1/64)\cdot(\alpha/2)$. The lemma follows by a union bound over all $w \in L$, since $|L| \leq 2/\alpha$. ■

We now define another event.

Event $\mathcal{F}$. Let $\mathcal{F}$ be the event that for all $w \in L$ we have $|y_w| < 1/2$.

Lemma 9. $\mathbf{Pr}_g[\mathcal{F} \mid \mathcal{D}] \geq 63/64$.

Proof. $\mathcal{D}$ occurring implies that

$$u_w \leq \sqrt{1/r} \leq \sqrt{1/(64(\log(2/\alpha)+6)}$$

(recall we assume $1/\alpha = O(1/\varepsilon^2)$ and pick $r = \Theta(\log(1/\varepsilon))$ sufficiently large, and u_w is as is defined in Eq. (1)), and we also have $\mathbf{E}_g[|y_w|^\ell \mid \mathcal{D}] < u_w^\ell\sqrt{\ell}^\ell 2^\ell$ by Lemma 4. Applying Markov's bound on the random variable $|y_w|^\ell$ for even $\ell \leq r_g$, we have $|y_w|^\ell$ is determined by r_g-wise independence of the g^j, and thus

$$\mathbf{Pr}_g[|y_w| \geq 1/2 \mid \mathcal{D}] < \left(\sqrt{\frac{16\ell}{64(\log(2/\alpha)+6)}}\right)^\ell,$$

which equals $(1/64)\cdot(\alpha/2)$ for $\ell = \log(2/\alpha)+6$. We then apply a union bound over all $w \in L$. ■

Lemma 10. *Given* $\mathcal{F}$, $|\Psi' - \Phi| < \varepsilon\|x_L\|_p^p$.

Proof. Observe

$$\Psi' = \sum_{w\in L}|x_w|^p\cdot(1+y_w)^p.$$

We have that $\ln(1+z)$, as a function of z, is holomorphic on the open disk of radius 1 about 0 in the complex plane, and thus $f(z) = (1+z)^p$ is holomorphic in this region since it is the composition $\exp(p\cdot\ln(1+z))$ of holomorphic functions. Therefore, $f(z)$ equals its Taylor expansion about 0 for all $z \in \mathbb{C}$ with $|z| < 1$ (see for example [51, Theorem 11.2]). Then since $\mathcal{F}$ occurs, we can Taylor-expand f about 0 for $z = y_w$ and apply Taylor's theorem to obtain

$$\begin{aligned}\Psi' &= \sum_{w\in L}|x_w|^p\left(\sum_{k=0}^{r/3}\binom{p}{k}y_w^k \pm O\left(\binom{p}{r/3+1}\cdot|y_w|^{-r/3-1}\right)\right) \\ &= \Phi + O\left(\|x_L\|_p^p\cdot\left(\binom{p}{r/3+1}\cdot|y_w|^{-r/3-1}\right)\right)\end{aligned}$$

The lemma follows since $\binom{p}{r/3+1} < 1$ and $|y_w|^{-r/3-1} < \varepsilon$ for $|y_w| < 1/2$. ■

Theorem 11. *The space used by* HighEnd *is*

$$O(\alpha^{-1}\log(1/\varepsilon)\log(mM/\varepsilon)\log^2(1/\varepsilon)\log n).$$

The update time is $O(\log^2(1/\varepsilon))$. *The reporting time is* $O(\alpha^{-1}\log(1/\varepsilon)\log(1/\alpha))$. *Also,* $\mathbf{Pr}_{h,g}[|\Psi - \|x_L\|_p^p| < O(\varepsilon)\cdot\|x\|_p^p] > 7/8$.

Proof. We first argue correctness. By a union bound, $\mathcal{E}$ and $\mathcal{D}$ hold simultaneously with probability 31/32. By Markov's inequality and Lemma 6, $V = O(\alpha \cdot \|x\|_p^{2p}/T)$ with probability 63/64. We then have by Chebyshev's inequality and Theorem 5 that $|\Phi - \|x_L\|_p^p| = O(\varepsilon) \cdot \|x\|_p^p$ with probability 15/16. Lemma 10 then implies $|\Psi' - \|x_L\|_p^p| = O(\varepsilon) \cdot \|x\|_p^p$ with probability $15/16 - \mathbf{Pr}[\neg\mathcal{F}] > 7/8$ by Lemma 9. In this case, the same must hold true for Ψ since $\Psi = \mathbf{Re}[\Psi']$ and $\|x_L\|_p^p$ is real.

Next we discuss space complexity. We start with analyzing the precision required to store the counters $D_{j,k}$. Since our correctness analysis conditions on $\mathcal{F}$, we can assume $\mathcal{F}$ holds. We store the real and imaginary parts of each counter $D_{j,k}$ separately. If we store each such part to within precision $\gamma/(2mT)$ for some $0 < \gamma < 1$ to be determined later, then each of the real and imaginary parts, which are the sums of at most m summands from the m updates in the stream, is stored to within additive error $\gamma/(2T)$ at the end of the stream. Let $\tilde{x}_w^*$ be our calculation of x_w^* with such limited precision. Then, each of the real and imaginary parts of $\tilde{x}_w^*$ is within additive error $\gamma/2$ of those for x_w^*. Since $\mathcal{F}$ occurs, $|x_w^*| > 1/2$, and thus $\gamma/2 < \gamma|x_w^*|$, implying $|\tilde{x}_w^*| = (1 \pm O(\gamma))|x_w^*|$. Now we argue $\arg(\tilde{x}_w^*) = \arg(x_w^*) \pm O(\sqrt{\gamma})$. Write $x_w^* = a + ib$ and $\tilde{x}_w^* = \tilde{a} + i\tilde{b}$ with $\tilde{a} = a \pm \gamma/2$ and $\tilde{b} = b \pm \gamma/2$. We have $\cos(\arg(x_w^*)) = a/\sqrt{a^2+b^2}$. Also, $\cos(\arg(\tilde{x}_w^*)) = (a \pm \gamma/2)/((1 \pm O(\gamma))\sqrt{a^2+b^2}) = (1 \pm O(\gamma))\cos(\arg(x_w^*)) \pm O(\gamma) = \cos(\arg(x_w^*)) \pm O(\gamma)$, implying $\arg(\tilde{x}_w^*) = \arg(x_w^*) \pm O(\sqrt{\gamma})$. Our final output is $\sum_{w\in L} |\tilde{x}_w^*|^p \cdot \cos(p \cdot \arg(\tilde{x}_w^*))$. Since cos never has derivative larger than 1 in magnitude, this is $\sum_{w\in L}[(1 \pm O(\gamma))|x_w^*|^p \cos(p \cdot \arg(x_w^*)) \pm O(\sqrt{\gamma}) \cdot (1 \pm O(\gamma))|x_w^*|^p]$. Since $\mathcal{F}$ occurs, $|x_w^*|^p < (3/2)^p \cdot |x_w|^p$, and thus our overall error introduced from limited precision is $O(\sqrt{\gamma} \cdot \|x_L\|_p^p)$, and it thus suffices to set $\gamma = O(\varepsilon^2)$, implying each $D_{j,k}$ requires $O(\log(mM/\varepsilon))$ bits of precision. For the remaining part of the space analysis, we discuss storing the hash functions. The hash functions h^j, g^j each require $O(\log(1/\varepsilon)\log n)$ bits of seed, and thus in total consume $O(\log^2(1/\varepsilon)\log n)$ bits.

Finally we discuss time complexity. To perform an update, for each $j \in [t]$ we must evaluate g^j and h^j then update a counter. Each of g^j, h^j require $O(\log(1/\varepsilon))$ time to evaluate. For the reporting time, we can mark all counters with the unique $w \in L$ which hashes to it under the corresponding h^j (if a unique such w exists) in $|L| \cdot t \cdot r_h = O(\alpha^{-1}\log(1/\varepsilon)\log(1/\alpha))$ time. Then, we sum up the appropriate counters for each $w \in L$, using the Taylor expansion of $\cos(p \cdot \arg(z))$ up to the $\Theta(\log(1/\varepsilon))$th degree to achieve additive error ε. Note that conditioned on $\mathcal{F}$, $\arg(x_w^*) \in (-\pi/4, \pi/4)$, so that $|p \cdot \arg(x_w^*)|$ is bounded away from $\pi/2$ for p bounded away from 2; in fact, one can even show via some calculus that $\arg(x_w^*) \in (-\pi/6, \pi/6)$ when $\mathcal{F}$ occurs by showing that $\cos(\arg(x_w^*)) = \cos(\arg(1 - y_w))$ is minimized for $|y_w| \le 1/2$ when $y_w = 1/4 + i\sqrt{3}/4$. Regardless, additive error ε is relative error $O(\varepsilon)$, since if $|p \cdot \arg(z)|$ is bounded away from $\pi/2$, then $|\cos(p \cdot \arg(z))| = \Omega(1)$. ∎

3. ESTIMATING THE CONTRIBUTION FROM LIGHT ELEMENTS

In this section, we show how to estimate the contribution to F_p from coordinates of x which are not heavy hitters. More precisely, given a list $L \subseteq [n]$ such that $|L| \le 2/\varepsilon^2$ and $|x_i|^p \le \varepsilon^2\|x\|_p^p$ for all $i \notin L$, we describe a subroutine LightEstimator that outputs a value that is $\|x_{[n]\setminus L}\|_p^p \pm O(\varepsilon) \cdot \|x\|_p^p$ with probability at least 7/8. We show the following in the full version.

Theorem 12. *For any $0 < p < 2$, there is a randomized data structure D_p, and a deterministic algorithm Est_p mapping the state space of the data structure to reals, such that*

1. $\mathbf{E}[\mathsf{Est}_p(D_p(x))] = (1 \pm \varepsilon)\|x\|_p^p$
2. $\mathbf{E}[\mathsf{Est}_p(D_p(x))^2] \le C_p \cdot \|x\|_p^{2p}$

for some constant $C_p > 0$ depending only on p, and where the expectation is taken over the randomness used by D_p. Aside from storing a length-$O(\varepsilon^{-p}\log(nmM))$ random string, the space complexity is $O(\log(nmM))$. The update time is the time to evaluate a $\Theta(1/\varepsilon^p)$-wise independent hash function over a field of size $\mathrm{poly}(nmM)$, and the reporting time is $O(1)$.

We also need the following algorithm for fast multipoint evaluation of polynomials.

Theorem 13 ([50, Ch. 10]). *Let $\mathbf{R}$ be a ring, and let $q \in \mathbf{R}[x]$ be a degree-d polynomial. Then, given distinct $x_1, \ldots, x_d \in \mathbf{R}$, all the values $q(x_1), \ldots, q(x_d)$ can be computed using $O(d\log^2 d \log\log d)$ operations over $\mathbf{R}$.*

The guarantees of the final LightEstimator are given in Theorem 15, which is a modified form of an algorithm designed in [41] for $p = 1$. A description of the modifications of the algorithm in [41] needed to work for $p \ne 1$ is in Remark 16, which in part uses the following uniform hash family of Pagh and Pagh [45].

Theorem 14 (Pagh and Pagh [45, Theorem 1.1]). *Let $S \subseteq U = [u]$ be a set of $z > 1$ elements, and let $V = [v]$, with $1 < v \le u$. Suppose the machine word size is $\Omega(\log(u))$. For any constant $c > 0$ there is a word RAM algorithm that, using time $\log(z)\log^{O(1)}(v)$ and $O(\log(z) + \log\log(u))$ bits of space, selects a family $\mathcal{H}$ of functions from U to V (independent of S) such that:*

1. *With probability $1 - O(1/z^c)$, $\mathcal{H}$ is z-wise independent when restricted to S.*
2. *Any $h \in \mathcal{H}$ can be represented by a RAM data structure using $O(z\log(v))$ bits of space, and h can be evaluated in constant time after an initialization step taking $O(z)$ time.*

Theorem 15 ([41]). *Suppose we are given $0 < \varepsilon < 1/2$, and given a list $L \subseteq [n]$ at the end of the data stream such that $|L| \le 2/\varepsilon^2$ and $|x_i|^p < \varepsilon^2\|x\|_p^p$ for all $i \notin L$. Then, given access to a randomized data structure satisfying properties (1) and (2) of Theorem 12, there is an algorithm LightEstimator satisfying the following. The randomness used by LightEstimator can be broken up into a certain random hash function h, and another random string s. LightEstimator outputs a value Φ' satisfying $\mathbf{E}_{h,s}[\Phi'] = (1 \pm O(\varepsilon))\|x_{[n]\setminus L}\|_p^p$, and $\mathbf{E}_h[\mathbf{Var}_s[\Phi']] = O(\varepsilon^2\|x\|_p^{2p})$. The space usage is $O(\varepsilon^{-2}\log(nmM))$ bits, the update time is $O(\log^2(1/\varepsilon)\log\log(1/\varepsilon))$, and the reporting time is $O(1/\varepsilon^2)$.*

Remark 16. The claim of Theorem 15 is not stated in the same form in [41], and thus we provide some explanation. The work of [41] only focused on the case $p = 1$.

There, in Section 3.2, LightEstimator was defined[2] by creating $R = 4/\varepsilon^2$ independent instantiations of D_1, which we label $D_1^1, \ldots, D_1^R$ (R chosen so that $R \geq 2|L|$), and picking a hash function $h : [n] \to [R]$ from a random hash family constructed as in Theorem 14 with $z = R$ and $c \geq 2$. Upon receiving an update to x_i in the stream, the update was fed to $D_1^{h(i)}$. The final estimate was defined as follows. Let $I = [R] \backslash h(L)$. Then, the estimate was $\Phi' = (R/|I|) \cdot \sum_{j \in I} \mathsf{Est}_1(D_1^j)$. In place of a generic D_1, the presentation in [41] used Li's geometric mean estimator [38], though the analysis (Lemmas 7 and 8 of [41]) only made use of the generic properties of D_1 and Est_1 given in Theorem 12. Let $s = (s_1, \ldots, s_R)$ be the tuple of random strings used by the D_1^j, where the entries of s are pairwise independent. The analysis then showed that (a) $\mathbf{E}_{h,s}[\Phi'] = (1 \pm O(\varepsilon))\|x_{[n]\backslash L}\|_1$, and (b) $\mathbf{E}_h[\mathbf{Var}_s[\Phi']] = O(\varepsilon^2\|x\|_1^2)$. For (a), the same analysis applies for $p \neq 1$ when using Est_p and D_p instead. For (b), it was shown that $\mathbf{E}_h[\mathbf{Var}_s[\Phi']] = O(\|x_{[n]\backslash L}\|_2^2 + \varepsilon^2\|x_{[n]\backslash L}\|_1^2)$. The same analysis shows that $\mathbf{E}_h[\mathbf{Var}_s[\Phi']] = O(\|x_{[n]\backslash L}\|_{2p}^{2p} + \varepsilon^2\|x_{[n]\backslash L}\|_p^p)$ for $p \neq 1$. Since L contains all the ε^2-heavy hitters, $\|x_{[n]\backslash L}\|_{2p}^{2p}$ is maximized when there are $1/\varepsilon^2$ coordinates $i \in [n]\backslash L$ each with $|x_i|^p = \varepsilon^2\|x\|_p^p$, in which case $\|x_{[n]\backslash L}\|_{2p}^{2p} = \varepsilon^2\|x\|_p^{2p}$.

To achieve the desired update time, we buffer every $d = 1/\varepsilon^p$ updates then perform the fast multipoint evaluation of Theorem 13 in batch (note this does not affect our space bound since $p < 2$). That is, although the hash function h can be evaluated in constant time, updating any D_p^j requires evaluating a degree-d (for $d = \Omega(1/\varepsilon^p)$) polynomial, which naïvely requires $\Omega(1/\varepsilon^p)$ time. One issue is that the different data structures D_p^j use different polynomials, and thus we may need to evaluate $1/\varepsilon^p$ different polynomials on the $1/\varepsilon^p$ points, defeating the purpose of batching. However, these polynomials are themselves pairwise independent. That is, we can assume there are two coefficient vectors a, b of length $d+1$, and the polynomial corresponding to D_p^j is given by the coefficient vector $j \cdot a + b$. Thus, we only need to perform fast multipoint evaluation on the two polynomials defined by a and b. To achieve worst-case update time, this computation can be spread over the next d updates. If a query comes before d updates are batched, we need to perform $O(d \log d \log\log d)$ work at once, but this is already dominated by our $O(1/\varepsilon^2)$ reporting time since $p < 2$.

4. THE FINAL ALGORITHM: PUTTING IT ALL TOGETHER

To obtain our final algorithm, one option is to run HighEnd and LightEstimator in parallel after finding L, then output the sum of their estimates. Note that by the variance bound in Theorem 15, the output of a single instantiation of LightEstimator is $\|x_{[n]\backslash L}\|_p^p \pm O(\varepsilon)\|x\|_p^p$ with large constant probability. The downside to this option is that Theorem 1 uses space that would make our overall F_p-estimation algorithm suboptimal by $\mathrm{polylog}(n/\varepsilon)$ factors, and HighEnd by an $O(\log(1/\varepsilon))$ factor for $\alpha = \varepsilon^2$ (Theorem 11). We can overcome this by a combination of balancing and universe reduction. Specifically, for balancing, notice that if we instead define L as a list of ϵ^2-heavy hitters for some $\epsilon > \varepsilon$, we could improve the space of both Theorem 1 and Theorem 11. To then make the variance in LightEstimator sufficiently small, i.e. $O(\varepsilon^2\|x\|_p^2)$, we could run $O((\epsilon/\varepsilon)^2)$ instantiations of LightEstimator in parallel and output the average estimate, keeping the space optimal but increasing the update time by a factor $\Omega((\epsilon/\varepsilon)^2)$. This balancing gives a smooth tradeoff between space and update time; in fact note that for $\epsilon = 1$, our overall algorithm simply becomes a derandomized variant of Li's geometric mean estimator. We would like though to have $\epsilon \ll 1$ to have small update time.

Doing this balancing does not resolve all our issues though, since Theorem 1 is suboptimal by a $\log n$ factor. That is, even if we picked $\epsilon = 1$, Theorem 1 would cause our overall space to be $\Omega(\log(n)\log(mM))$, which is suboptimal. To overcome this issue we use universe reduction. Specifically, we set $N = 1/\varepsilon^{18}$ and pick hash functions $h_1 : [n] \to [N]$ and $\sigma : [n] \to \{-1, 1\}$. We define a new N-dimensional vector y by $y_i = \sum_{h_1(j)=i} \sigma(j)x_j$. Henceforth in this section, y, h_1, and σ are as discussed here. Rather than computing a list L of heavy hitters of x, we instead compute a list L' of heavy hitters of y. Then, since y has length only $\mathrm{poly}(1/\varepsilon)$, Theorem 1 is only suboptimal by $\mathrm{polylog}(1/\varepsilon)$ factors and our balancing trick applies. The list L' is also used in place of L for both HighEnd and LightEstimator. Though, since we never learn L, we must modify our choice of hash functions in LightEstimator. Specifically, the hash function $h : [n] \to [R]$ in Remark 16 should be implemented as the composition of h_1, and a hash function $h_2 : [N] \to [R]$ chosen as in Theorem 14 (again with $z = R$ and $c = 2$). Then, we let $I = [R]\backslash h_2(L')$. The remaining parts of the algorithm are the same.

There are several issues we must address to show that our universe reduction step maintains correctness. Informally, we need that (a) any i which is a heavy hitter for y should have exactly one $j \in [n]$ with $h_1(j) = i$ such that j was a heavy hitter for x, (b) if i is a heavy hitter for x, then $h_1(i)$ is a heavy hitter for y, and $|y_{h_1(i)}|^p = (1 \pm O(\varepsilon))|x_i|^p$ so that x_i's contribution to $\|x\|_p^p$ is properly approximated by HighEnd, (c) $\|y\|_p^p = O(\|x\|_p^p)$ with large probability, since the error term in HighEnd is $O(\varepsilon \cdot \|y\|_p^p)$, and (d) the amount of F_p mass not output by LightEstimator because it collided with a heavy hitter for x under h_1 is negligible. Also, the composition $h = h_1 \circ h_2$ for LightEstimator does not satisfy the conditions of Theorem 14 even though h_1 and h_2 might do so individually. To see why, as a simple analogy, consider that the composition of two purely random functions is no longer random. For example, as the number of compositions increases, the probability of two items colliding increases as well. Nevertheless, the analysis of LightEstimator carries over essentially unchanged in this setting, since whenever considering the distribution of where two items land under h, we can first condition on them not colliding under h_1. Not colliding under h_1 happens with $1 - O(\varepsilon^{18})$ probability, and thus the probability that two items land in two particular buckets $j, j' \in [R]$ under h is still $(1 \pm o(\varepsilon))/R^2$. The details are in the full version.

We now put everything together. We set $\epsilon = \varepsilon\log(1/\varepsilon)$. As stated earlier, we define L' to be the sublist of those w output by our $\mathsf{F_pHH}$ instantiation with $\phi = \epsilon^2$ such that $|\tilde{y}_w|^p \geq (2\varepsilon^2/7)\tilde{F}_p$. For ease of presentation, define L_ϕ to be the list of ϕ-heavy hitters of x with respect to F_p ("L", without a subscript, always denotes the ε^2-heavy hitters with respect to x), and define $z_i = \sum_{w \in h_1^{-1}(i)\backslash L_{\varepsilon^8}} \sigma(w)x_w$, i.e. z_i is the contribution to y_i from the significantly light elements of x. We interpret updates to x as updates to y to then be fed into HighEnd, with $\alpha = \epsilon^2/(34C)$. Thus both

[2]The estimator given there was never actually named, so we name it LightEstimator here.

HighEnd and $\mathsf{F_pHH}$ require $O(\varepsilon^{-2}\log(nmM/\varepsilon))$ space. We now define some events.

Event $\mathcal{A}$. L_{ε^8} is perfectly hashed under h_1, and $\forall i \in [N], |z_i|^p = O(\log(1/\varepsilon)^{p/2}\cdot\varepsilon^6\|x\|_p^p)$.

Event $\mathcal{B}$. $\forall w \in L_{\epsilon^2}$, $h_1(w)$ is output as an $\epsilon^2/(34C)$-heavy hitter by $\mathsf{F_pHH}$.

Event $\mathcal{C}$. $\forall w \in L_{\epsilon^2/18}$, $|y_{h_1(w)}| = (1\pm O(\varepsilon))|x_w|$.

Event $\mathcal{D}$. $\tilde{F}_p \in [(1/2)\cdot\|x\|_p^p, (3/2)\cdot\|x\|_p^p]$, and HighEnd, LightEstimator, and $\mathsf{F_pHH}$ succeed.

Event $\mathcal{E}$. $\|y\|_p^p = O(\|x\|_p^p)$.

Now, suppose $\mathcal{A}$, $\mathcal{B}$, $\mathcal{C}$, $\mathcal{D}$, and $\mathcal{E}$ all occur. Then for $w \in L_{\epsilon^2}$, w is output by $\mathsf{F_pHH}$, and furthermore $|y_{h_1(w)}|^p \geq (1-O(\varepsilon))|x_w|^p \geq |x_w|^p/2 \geq \epsilon^2\|x\|_p^p/2$. Also, $\tilde{y}^p_{h_1(w)} \geq (6/7)\cdot|y_{h_1(w)}|^p$. Since $\tilde{F}_p \leq 3\|x\|_p^p/2$, we have that $h_1(w) \in L'$. Furthermore, we also know that for i output by $\mathsf{F_pHH}$, $\tilde{y}_i^p \leq (9/7)\cdot|y_i|^p$, and thus $i \in L'$ implies $|y_i|^p \geq (\epsilon^2/9)\cdot\|x\|_p^p$. Notice that by event $\mathcal{A}$, each y_i is z_i, plus potentially $x_{w(i)}$ for some $x_{w(i)} \in L_{\varepsilon^8}$. If $|y_i|^p \geq (\epsilon^2/9)\cdot\|x\|_p^p$, then there must exist such a $w(i)$, and furthermore it must be that $|x_{w(i)}|^p \geq (\epsilon^2/18)\cdot\|x\|_p^p$. Thus, overall, L' contains $h_1(w)$ for all $w \in L_{\epsilon^2}$, and furthermore if $i \in L'$ then $w(i) \in L_{\epsilon^2/18}$.

Since L' contains $h_1(L_{\epsilon^2})$, LightEstimator's output is

$$\|x_{[n]\setminus h^{-1}(L')}\|_p^p \pm O(\varepsilon\|x\|_p^p).$$

Also, HighEnd outputs $\|y_{L'}\|_p^p \pm O(\varepsilon)\cdot\|y\|_p^p$, which is $\|x_{L'}\|_p^p \pm O(\varepsilon)\cdot\|x\|_p^p$ by events $\mathcal{C}$ and $\mathcal{E}$. In the full version, we show that all these events occur simultaneously with constant probability above 1/2.

Also, notice for a single instantiation of LightEstimator we have $\mathbf{E}_h[\mathbf{Var}_s[\Phi']] = O(\epsilon^2\|x\|_p^{2p})$. Once h is fixed, the variance of Φ' is simply the sum of variances across the D_j for $j \notin h_1(L')$. Thus, it suffices for the D_j to use pairwise independent randomness. Furthermore, in repeating $O((\epsilon/\varepsilon)^2)$ parallel repetitions of LightEstimator, it suffices that all the D_j across all parallel repetitions use pairwise independent randomness, and the hash function h can remain the same. Thus coefficients of the degree-$O(1/\varepsilon^p)$ polynomials used in all D_j combined can be generated by just two coefficient vectors, as in Remark 16, and thus the update time of LightEstimator with $O((\epsilon/\varepsilon)^2)$ parallel repetitions is just $O((\epsilon/\varepsilon)^2 + O(\log^2(1/\varepsilon)\log\log(1/\varepsilon))) = O(\log^2(1/\varepsilon)\log\log(1/\varepsilon))$. Overall, we have the following.

Theorem 17. *There exists an algorithm such that given* $0 < p < 2$ *and* $0 < \varepsilon < 1/2$*, the algorithm outputs* $(1\pm\varepsilon)\|x\|_p^p$ *with probability* $2/3$ *using* $O(\varepsilon^{-2}\log(nmM/\varepsilon))$ *space. The update time is* $O(\log^2(1/\varepsilon)\log\log(1/\varepsilon))$*. The reporting time is* $O(\varepsilon^{-2}\log^2(1/\varepsilon)\log\log(1/\varepsilon))$.

The space bound above can be assumed $O(\varepsilon^{-2}\log(mM) + \log\log n)$ by comments in Section 1.3.

5. REFERENCES

[1] C. C. Aggarwal, A. Hinneburg, and D. A. Keim. On the surprising behavior of distance metrics in high dimensional spaces. In *ICDT*, pages 420–434, 2001.

[2] N. Alon, P. B. Gibbons, Y. Matias, and M. Szegedy. Tracking join and self-join sizes in limited storage. *J. Comput. Syst. Sci.*, 64(3):719–747, 2002.

[3] N. Alon, Y. Matias, and M. Szegedy. The Space Complexity of Approximating the Frequency Moments. *J. Comput. Syst. Sci.*, 58(1):137–147, 1999.

[4] K. D. Ba, P. Indyk, E. Price, and D. P. Woodruff. Lower bounds for sparse recovery. In *SODA*, pages 1190–1197, 2010.

[5] Z. Bar-Yossef. *The Complexity of Massive Data Set Computations*. PhD thesis, University of California at Berkeley, 2002.

[6] Z. Bar-Yossef, T. S. Jayram, R. Kumar, and D. Sivakumar. An information statistics approach to data stream and communication complexity. *J. Comput. Syst. Sci.*, 68(4):702–732, 2004.

[7] L. Bhuvanagiri and S. Ganguly. Estimating entropy over data streams. In *Proceedings of the 14th Annual European Symposium on Algorithms (ESA)*, pages 148–159, 2006.

[8] L. Bhuvanagiri, S. Ganguly, D. Kesh, and C. Saha. Simpler algorithm for estimating frequency moments of data streams. In *Proceedings of the 17th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 708–713, 2006.

[9] A. Chakrabarti, G. Cormode, and A. McGregor. A near-optimal algorithm for estimating the entropy of a stream. *ACM Transactions on Algorithms*, 6(3), 2010.

[10] A. Chakrabarti, K. Do Ba, and S. Muthukrishnan. Estimating Entropy and Entropy Norm on Data Streams. In *Proceedings of the 23rd Annual Symposium on Theoretical Aspects of Computer Science (STACS)*, pages 196–205, 2006.

[11] A. Chakrabarti, S. Khot, and X. Sun. Near-optimal lower bounds on the multi-party communication complexity of set disjointness. In *Proceedings of the 18th Annual IEEE Conference on Computational Complexity (CCC)*, pages 107–117, 2003.

[12] M. Charikar, K. Chen, and M. Farach-Colton. Finding frequent items in data streams. In *Proceedings of the 29th International Colloquium on Automata, Languages and Programming (ICALP)*, pages 693–703, 2002.

[13] K. L. Clarkson and D. Woodruff. Numerical linear algebra in the streaming model. In *STOC*, 2009.

[14] G. Cormode, P. Indyk, N. Koudas, and S. Muthukrishnan. Fast mining of massive tabular data via approximate distance computations. In *ICDE*, pages 605–, 2002.

[15] G. Cormode and S. Muthukrishnan. An improved data stream summary: the count-min sketch and its applications. *J. Algorithms*, 55(1):58–75, 2005.

[16] G. Cormode, S. Muthukrishnan, and I. Rozenbaum. Summarizing and mining inverse distributions on data streams via dynamic inverse sampling. In *VLDB*, pages 25–36, 2005.

[17] I. Diakonikolas, D. M. Kane, and J. Nelson. Bounded independence fools degree-2 threshold functions. In *Proceedings of the 51st Annual IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 11–20, 2010.

[18] J. Feigenbaum, S. Kannan, M. Strauss, and M. Viswanathan. An approximate L1-difference algorithm for massive data streams. *SIAM J. Comput.*, 32(1):131–151, 2002.

[19] M. L. Fredman, J. Komlós, and E. Szemerédi. Storing a sparse table with 0(1) worst case access time. *J. ACM*, 31(3):538–544, 1984.

[20] S. Ganguly. Estimating frequency moments of data streams using random linear combinations. In *Proceedings of the 8th International Workshop on Randomization and Computation (RANDOM)*, pages 369–380, 2004.

[21] S. Ganguly, A. N. Singh, and S. Shankar. Finding frequent items over general update streams. In *Proceedings of the 20th International Conference on Scientific and Statistical Database Management (SSDBM)*, pages 204–221, 2008.

[22] D. Geiger, T.-L. Liu, and M. J. Donahue. Sparse representations for image decompositions. *International Journal of Computer Vision*, 33(2):139–156, 1999.

[23] A. Gronemeier. Asymptotically optimal lower bounds on the NIH-multi-party information complexity of the AND-function and disjointness. In *Proceedings of the 26th International Symposium on Theoretical Aspects of Computer Science (STACS)*, pages 505–516, 2009.

[24] Y. Gu, A. McCallum, and D. F. Towsley. Detecting anomalies in network traffic using maximum entropy estimation. In *Proceedings of the Internet Measurement Conference (IMC)*, pages 345–350, 2005.

[25] S. Guha, P. Indyk, and A. McGregor. Sketching information divergences. *Machine Learning*, 72(1-2):5–19, 2008.

[26] S. Guha, A. McGregor, and S. Venkatasubramanian. Streaming and sublinear approximation of entropy and information distances. In *Proceedings of the 17th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 733–742, 2006.

[27] U. Haagerup. The best constants in the Khintchine inequality. *Studia Math.*, 70(3):231–283, 1982.

[28] N. J. A. Harvey, J. Nelson, and K. Onak. Sketching and

streaming entropy via approximation theory. In *Proceedings of the 49th Annual IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 489–498, 2008.

[29] P. Indyk. Stable distributions, pseudorandom generators, embeddings, and data stream computation. *J. ACM*, 53(3):307–323, 2006.

[30] P. Indyk and A. McGregor. Declaring independence via the sketching of sketches. In *SODA*, pages 737–745, 2008.

[31] P. Indyk and D. P. Woodruff. Optimal approximations of the frequency moments of data streams. In *Proceedings of the 37th Annual ACM Symposium on Theory of Computing (STOC)*, pages 202–208, 2005.

[32] T. S. Jayram. Hellinger strikes back: A note on the multi-party information complexity of AND. In *APPROX-RANDOM*, pages 562–573, 2009.

[33] T. S. Jayram, R. Kumar, and D. Sivakumar. The one-way communication complexity of hamming distance. *Theory of Computing*, 4(1):129–135, 2008.

[34] T. S. Jayram and D. P. Woodruff. Optimal bounds for Johnson-Lindenstrauss transforms and streaming problems with low error. In *Proceedings of the 22nd Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 1–10, 2011.

[35] D. M. Kane and J. Nelson. A derandomized sparse Johnson-Lindenstrauss transform. *CoRR*, abs/1006.3585v1, 2010.

[36] D. M. Kane, J. Nelson, and D. P. Woodruff. On the exact space complexity of sketching and streaming small norms. In *Proceedings of the 21st Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 1161–1178, 2010.

[37] B. Krishnamurthy, S. Sen, Y. Zhang, and Y. Chen. Sketch-based change detection: methods, evaluation, and applications. In *Proceedings of the 3rd ACM SIGCOMM Conference on Internet Measurement (IMC)*, pages 234–247, 2003.

[38] P. Li. Estimators and tail bounds for dimension reduction in l_p $(0 < p \leq 2)$ using stable random projections. In *Proceedings of the 19th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 10–19, 2008.

[39] P. Li. On practical algorithms for entropy estimation and the improved sample complexity of compressed counting. *CoRR*, abs/1004.3782, 2010.

[40] S. Muthukrishnan. Data Streams: Algorithms and Applications. *Foundations and Trends in Theoretical Computer Science*, 1(2):117–236, 2005.

[41] J. Nelson and D. Woodruff. Fast Manhattan sketches in data streams. In *Proceedings of the 29th ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems (PODS)*, pages 99–110, 2010.

[42] J. Nelson and D. P. Woodruff. A near-optimal algorithm for L1-difference. *CoRR*, abs/0904.2027, 2009.

[43] N. Nisan. Pseudorandom generators for space-bounded computation. *Combinatorica*, 12(4):449–461, 1992.

[44] Open Problems in Data Streams and Related Topics. IITK Workshop on Algorithms for Data Streams, 2006. `http://www.cse.iitk.ac.in/users/sganguly/data-stream-probs.pdf`.

[45] A. Pagh and R. Pagh. Uniform hashing in constant time and linear space. *SIAM J. Comput.*, 38(1):85–96, 2008.

[46] M. E. Saks and X. Sun. Space lower bounds for distance approximation in the data stream model. In *Proceedings of the 34th Annual ACM Symposium on Theory of Computing (STOC)*, pages 360–369, 2002.

[47] T. Sarlós. Improved approximation algorithms for large matrices via random projections. In *In Proc. 47th Annu. IEEE Sympos. Found. Comput. Sci*, pages 143–152. IEEE Computer Society, 2006.

[48] D. Srivastava and S. Venkatasubramanian. Information theory for data management. In *SIGMOD '10: Proceedings of the 2010 international conference on Management of data*, pages 1255–1256, New York, NY, USA, 2010. ACM.

[49] M. Thorup and Y. Zhang. Tabulation based 4-universal hashing with applications to second moment estimation. In *Proceedings of the 15th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 615–624, 2004.

[50] J. von zur Gathen and J. Gerhard. *Modern Computer Algebra*. Cambridge University Press, 1999.

[51] M. Wong. *Complex Analysis*, volume 2. World Scientific, 2008.

[52] D. P. Woodruff. Optimal space lower bounds for all frequency moments. In *Proceedings of the 15th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 167–175, 2004.

[53] D. P. Woodruff. *Efficient and Private Distance Approximation in the Communication and Streaming Models*. PhD thesis, Massachusetts Institute of Technology, 2007.

[54] H. Zhao, A. Lall, M. Ogihara, O. Spatscheck, J. Wang, and J. Xu. A Data Streaming Algorithm for Estimating Entropies of OD Flows. In *Proceedings of the Internet Measurement Conference (IMC)*, 2007.

APPENDIX

A. IMPROVEMENTS TO ENTROPY ESTIMATION

In this section we give the details on how our algorithm, when combined with the work of [28], gives near-optimal space algorithms with fast update time for additive and multiplicative estimation of empirical entropy in data streams.

First we discuss additive ε approximation of entropy. The algorithm of [28] computes $(1+\varepsilon')$-approximations to F_p for $k = \log(1/\varepsilon) + \log\log(mM)$ different values of $p \in (0,1)$, where $\varepsilon' = \Theta(\varepsilon/(k^3 \log(mM)))$. Using our new F_p-estimation algorithm thus gives

$$O(k \log^2(1/\varepsilon') \log\log(1/\varepsilon')) = \mathrm{poly}(\log\log(mM), \log(1/\varepsilon))$$

update time and space

$$O(k\varepsilon'^{-2} \log(mM)) = \tilde{O}(1/\varepsilon^2) \cdot \mathrm{polylog}(mM).$$

Now we discuss multiplicative $(1+\varepsilon)$-approximation of entropy. Let i^* be such that $|x_{i^*}| = \|x\|_\infty$. Define $F_p^{\mathrm{res}} = \sum_{i \neq i^*} |x_i|^p$. The multiplicative approximation algorithm of [28] divides into two cases. In the first case $|x_{i^*}| \geq (5/6) \cdot \|x\|_1$, in which case an algorithm is executed which requires $(1+\varepsilon')$-approximations to both F_1^{res} and F_p^{res} for $k = O(\log(1/\varepsilon))$ different values of $p \in (0,1)$. In the second case $|x_{i^*}| < (2/3) \cdot \|x\|_1$, in which case an algorithm is executed which needs $(1+\varepsilon')$-approximations to F_p for k different values of $p \in (0,1)$. In both cases $\varepsilon' = \Theta(\varepsilon/(k^3 \log(mM)))$. In the case $(2/3) \cdot \|x\|_1 \leq |x_{i^*}| \leq (5/6) \cdot \|x\|_1$, executing either of the two cases yields a correct algorithm.

We can detect which of the two cases holds by using the CountMin sketch (recall that in the case $2/3 \leq |x_{i^*}|/\|x\|_1 \leq 5/6$, classifying as either case is permissible). We do this after performing the reduction to a universe of size $N = \mathrm{poly}(1/\varepsilon)$ described in Section 4, and thus the update time is $O(\log(1/\varepsilon))$ and the space is $O(\log(1/\varepsilon)\log(mM))$. Handling the second case is then straightforward: we run our F_p-estimation algorithm for k values of p to obtain update time $O(k \log^2(1/\varepsilon') \log\log(1/\varepsilon')) = \mathrm{poly}(\log\log(mM), \log(1/\varepsilon))$ and space $O(k\varepsilon'^{-2}\log(mM)) = \tilde{O}(1/\varepsilon^2) \cdot \mathrm{polylog}(mM)$. To handle the first case, we simply run a slightly altered version of our algorithm: we keep all parts of the algorithm the same, but we change our definition of Ψ' in HighEnd to be $\Psi' = \sum_{w \in L \setminus \{i^*\}} (x_w^*)^p$. That is, we do not include the contribution from the heaviest element. The same analysis shows that this gives an estimate of $x_{L \setminus \{i^*\}}$ with additive error $\varepsilon' F_p^{\mathrm{res}}$. Thus again our space and time are the same as when handling the second case.