Perfect Sampling in Turnstile Streams Beyond Small Moments

DAVID P. WOODRUFF, Carnegie Mellon University and Google Research, USA
SHENGHAO XIE, Texas A&M University, USA
SAMSON ZHOU, Texas A&M University, USA

Given a vector $x \in \mathbb{R}^n$ induced by a turnstile stream S, a non-negative function $G : \mathbb{R} \to \mathbb{R}$, a perfect G-sampler outputs an index i with probability $\frac{G(x_i)}{\sum_{j\in[n]} G(x_j)} + \frac{1}{\text{poly}(n)}$. Jayaram and Woodruff (FOCS 2018) introduced a perfect L_p-sampler, where $G(z) = |z|^p$, for $p \in (0, 2]$. In this paper, we solve this problem for $p > 2$ by a sampling-and-rejection method. Our algorithm runs in $n^{1-2/p} \cdot \text{polylog}(n)$ bits of space, which is tight up to polylogarithmic factors in n. Our algorithm also provides a $(1 + \varepsilon)$-approximation to the sampled item x_i with high probability using an additional $\varepsilon^{-2} n^{1-2/p} \cdot \text{polylog}(n)$ bits of space.

Interestingly, we show our techniques can be generalized to perfect polynomial samplers on turnstile streams, which is a class of functions that is not scale-invariant, in contrast to the existing perfect L_p samplers. We also achieve perfect samplers for the logarithmic function $G(z) = \log(1 + |z|)$ and the cap function $G(z) = \min(T, |z|^p)$. Finally, we give an application of our results to the problem of norm/moment estimation for a subset Q of coordinates of a vector, revealed only after the data stream is processed, e.g., when the set Q represents a range query, or the set $n \setminus Q$ represents a collection of entities who wish for their information to be expunged from the dataset.

CCS Concepts: • **Theory of computation → Streaming, sublinear and near linear time algorithms**.

Additional Key Words and Phrases: streaming algorithms, sampling

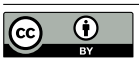
ACM Reference Format:
David P. Woodruff, Shenghao Xie, and Samson Zhou. 2025. Perfect Sampling in Turnstile Streams Beyond Small Moments. *Proc. ACM Manag. Data* 3, 2 (PODS), Article 106 (May 2025), 27 pages. https://doi.org/10.1145/3725243

1 Introduction

As databases manage increasingly larger real-time information, the streaming model of computation has become a crucial setting to analyze algorithms for processing massive, dynamic datasets, such as real-time social media feeds, sensor data for smart cities, live video analytics, detecting and preventing distributed denial of service (DDoS) attacks, and real-time indexing and querying in large-scale databases. In the one-pass streaming model, a frequency vector on an underlying universe $[n]$ is implicitly defined through sequential updates to the coordinates of the vector. These updates can only be observed once and the goal is to aggregate statistics about the frequency vector while using space that is sublinear in the size of both the frequency vector and the data stream.

Sampling items from the dataset is a central and versatile technique for analyzing large-scale datasets. For example, various works have explored sampling methods in the context of big data applications [24, 25, 39, 74], such as virtual network traffic monitoring [34, 41, 49, 67, 72], database management [26, 40, 45–47, 63, 64], distributed computing [28, 30, 54, 73, 77], and data summarization [1, 27, 31, 32, 35, 51, 52, 65, 66]. Formally, there exists an underlying vector $x \in \mathbb{R}^n$, which is

Authors' Contact Information: David P. Woodruff, Carnegie Mellon University and Google Research, Pittsburgh, PA, USA, dwoodruf@cs.cmu.edu; Shenghao Xie, Texas A&M University, College Station, TX, USA, xsh1302@gmail.com; Samson Zhou, Texas A&M University, College Station, TX, USA, samsonzhou@gmail.com.



ACM 2836-6573/2025/5-ART106
https://doi.org/10.1145/3725243

defined through a sequence of m updates. For each $t \in [m]$, an update (i_t, Δ_t) changes coordinate $x_{i_t} \in [n]$ by $\Delta_t \in \{-M, \ldots, M\}$. Therefore, for each $i \in [n]$, the i-th coordinate of the frequency vector x is defined by

$$x_i = \sum_{t \in [m] : i_t = i} \Delta_t.$$

As the updates $\{\Delta_t\}$ are permitted to both increase and decrease coordinates of x, this is called the *turnstile* model of streaming, whereas in the *insertion-only* model, all updates must satisfy $\Delta_t \geq 0$. The goal is then to extract a coordinate $i \in [n]$, possibly along with an estimate of x_i, with probability proportional to $G(x_i)$ for some function G:

Definition 1.1 (G-sampler). Given $x \in \mathbb{R}^n$, $\varepsilon \geq 0$, and a non-negative function $G : \mathbb{R} \to \mathbb{R}$, an ε-approximate G-sampler outputs an index $i^* \in [n]$ with probability at least $\frac{2}{3}$, or otherwise it returns a failure symbol $\perp$. Furthermore, conditioned on $i^* \neq \perp$, we have for each $i \in [n]$:

$$\mathbf{Pr}\left[i = i^*\right] = (1 \pm \varepsilon) \cdot \frac{G(x_i)}{\sum_{j \in [n]} G(x_j)} + n^{-c},$$

where $c > 0$ is a constant input parameter. If $\varepsilon = 0$, we say the sampler is *perfect*.

1.1 Related Work

L_p samplers. The most popular choice for $G(z)$ is the class of functions $G(z) = |z|^p$ for $p > 0$, known as L_p samplers. Introduced by [69], L_p samplers have been used to design streaming algorithms for heavy hitters, L_p norm and F_p moment estimation, cascaded norm approximation, and finding duplicates [5, 18, 27, 55, 56, 58, 69]. For insertion-only streams, the classic technique of reservoir sampling [74] acquires a truly perfect L_1 sample using $O(\log n)$ bits of space, i.e., $\varepsilon = 0$ and furthermore there is no additive $\frac{1}{\text{poly}(n)}$ distortion in the sampling probabilities. However, for either $p \neq 1$ or turnstile streams, the problem becomes significantly more difficult and thus, the existence of sublinear-space L_p samplers was asked by Cormode, Murthukrishnan, and Rozenbaum [29].

The question was first answered in the affirmative by Monemizadeh and Woodruff [69], who gave an ε-approximate L_p sampler for the turnstile model that uses poly $\left(\frac{1}{\varepsilon}, \log n\right)$ bits of space for $p \in [1, 2]$. These L_p samplers were improved first by Andoni, Krauthgamer, and Onak [5] and subsequently by Jowhari, Saglam, and Tardos [58], to use roughly $O\left(\frac{1}{\varepsilon^{\max(1,p)}} \log^2 n\right)$ bits of space for $p \in (0, 2)$ and $O\left(\frac{1}{\varepsilon^2} \log^3 n\right)$ bits of space for $p = 2$. [58] also showed a lower bound of $\Omega(\log^2 n)$ space for $p < 2$ but curiously there were no known lower bounds in terms of ε. This gap was explained by [55], who gave a perfect L_p sampler that uses $\tilde{O}\left(\log^2 n\right)$ bits of space for $p \in (0, 2)$ and $O\left(\log^3 n\right)$ bits of space for $p = 2$. However, it is not immediately clear how to extend their techniques to $p > 2$. Although truly perfect L_p samplers for turnstile streams were ruled out by [56], truly perfect L_p samplers for insertion-only streams were obtained by [56] using $\tilde{O}\left(n^{1-1/p}\right)$ space for $p > 1$ and by [71] using $O(\log n)$ bits of space for $p \in (0, 1)$, albeit in the random oracle model.

Other G-samplers. Other popular choices of $G(z)$ for G-samplers include the logarithmic function $G(z) = \log(1 + z)$, the cap function $G(z) = \min(T, z^p)$ for a threshold T, and the concave sublinear functions $G(z) = \int_0^\infty a(t) \min(1, zt)\, dt$, where $a(t) \geq 0$ [26]. However, significantly less is known about these functions. [56] gave truly perfect G-samplers for monotonically increasing functions $G : \mathbb{R} \to \mathbb{R}_{\geq 0}$ on insertion-only streams with $G(0) = 0$, using space roughly proportional to $\frac{\|x\|_1}{G(X)}$, where $G(X) = \sum_{i \in [n]} G(x_i)$. In particular, their results apply to M-estimators such as the $L_1 - L_2$

estimator $G(z) = 2\left(\sqrt{1+\frac{z^2}{2}}-1\right)$, the Fair estimator $G(z) = \tau|z| - \tau^2\log\left(1+\frac{|z|}{\tau}\right)$, and the Huber estimator $G(z) = \frac{z^2}{2\tau}$ for $|z| \le \tau$ and $G(z) = |z| - \frac{\tau}{2}$ otherwise, where $\tau > 0$ is some constant parameter. [26] approximated the class of concave sublinear functions with the class of soft concave sublinear functions, i.e., $G(z) = \int_0^\infty a(t)(1-e^{-zt})\,dt$ and developed approximate G-samplers on insertion-only streams for soft concave sublinear functions. [71] subsequently developed truly perfect samplers on insertion-only streams for the class of functions

$$G(z) = c\cdot \mathbb{1}[z>0] + \gamma_0 z + \int_0^\infty (1-e^{-tz})\,\nu(dt),$$

which has a bijection with the Laplace exponents of non-negative, one-dimensional Lévy processes. This class of functions includes the L_p samplers $G(z) = z^p$ for $p \in (0,1)$, the soft-cap sampler $G(z) = 1-e^{\tau z}$, and $G(z) = \log(1+z)$. Their truly perfect G-samplers use only two words of memory, but require both the random oracle model and the insertion-only streaming model, and cannot immediately output an estimate of the sampled coordinate x_i.

1.2 Our Contributions

In this paper, we present a number of new techniques and applications for sampling from turnstile streams.

Perfect L_p samplers. We present the first perfect L_p sampler for $p > 2$ for turnstile streams. That is, our algorithm samples a coordinate $i \in [n]$ with probability $\frac{|x_i|^p}{\|x\|_p^p} \pm \frac{1}{\mathrm{poly}(n)}$, where x is defined through a turnstile stream. Formally, our guarantees are:

Theorem 1.2. *For any $p > 2$ and failure probability $\delta \in (0,1)$, there exists a perfect L_p sampler on a turnstile stream that uses $\tilde{O}\left(n^{1-2/p}\log\frac{1}{\delta}\right)$ bits of space and succeeds with probability at least $1-\delta$. Moreover, it obtains a $(1+\varepsilon)$-estimation to the sampled item with probability at least $1-\delta$ using $\tilde{O}\left(\varepsilon^{-2}n^{1-2/p}\log\frac{1}{\delta}\right)$ bits of space.*

By comparison, the existing perfect L_p sampler of [55] handles $p \le 2$. Their techniques rely on duplicating each coordinate $i \in [n]$ a total of $N = n^c$ times for some sufficiently large constant $c > 1$ and then performing a separate scaling of each of the n^{c+1} coordinates. Ultimately these coordinates are then hashed into a CountSketch data structure [22], which has an error roughly on the order of the L_2 norm of the input vector and uses space logarithmic in the universe size. Note that $\log N = O(\log n)$ and thus the algorithm of [55] uses space that is polylogarithmic in n. Generalizing this to $p > 2$ would require an error roughly on the order of the L_p norm of the input vector, which would use space *polynomial* in the universe size. Unfortunately, the universe size after duplication is $N \gg n$, and so the resulting data structure would use space significantly larger than n, which is pointless for our purposes because we could just maintain the entire vector using linear space.

Another point of comparison is the existing truly perfect samplers. The truly perfect sampler of [56] uses space $\tilde{O}\left(n^{1-1/p}\right)$ for $p > 1$, which is prohibitively large for our purposes. On the other hand, the truly perfect sampler of [71] uses $O(\log n)$ bits of space, but can only handle $p < 1$ and requires the random oracle model. Moreover, both of these truly perfect samplers can only be implemented in the insertion-only model, as opposed to the more general turnstile setting of Theorem 1.2. Therefore, we require new techniques in achieving Theorem 1.2.

Approximate L_p samplers. We also present the first approximate L_p sampler for $p > 2$ for turnstile streams, which samples an index $i \in [n]$ with probability $\frac{|x_i|^p}{\|x\|_p^p} \cdot (1 \pm \varepsilon)$, and outputs FAIL with probability at most 0.1. Formally, our guarantees are:

Theorem 1.3. *For any $p > 2$ and accuracy parameter $\varepsilon \in (0, 1)$, there exists an approximate L_p sampler on a turnstile stream that uses $n^{1-2/p} \log^2 n \log \frac{1}{\varepsilon} \cdot \mathrm{poly}(\log \log(n))$ bits of space to run and has update time $\frac{1}{\varepsilon} \cdot \mathrm{polylog}\left(n, \frac{1}{\varepsilon}\right)$. In addition, it gives a $(1+\varepsilon)$-estimation to the sampled item using extra $\frac{1}{\varepsilon^2} n^{1-2/p} \log^2 n \log \frac{1}{\varepsilon} \cdot \mathrm{poly}(\log \log(n))$ bits of space.*

We complete our discussion by providing a sketching lower bound for the approximate L_p samplers, which shows that our algorithm has space optimality in both n and $\log n$ factors.

Theorem 1.4. *Let $x \in \mathbb{R}^n$ be a vector. Suppose that there is a linear sketch that outputs an index $i \in [n]$ with probability $\frac{|x_i|^p}{\|x\|_p^p} \cdot (1 \pm 0.01)$, and outputs FAIL with probability at most 0.1. Then, its sketching dimension is at least $\Omega\left(n^{1-2/p} \log n\right)$.*

A comparison of related work for sampling on data streams and our proposed samplers is displayed in Table 1.

Sampler	Data Stream	Distortion	Randomness	Remarks
[69]	Turnstile	Approximate	Standard	L_p, $p \in [0, 2]$
[5]	Turnstile	Approximate	Standard	L_p, $p \in [0, 2]$
[58]	Turnstile	Approximate	Standard	L_p, $p \in [0, 2]$
[55]	Turnstile	Perfect	Standard	L_p, $p \in [0, 2]$
[26]	Insertion-Only	Approximate	Standard	Soft Concave Sublinear
[56]	Insertion-Only	Truly Perfect	Standard	L_p, $p \geq 1$; M-estimators
[71]	Insertion-Only	Truly Perfect	Random-Oracle Model	L_p, $p \in (0, 1)$; Lévy processes
Our Work	Turnstile	Perfect	Standard	L_p, $p > 2$; polynomials
Our Work	Turnstile	Approximate	Standard	L_p, $p > 2$

Table 1. Summary of related work for sampling on data streams

General samplers. Another consequence of the existing techniques is that known perfect samplers for turnstile streams only handle the class of functions $G(z) = |z|^p$, for which the probability of sampling a coordinate i is $\frac{x_i^p}{\|x\|_p^p} = \frac{(\alpha x_i)^p}{\|\alpha x\|_p^p}$, for any parameter $\alpha > 0$. Thus the probability of sampling $i \in [n]$ from the vector x is the same as the probability sampling $i \in [n]$ from the vector αx for any scalar $\alpha > 0$. Unfortunately, many interesting functions $G(z)$, such as general polynomials, are not scale-invariant. Using our new techniques, we achieve perfect samplers for a wider class of functions:

Theorem 1.5. *For any polynomial $G(z) = \sum_{d \in [D]} \alpha_d z^{p_d}$ with $0 < p_1 < p_2 < \ldots < p_D = p$ and $0 < \alpha_d < M$ for all $d \in [D]$, where M and D are some fixed constants, there exists a perfect polynomial sampler on a general turnstile stream, which outputs $i^* \in [n]$ with $\mathbf{Pr}[i^* = i] = \frac{G(x_i)}{\sum_{j \in [n]} G(x_j)} + \frac{1}{\mathrm{poly}(n)}$. The algorithm uses $\tilde{O}\left(n^{\max\{0, 1-2/p\}} \cdot \log \frac{1}{\delta}\right)$ bits of space and succeeds with probability at least $1 - \delta$.*

Indeed, Theorem 1.5 includes functions such as polynomials that are not scale-invariant. Our techniques can also be used to obtain perfect G-samplers for the cap function $G(z) = \min(T, z^p)$ and the logarithmic function $G(z) = \log(1+z)$. Previously, approximate and perfect samplers for these functions were considered by [26, 71] for the insertion-only setting, but there were no known perfect samplers for the turnstile model. See Table 1 for a summary of these contexts.

Application to norm estimation. Next, we describe an interesting application of our perfect L_p samplers to the problem of norm estimation, where the goal is to estimate $\|x\|_p = \left(x_1^p + \ldots + x_n^p\right)^{1/p}$ up to a $(1+\varepsilon)$-multiplicative factor, for some input accuracy parameter $\varepsilon \in (0,1)$. Note that up to constants, the problem is equivalent to estimating the p-th frequency moment of the vector, defined by $F_p(x) = x_1^p + \ldots + x_n^p$ up to a $(1 + O(\varepsilon))$-multiplicative approximation. The estimation of norms/frequency moments is a fundamental problem for the streaming model. Indeed, since the seminal paper of Alon, Matias, and Szegedy [3], there has been a long line of research analyzing the space or time complexity of this problem [4, 9, 13, 14, 16, 17, 19, 21, 36, 38, 50, 53, 57, 59–61, 75, 78, 79].

More generally, it is often desirable to understand the behavior of certain subsets of coordinates, e.g., iceberg queries in databases, range queries in computational geometry, etc. However, the identity of these coordinates may not be known as an input parameter. Formally, the goal is to estimate $\sum_{i \in Q} x_i^p$, for a subset $Q \subseteq [n]$ that is queried only on the data structure after the stream is processed. In this setting, many of the existing approaches are either suboptimal or fail altogether. For example, it is not clear how to adapt algorithms based on linear sketches to only consider the coordinates in Q. Similarly, approaches based on subsampling and heavy-hitters, e.g., [53] are suboptimal. Our perfect L_p samplers can achieve the following near-optimal guarantees:

Theorem 1.6. *Given $p > 2$, there exists an algorithm that processes a turnstile stream defining a vector $x \in \mathbb{R}^n$ and a post-processing query set $Q \subseteq [n]$, and with probability at least 0.99, outputs a $(1+\varepsilon)$-approximation to $\|x_Q\|_p^p$. For $\|x_Q\|_p^p \geq \alpha \|x\|_p^p$, the algorithm uses $\tilde{O}\left(\frac{1}{\alpha\varepsilon^2} n^{1-2/p}\right)$ bits of space.*

Naively, CountSketch requires roughly $\frac{1}{\alpha^2\varepsilon^2} \cdot n^{1-2/p}$ space to ensure the estimation error is below $\varepsilon \cdot \|x_Q\|_p^p$, to achieve a $(1+\varepsilon)$-approximation to $\|x_Q\|_p^p$. By comparison, our algorithm uses $\frac{1}{\alpha\varepsilon^2} \cdot n^{1-2/p}$ space, which is better by a factor of $\frac{1}{\alpha}$. A more sophisticated approach is given by Proposition 5.1 in [68]. However, this approach can only handle queries on subsets Q with small size, e.g., polylogarithmic in n.

It is known that even for the setting where $Q = [n]$, any $(1+\varepsilon)$-approximation algorithm for F_p estimation on insertion-only streams requires $\Omega\left(\frac{1}{\varepsilon^2} n^{1-2/p}\right)$ bits of space [78]. Thus, our algorithm in Theorem 1.6 is optimal up to polylogarithmic factors in n and $\frac{1}{\varepsilon}$. We also provide further optimizations to achieve fast update time.

A specific application of our setting is the "right to be forgotten data streaming" (RFDS) model, recently introduced by [70]. Motivated by the right of any entity to decide whether their personal data should remain within a specific dataset, the RFDS model permits a forget operation in the data stream, which sets $x_i = 0$ for an input $i \in [n]$. Although [70] shows that the RFDS model is in general difficult, [62] observed that L_p-sampling is useful in the RFDS model, i.e., the streaming model with forget requests, where it put forth the idea of taking $\frac{1}{\alpha\varepsilon^2}$ perfect L_p-samples (in their notation α is replaced with $1-\alpha$) and obtaining unbiased estimates for them using a Taylor series and averaging them. In fact, [62] solves the harder problem, where forget requests can occur multiples times for an item throughout the course of a stream. Here we only allow the coordinates of entities who wish to have their information expunged to be requested after all the data is curated, i.e., at the end of the stream. However, one advantage of only allowing forget requests to occur at the end of the stream is that this allows us to solve the problem in the turnstile streaming model,

whereas [62] shows that if forget requests can occur before the end of the stream then no sublinear space algorithms are possible.

1.3 Motivation and Applications

Statistical indistinguishability. Sampling is a fundamental primitive for extracting key information from large datasets. In particular, L_p sampling has been used as a subroutine toward central data stream problems such as heavy-hitters, norm/moment estimation, cascaded norm estimation, duplicate detection and identification, and data summarization [5, 18, 55, 58, 69]. For example, L_1 sampling is used to extract a number of samples, thus generating a histogram that subsequently serves as a representative summary of the dataset, which is then the input to more complex downstream algorithms [29, 40, 41, 49], such as anomaly/event detection in network monitoring. Since these histograms effectively represent the entire dataset, it is important that these samplers capture the true distribution of the dataset with minimal bias or distortion. Unfortunately, approximate samplers have a relative error in their probabilities, consequently introducing potential statistical bias that propagates through the downstream algorithms. For example, algorithms that assume uniform sampling may experience inaccuracies due to this bias. These biases can be leveraged by a malicious attacker who adaptively queries a database for samples, which is the basis for the field of adversarial robust streaming [2, 6–8, 10–12, 15, 20, 23, 42, 43, 48, 76, 79, 80]. However, perfect L_p samplers mitigate these issues by ensuring near-uniformity in their output distribution without increasing space complexity. In fact, even if we wish to extract n^c samples from the true distribution for any constant $c > 0$, we can set the additive distortion in the sampling probabilities to be $\frac{1}{n^{c+100}}$, so that the resulting total variation distance over the joint distribution of the samples remain statistically indistinguishable from extracting n^c truly uniform samples, making them ideal for black-box applications.

Distributed databases. In particular, an important application of sampling is in distributed databases, where independent samplers are locally implemented on disjoint portions of the dataset across multiple machines, which subsequently serve as compact summaries, providing valuable statistical insights into the distribution of data across the entire system. However, the accumulation of small biases from approximate samplers, manifested as variation distance from the true distribution, can present significant challenges, such as compromising the accuracy of downstream algorithms or sensitive statistical tests that rely on the fidelity of the sampled data. Perfect L_p samplers can address these challenges by ensuring minimal distortion in their output, maintaining the integrity of both local and aggregate statistical summaries.

Privacy considerations. Another compelling motivation for the use of perfect L_p samplers is their role in privacy-preserving applications. In such scenarios, the dataset $x \in \mathbb{R}^n$ is sampled to reveal an index $i \in [n]$ to an external party, while minimizing the leakage of global information about x. Since approximate samplers introduce a multiplicative bias into the sampling probabilities, which may depend on the global properties of the dataset, potential adversaries could exploit this bias to infer sensitive information about x. For example, under such guarantees, it is permissible for a sampler to bias the sampling probabilities for a large set S of indices by $(1 + \varepsilon)$ if a certain global property $\mathcal{P}$ holds for x and might instead bias the sampling probabilities of S by $(1 - \varepsilon)$ if $\mathcal{P}$ does not hold. Then an observer can deduce from a small number of samples whether the property $\mathcal{P}$ holds by estimating the total sampling probabilities of indices in S. On the other hand, perfect L_p samplers produce samples with polynomially small additive bias, reducing the potential for such leakage. This characteristic makes them better suited for privacy-sensitive tasks, where the goal is to reveal minimal information about the underlying data.

Heavy-tailed emphasis. A key advantage of L_p sampler for $p > 2$ is its focus on dominant contributions. When analyzing the vector p-norm $\|x\|_p$ where $p > 2$, heavier emphasis is placed on the coordinates of the vector x that have larger frequencies. This property makes the p-norm particularly useful in scenarios where the focus is on prioritizing elements with larger contributions, such as in sparse signal recovery, outlier detection, and high-dimensional data analysis. More generally, the p parameter can be interpreted as a interpolation between L_0, where all coordinates have the same contribution, regardless of their magnitude, and L_∞, where only the largest coordinate is relevant.

1.4 Preliminaries

In this paper, we use the notation $[n]$ to denote the set $\{1, 2, \ldots, n\}$ for an integer $n \geq 1$. We use the notation $\text{poly}(n)$ to denote a fixed polynomial whose degree can be determined by setting the appropriate constants in the algorithm and analysis. We similarly use the notation $\text{polylog}(n)$ to denote $\text{poly}(\log n)$. We say an event occurs with high probability if it occurs with probability at least $1 - \frac{1}{\text{poly}(n)}$. For a possibly multivariate function f, we use the notation $\tilde{O}(f) = f \cdot \text{polylog}(f)$. For a vector $x \in \mathbb{R}^n$, we define the p-norm of x by $\|x\|_p = \left(x_1^p + \ldots + x_n^p\right)^{1/p}$ and we define the p-th moment of x by $F_p(x) = \|x\|_p^p$.

Khintchine inequality. We recall the following formulation of the Khintchine inequality:

Theorem 1.7 (Khintchine inequality). *[44] Let $r_1, \ldots, r_n \in \{-1, +1\}$ be independent random signs and let $p \geq 2$. Then*

$$\mathbb{E}\left[|r_1 x_1 + \ldots + r_n x_n|^p\right] \leq (B_p)^p \cdot \|x\|_2^p,$$

where $B_p = \sqrt{2} \cdot \left(\frac{1}{\sqrt{\pi}} \cdot \Gamma\left(\frac{p+1}{2}\right)\right)^{1/p}$ and Γ is the Gamma function.

Moreover, we recall the following property of the Gamma function.

Proposition 1.8. $\left(\Gamma\left(\frac{p+1}{2}\right)\right)^{1/p} = \Theta(\sqrt{p})$.

Perfect L_p sampler, $p < 2$. [55] introduced a perfect L_p sampler for $p \in (0, 2]$ with the following guarantees:

Theorem 1.9 (Perfect L_p sampler for $p \leq 2$, c.f. Theorem 9 in [55]). *Given a turnstile data stream S, let $x \in \mathbb{R}^n$ be the vector induced by S. For $p \in (0, 2]$, there exists a perfect L_p sampler with failure probability at most δ_1. Moreover, it outputs an estimate $\widehat{x}$ such that $\widehat{x} = (1 \pm \varepsilon)x_i$ with probability $1 - \delta_2$. The sampler uses*

$$O\left(\left(\log^2 n (\log\log n)^2 + \beta \log n \log(1/\delta_2)\right) \log(1/\delta_1)\right)$$

bits of space for $p < 2$, where $\beta = \min\left\{\varepsilon^{-2}, \varepsilon^{-p} \log(1/\delta_2)\right\}$, and

$$O\left(\left(\log^3 n + \varepsilon^{-2} \log^2 n \log(1/\delta_2)\right) \log(1/\delta_1)\right)$$

bits of space for $p = 2$.

Exponential random variables. Throughout our work, we shall frequently use exponential random variables and a number of their properties. We first define an exponential random variable:

Definition 1.10 (Exponential random variable). If $\mathbf{e}$ is a exponential random variable with scale parameter $\lambda > 0$, then the probability density function for $\mathbf{e}$ is

$$p(x) = \lambda e^{-\lambda x}.$$

We say $\mathbf{e}$ is a standard exponential random variable if $\lambda = 1$.

We define an anti-rank vector to be the ranking of the indices based on their magnitudes, generally in the context of after scaling by exponential random variables. Formally, for a vector $z \in \mathbb{R}^n$ and for $k \in [n]$, we define the k-th anti-rank $D(k) \in [n]$ of z to be the index $D(k)$ so that $|z_{D(1)}| \geq \ldots \geq |z_{D(n)}|$. [55] showed the following characterization of each coordinate $z_{D(k)}$ under the scaling $z_i = \frac{x_i}{\mathbf{e}_i^{1/p}}$, where $\mathbf{e}_i$ is an independent exponential random variable for each $i \in [n]$.

Lemma 1.11 ([55]). *Let $f \in \mathbb{R}^n$ be a vector, let $(\mathbf{e}_1, \ldots, \mathbf{e}_n)$ be i.i.d. exponential random variables with rate 1, let $z_i = x_i/\mathbf{e}_i^{1/p}$, and let $(D(1), \ldots, D(n))$ be the anti-rank vector of the vector $(|z_1|^{-p}, \cdots |z_n|^{-p})$. Then we have*

$$\mathbf{Pr}\left[D(1) = i\right] = \frac{|x_i|^p}{\|x\|_p^p}.$$

As a result, the probability that $|z_i| = \arg\max_j \left\{\left|z_j\right|\right\}$ is precisely $|x_i|^p / \|x\|_p^p$, so for a perfect L_p sampler it suffices to return $i \in [n]$ with $|z_i|$ maximum. Moreover, we have

$$z_{D(k)} = \left(\sum_{i=1}^{k} \frac{E_i}{\sum_{j=i}^{N} f_{D(j)}^p}\right)^{-1/p},$$

where E_i 's are i.i.d. exponential random variables with mean 1, and are independent of the anti-rank vector $(D(1), \ldots, D(n))$.

The next statement shows that the maximum scaled vector is roughly a $\frac{1}{\log^2 n}$-heavy hitter with respect to the entire scaled vector.

Lemma 1.12 ([33]). *Let $\mathbf{e}_1, \ldots, \mathbf{e}_n$ be independent standard exponential random variables, let $\alpha_1, \ldots, \alpha_n \geq 0$, and let $C > 0$ be a fixed constant. Then*

$$\Pr\left[\frac{\max_{i\in[n]} \alpha_i/\mathbf{e}_i}{\sum_{i=1}^n \alpha_i/\mathbf{e}_i} \geq \frac{1}{C\log^2 n}\right] \geq 1 - \frac{1}{\mathrm{poly}(n)}.$$

Now, we state that the coordinate with the largest magnitude can be related to the L_p-norm of the original input vector. It is a corollary of Lemma 1.11.

Lemma 1.13. *Let $f \in \mathbb{R}^n$ be a vector, let $(\mathbf{e}_1, \ldots, \mathbf{e}_n)$ be i.i.d. exponential random variables with rate 1, let $z_i = x_i/\mathbf{e}_i^{1/p}$, and let $(D(1), \ldots, D(n))$ be the anti-rank vector of the vector $(|z_1|^{-p}, \cdots |z_n|^{-p})$. Then, we have that $|z_{D(1)}| > \frac{\|x\|_p}{100\log\frac{1}{\varepsilon}}$ holds with probability $1 - \mathrm{poly}(\varepsilon)$.*

2 Perfect Sampling

In this section, we describe our implementations for both the perfect L_p samplers and the perfect polynomial samplers.

2.1 Integer p

We first provide the intuition for our perfect L_p sampler for integer $p > 2$. A natural starting point would be to adapt the techniques for existing perfect L_p samplers with $p \in (0, 2]$. The only existing implementation requires duplicating each coordinate a polynomial number of times to utilize the max-stability property of exponential random variables. To identify the maximum coordinate, [55] only needs polylogarithmic space to find the L_2-heavy hitters, but to find the L_p-heavy hitters for $p > 2$, the space required is polynomial in the universe size, which is now substantially large due to the duplication.

Instead, we use perfect L_2 samplers as a black-box subroutine to extract a coordinate $i \in [n]$. We would like to output i with probability $\frac{x_i^p}{\|x\|_p^p} + \frac{1}{\text{poly}(n)}$ and we have sampled i with probability roughly $\frac{x_i^2}{\|x\|_2^2} + \frac{1}{\text{poly}(n)}$. Thus, conditioned on the L_2 sampler outputting i, we would like to output i with probability $x_i^{p-2} \cdot \frac{F_2(x)}{F_p(x)}$. Unfortunately, this expression may not be a well-defined probability because it may be larger than 1, for instance if $x_i = n^{1/p}$ and $F_2(x) = F_p(x) = \Theta(n)$. Therefore, we instead would like to only output i with probability $x_i^{p-2} \cdot \frac{F_2(x)}{n^{1-2/p} F_p(x)}$. Although we do not have each of the terms x_i^{p-2}, $F_2(x)$, and $F_p(x)$, we can obtain constant-factor approximations to $F_2(x)$ and $F_p(x)$ using existing procedures [3, 37]. It thus remains to estimate x_i^{p-2}.

In fact, the index returned by the perfect L_2 sampler of [55] is the largest scaled coordinate $\frac{x_i}{\sqrt{\mathbf{e}_i}}$, where $\mathbf{e}_i$ is an independent exponential random variable for all $i \in [n]$ and in particular, a heavy-hitter of the resulting scaled vector. We can thus acquire an unbiased estimate $\widehat{x_i^{p-2}}$ to x_i^{p-2} by running $(p-2)$ independent instances of CountSketch on the scaled coordinates, which has a small relative variance since $\frac{x_i}{\sqrt{\mathbf{e}_i}}$ is a heavy-hitter. However, this is still not enough because there is a non-trivial probability that $\widehat{x_i^{p-2}} \cdot \frac{F_2(x)}{F_p(x)}$ is still larger than 1 if the estimate is procured through $(p-2)$ instances of CountSketch alone. Hence we further show that by the Khintchine inequality, a sufficiently tight approximation of x_i^{p-2} can be obtained using $\text{polylog}(n)$ instances of CountSketch. Finally, we show that with $n^{1-2/p} \cdot \text{polylog}(n)$ number of perfect L_2 samples, one of these samples will be passed through the subsequent rejection sampling. Our algorithm appears in full in Algorithm 1.

Algorithm 1 Perfect L_p sampler for integer $p > 2$

Input: Input vector $x \in \mathbb{R}^n$ in a stream
Output: Perfect L_p sample

Let C be the constant from Corollary 2.3 and let $N = O\left(n^{1-2/p}\right)$
$s_1, \ldots, s_N$ be N perfect L_2 samples from x ▷See Theorem 1.9
Use AMS to get a 2-approximation $\widehat{F_2}$ to $F_2(x)$
Use FpEst to get a 2-approximation $\widehat{F_p}$ to $F_p(x)$
for each $i \in [N]$ **do**
    Let j be the index of s_i
    for $a \in [p-2]$ **do**
        Run $\text{polylog}(n)$ instances of CountSketch to acquire estimates $\widehat{x_j^{(q,1)}}, \ldots, \widehat{x_j^{(q,\text{polylog}(n))}}$
        $\widehat{x_j^{(q)}} \leftarrow \frac{1}{\text{polylog}(n)} \sum_{l \in [\text{polylog}(n)]} \widehat{x_j^{(q,l)}}$
    return j with probability $\frac{\widehat{F_2}}{8n^{1-2/p} \cdot \widehat{F_p}} \cdot \prod_{a \in [p-2]} \left|\widehat{x_j^{(a)}}\right|$
    Otherwise, continue to next i

To analyze our algorithm, we first show that the magnitude of a signed sum of coordinates can be bounded in terms of the L_2 of the vector with high probability.

Lemma 2.1. *Let m be some large constant to be determined later. For all $l \in [\mathrm{polylog}(n)]$, let $r_1^{(l)}, \ldots, r_n^{(l)} \in \{-1, +1\}$ be independent random signs. Then we have,*

$$\left| \frac{1}{\log^m(n)} \cdot \sum_{l \in [\log^m(n)]} r_1^{(l)} x_1 + \ldots + r_n^{(l)} x_n \right| \leq \frac{1}{\log^{m/4}(n)} \cdot \|x\|_2.$$

with probability at least $1 - \frac{1}{\mathrm{poly}(n)}$.

Proof. Let m be a large enough constant, by the linearity of expectation, we have

$$\mathbb{E}\left[\left| \frac{1}{\log^m(n)} \cdot \sum_{l \in [\log^m(n)]} r_1^{(l)} x_1 + \ldots + r_n^{(l)} x_n \right|^p \right] = \frac{1}{\log^{mp}(n)} \cdot \mathbb{E}\left[\left| \sum_{l \in [\log^m(n)]} r_1^{(l)} x_1 + \ldots + r_n^{(l)} x_n \right|^p \right].$$

Then, by Theorem 1.7 and Property 1.8, we have that

$$\begin{aligned} \mathbb{E}\left[\left| \frac{1}{\log^m(n)} \cdot \sum_{l \in [\log^m(n)]} r_1^{(l)} x_1 + \ldots + r_n^{(l)} x_n \right|^p \right] &\leq \frac{1}{\log^{mp}(n)} \cdot (B \cdot \sqrt{p})^p \cdot (\log^m(n) \cdot \|x\|_2^2)^{p/2} \\ &\leq \frac{1}{\log^{(m/2)\cdot p} n} (B \cdot \sqrt{p})^p \cdot \|x\|_2^p. \end{aligned}$$

for some absolute constant $B > 0$. Then for any constant $c > 0$, we have by Markov's inequality,

$$\mathbf{Pr}\left[\left| \frac{1}{\log^m(n)} \cdot \sum_{l \in [\log^m(n)]} r_1^{(l)} x_1 + \ldots + r_n^{(l)} x_n \right|^p \geq \frac{n^c}{\log^{(m/2)\cdot p} n} \cdot (B \cdot \sqrt{p})^p \cdot \|x\|_2^p \right] \leq \frac{1}{\mathrm{poly}(n)}.$$

Taking for $p = \log(n)$, we have our desired result. □

As a result, it follows that CountSketch can be used to give a good additive estimate $\widehat{x_i}$ to each coordinate x_i.

Corollary 2.2. *For each $i \in [n]$, the mean of $\log^m(n)$ instances of CountSketch gives an estimate $\widehat{x_i}$ such that with probability at least $1 - \frac{1}{\mathrm{poly}(n)}$*

$$\max_{i \in [n]} |\widehat{x_i} - x_i| \leq \frac{1}{\log^{m/4}(n)} \cdot \|x\|_2.$$

Similarly, we can bound the error of the estimated value of the sampled coordinate.

Corollary 2.3. *For the index i output by a perfect L_2 sample, we have an estimate $\widehat{x_i}$ such that with probability at least $1 - \frac{1}{\mathrm{poly}(n)}$,*

$$|\widehat{x_i} - x_i| \leq \frac{1}{\mathrm{polylog}(n)} \cdot |x_i|.$$

Proof. First, we describe how we estimate the sampled entry. In the L_p sampler introduced by [55], for a vector x, we scale each entry by $g_i := \frac{x_i}{\mathbf{e}_i^{1/2}}$ where $\mathbf{e}_i$'s are i.i.d. exponential random variables with rate 1, and we find the maximum entry of g. Notice that we have the following property of exponential variables (c.f. Lemma 1.12):

$$\mathbf{Pr}\left[\max g_i^2 \geq \frac{1}{C \log^2 n} \|g\|_2^2 \right] \geq 1 - \frac{1}{\mathrm{poly}(n)}.$$

Therefore, we implement CountSketch on the scaled vector g to get an estimation of the sampled item g_i. By Corollary 2.2, it has error at most $\frac{1}{\log^{m/4}(n)} \cdot \|g\|_2$ with probability at least $1 - \frac{1}{\mathrm{poly}(n)}$. Hence, the error is bounded by $\frac{1}{\mathrm{polylog}(n)} \cdot g_i$ with probability at least $1 - \frac{1}{\mathrm{poly}(n)}$ for sufficiently large m. We output $g_i \cdot \mathbf{e}_i$ as an estimation to x_i, so that the error is bounded by $\frac{1}{\mathrm{polylog}(n)} \cdot x_i$ with probability at least $1 - \frac{1}{\mathrm{poly}(n)}$. □

We now show that our algorithm returns a random coordinate under the correct probability distribution for L_p sampling.

Lemma 2.4. *With high probability, Algorithm 1 outputs an index $i \in [n]$. Moreover, for each $j \in [n]$, we have that*

$$\mathbf{Pr}\left[i = j\right] = \frac{|x_j|^p}{\|x\|_p^p} \pm \frac{1}{\mathrm{poly}(n)}.$$

Proof. For each $i \in [N]$, we sample $j \in [n]$ with probability $\frac{|x_j|^2}{\|x\|_2^2} \pm \frac{1}{\mathrm{poly}(n)}$ by the correctness of perfect F_2 sampling. For each $a \in [p-2]$, $\widehat{x_j^{(a)}}$ is an unbiased estimate of x_j such that with high probability, $|x_j - \widehat{x_j^{(a)}}| \leq \frac{1}{\mathrm{polylog}(n)} \cdot |x_j|$. Then the index j is returned with probability

$$\left(\frac{|x_j|^2}{\|x\|_2^2} \pm \frac{1}{\mathrm{poly}(n)}\right) \cdot \frac{\widehat{F_2}}{N \cdot \widehat{F_p}} \cdot \prod_{a \in [p-2]} \widehat{x_j^{(a)}}.$$

Let $\mathcal{E}_1$ be the event that $|x_j - \widehat{x_j^{(a)}}| \leq \frac{1}{\mathrm{polylog}(n)} \cdot |x_j|$ for all $a \in [p-2]$ so that by Corollary 2.3 and a union bound, $\mathbf{Pr}\left[\mathcal{E}_1\right] \geq 1 - \frac{1}{\mathrm{poly}(n)}$. Conditioned on $\mathcal{E}_1$, we have that

$$\prod_{a \in [p-2]} \widehat{x_j^{(a)}} = \prod_{a \in [p-2]} x_j \cdot \left(1 \pm \frac{1}{\mathrm{polylog}(n)}\right).$$

Let $\mathcal{E}_2$ be the event that AMS and FpEst return 2-approximations of $\|x\|_2^2$ and $\|x\|_p^p$ respectively. Conditioned on $\mathcal{E}_1$ and $\mathcal{E}_2$, then we have that

$$\begin{aligned}
\frac{\widehat{F_2}}{8n^{1-2/p} \cdot \widehat{F_p}} \cdot \prod_{a \in [p-2]} |\widehat{x_j^{(a)}}| &\leq \frac{|x_j|^{p-2} \cdot \|x\|_2^2}{n^{1-2/p} \cdot \|x\|_p^p} \\
&\leq \frac{|x_j|^{p-2} \cdot \|x\|_2^2}{n^{1-2/p} \cdot (|x_1|^p + \ldots + |x_n|^p)^{1-2/p} \cdot \|x\|_p^2} \\
&\leq \frac{|x_j|^{p-2} \cdot \|x\|_2^2}{n^{1-2/p} \cdot |x_j|^{p-2} \cdot \|x\|_p^2} \leq 1.
\end{aligned}$$

Therefore, our rejection probability is smaller than 1, so it is well-defined. Moreover, we have $\mathbb{E}\left[\widehat{x_j^{(a)}}\right] = x_j$ for all $a \in [p-2]$. Notice that $x_j^{(a)}$ has the same sign as x_j conditioned on $\mathcal{E}_1$, we have $\mathbb{E}\left[|\widehat{x_j^{(a)}}|\right] = |x_j|$. Thus in expectation, the probability that j is returned is

$$\frac{|x_j|^p}{8n^{1-2/p} \cdot \|x\|_p^p} \pm \frac{1}{\mathrm{poly}(n)}.$$

Hence, conditioned on some index being output, the probability that j is returned is

$$\left(\frac{|x_j|^p}{8n^{1-2/p}\cdot\|x\|_p^p}\pm\frac{1}{\mathrm{poly}(n)}\right)\cdot\left(\sum_{\ell\in[n]}\frac{|x_\ell|^p}{8n^{1-2/p}\cdot\|x\|_p^p}\pm\frac{1}{\mathrm{poly}(n)}\right)^{-1}$$
$$=\frac{|x_j|^p}{\|x\|_p^p}\pm\frac{1}{\mathrm{poly}(n)},$$

which is the correct probability distribution. Finally, the probability that some index being output for each sample $i\in[N]$ is

$$\sum_{j\in[n]}\frac{|x_j|^p}{8n^{1-2/p}\cdot\|x\|_p^p}\pm\frac{1}{\mathrm{poly}(n)}\geq\frac{1}{16n^{1-2/p}}.$$

Thus by repeating $N=O\left(n^{1-2/p}\right)$ times, we have that a sample is returned with probability at least $1-\frac{1}{\mathrm{poly}(n)}$. □

Next, we analyze the space complexity of our algorithm.

Lemma 2.5. *Algorithm 1 uses $n^{1-2/p}\cdot\mathrm{polylog}(n)$ bits of space.*

Proof. For each L_2 sampler, we need its failure probability δ_1 to be $\frac{1}{\mathrm{poly}(n)}$, so that by a union bound, all L_2 samplers succeed with probability $1-\frac{1}{\mathrm{poly}(n)}$. Then, by Theorem 1.9, we uses $N\cdot O\left(\log^4 n\right)=O\left(n^{1-2/p}\cdot\log^4 n\right)$ to acquire the L_2 samples. Moreover, we implement $\mathrm{polylog}(n)$ instances of CountSketch to estimate the value of each sample, which uses $N\cdot\mathrm{polylog}(n)$ bits of space. Notice that the space consumption of AMS, FpEst and the $p-2$ instances of CountSketch is dominated by the L_2 samplers. Thus, we have the stated space complexity. □

Putting together Lemma 2.4 and Lemma 2.5, we obtain the full guarantees for our perfect L_p sampler for integer $p>2$.

Theorem 2.6. *Given an integer $p>2$, there exists a perfect L_p sampler that uses $\tilde{O}\left(n^{1-2/p}\right)$ bits of space.*

2.2 Fractional p

Next, we generalize our results from integer $p>2$ to general $p>2$. The main challenge in the previous techniques is that to estimate x_i^{p-2} for integer $p>2$, an intuitive approach would be to acquire $p-2$ independent estimates for x_i. To estimate x_i^{p-2} for fractional $p>2$, we utilize the Taylor series expansion of x_i^{p-2}. Our full algorithm appears in Algorithm 2.

The next lemma shows that a Taylor series expansion truncated at $Q=O(\log n)$ terms is a good approximation to x^p for any fixed constant $p>2$.

Lemma 2.7. *Let $Q=O(\log n)$ and $y_i\in\left[\frac{99x_i}{100},\frac{101x_i}{100}\right]$. Then*

$$\left|x_i^p-\sum_{q=0}^{Q}\binom{p}{q}y_i^{p-q}(x_i-y_i)^q\right|\leq\frac{1}{\mathrm{poly}(n)}\cdot x_i^p.$$

With the guarantee of the truncated Taylor series, we claim that our estimate of the sampled item from the L_2 sampler still ensures a $1+\frac{1}{\mathrm{polylog}(n)}$-multiplicative error. And thus, our algorithm produces a sample i according to the correct probability distribution. The analysis is similar to the case of integer p, so we defer it to our full version.

Algorithm 2 Perfect L_p sampler for general $p > 2$

Input: Input vector $x \in \mathbb{R}^n$ in a stream
Output: Perfect L_p sample

Let $Q = O(\log n)$ with large enough constant term
Let $N' = n^{1-2/p} \cdot \text{polylog}(n)$. Let $N = N' \cdot O(\log n)$
$s_1, \ldots, s_N$ be N perfect L_2 samples from x ▷See Theorem 1.9
for each $i \in [N]$ **do**
    Let y_{s_i} be the constant approximation to x_{s_i} satisfying $y_{s_i} \in [\frac{99x_{s_i}}{100e^p \log n}, \frac{101x_{s_i}}{100e^p \log n}]$
Use AMS to get a 2-approximation $\widehat{F_2}$ to $F_2(x)$
Use FpEst to get a 2-approximation $\widehat{F_p}$ to $F_p(x)$
for each $i \in [N]$ **do**
    Let j be the index of s_i
    for $q \in [Q]$ **do**
        Run polylog(n) instances of CountSketch to acquire estimates $\widehat{x_j^{(q,1)}}, \ldots, \widehat{x_j^{(q,\text{polylog}(n))}}$
        $\widehat{x_j^{(q)}} \leftarrow \frac{1}{\text{polylog}(n)} \sum_{l \in [\text{polylog}(n)]} \widehat{x_j^{(q,l)}}$
    $\widehat{x_j^{p-2}} \leftarrow \sum_{q=0}^{Q} \left(\binom{p-2}{q} y_j^{p-2-q} \cdot \prod_{a \in [q]} (\widehat{x_j^{(a)}} - y_j) \right)$ ▷See Lemma 2.7
    return j with probability $\frac{\widehat{F_2}}{4N' \cdot \widehat{F_p}} \cdot \left| \widehat{x_j^{p-2}} \right|$
    Otherwise, continue to next i

Lemma 2.8. *With high probability, Algorithm 2 outputs an index $i \in [n]$. Moreover, for each $j \in [n]$, we have that*

$$\mathbf{Pr}[i = j] = \frac{|x_j|^p}{\|x\|_p^p} \pm \frac{1}{\text{poly}(n)}.$$

Next, we state the formal guarantee of our perfect L_p sampler for fractional $p > 2$.

Theorem 2.9. *Given $p > 2$, there exists a perfect L_p sampler that uses $\tilde{O}\left(n^{1-2/p}\right)$ bits of space. In addition, it gives an $(1+\varepsilon)$-estimate to the sampled item using extra $\tilde{O}\left(\varepsilon^{-2} n^{1-2/p}\right)$ bits of space.*

Proof. By Theorem 1.9, the L_2 sampler gives an $(1+\varepsilon)$-estimate to the sampled item with high probability using $O\left(\varepsilon^{-2} \log^4 n\right)$ bits of space. Our space bound follows from the fact that we have $\tilde{O}\left(n^{1-2/p}\right)$ L_2 samplers. □

2.3 Perfect Polynomial Sampler

In this section, we further generalize our results from perfect L_p samplers to the following notion of perfect polynomial samplers:

Definition 2.10 (Perfect polynomial sampler). Let polynomial $G(z) = \sum_{d \in [D]} \alpha_d |z|^{p_d}$ with $0 < \alpha_d < M$ for all $d \in [D]$, where M and D are some fixed constants. Given a vector $x \in \mathbb{R}^n$, a perfect polynomial sampler reports an index $i^* \in [n]$ such that for each $i \in [n]$, we have

$$\mathbf{Pr}[i^* = i] = \frac{G(x_i)}{\sum_{j \in [n]} G(x_j)} + \frac{1}{\text{poly}(n)}.$$

Note that parameters D and M are considered constants in our setting.

Similar to the above approach, we acquire an unbiased estimate to $\frac{G(x_i)}{x_i^p}$ for an index x_i acquired from perfect L_p sampling. This is done by applying the truncated Taylor series estimator to $x_i^{p_d-p}$. We can then choose to accept the sampled index with a probability that must be well-defined, regardless of the sampled index $i \in [n]$. Our full algorithm appears in Algorithm 3.

Algorithm 3 Perfect polynomial sampler for polynomial with degree at most p

Input: Input vector $x \in \mathbb{R}^n$ in a stream, polynomial $G_p(u) = \sum_{d\in[D]} \alpha_d |u^{p_d}|$ with $0 < p_1 < p_2 < \ldots < p_d = p$ and $0 < \alpha_d < M$ for all $d \in [D]$

Output: Perfect p-polynomial sample

1: Let $N = O(\log n)$
2: $s_1, \ldots, s_N$ be N perfect L_p samples from x ▹See Theorem 1.9 and Theorem 2.9
3: **for** each $i \in [N]$ **do**
4: Let j be the index of s_i
5: **for** each $d \in [D]$ **do**
6: Let $\widehat{x_j^{p_d-p}}$ be an unbiased $(1 + \frac{1}{\log(n)})$-estimate of $x_j^{p_d-p}$ ▹See Theorem 2.9
7: **return** j with probability $\frac{1}{5DM} \cdot \sum_{d\in[D]} \alpha_d \left|\widehat{x_j^{p_d-p}}\right|$
8: Otherwise, continue to next i

Now, we state the formal theorem statement of our perfect polynomial sampler.

Theorem 2.11 (Perfect polynomial sampler). *There exists a perfect polynomial sampler for a polynomial of degree at most p, which uses $\tilde{O}\left(n^{1-2/p}\right)$ bits of space. In addition, it gives an $(1+\varepsilon)$-estimate $\widehat{x_i}$ to the sampled item using extra $\tilde{O}\left(\varepsilon^{-2}n^{1-2/p}\right)$ bits of space.*

3 Approximate L_p Sampler for $p > 2$ with Fast Update Time

In this section, we present an approximate L_p sampler with optimal space dependence on n and $\log n$, which achieves fast update time. We give our algorithm in Algorithm 4.

As we mentioned before, the existing implementation of the perfect L_p sampler requires duplicating each coordinate a polynomial number of times. This is because of the adversarial error we encounter when we condition on some specific index achieving the max. In the implementation, we fail the sampler if we do not witness a large gap between our estimate of the max and second max, since we cannot distinguish them from our CountSketch estimation with additive error. However, the failure probability may change drastically if we conditioned on different indices achieving the max. For example, consider vector $x = (100n, 1, \ldots, 1) \in \mathbb{R}^n$. We would expect the first index to be the maximum in the scaled vector. And if we condition on the second index achieving the max, we would expect the max and the second max to not have a large gap. Then, the failure probability shifts by an additive constant which leads to an incorrect sampling distribution. Therefore, we duplicate each entry polynomial times so that there are no heavy-hitters in the resulting vector. This would reduce the dependency on the anti-ranks to a negligible small value. Below, we state the formal definition of the duplication vector.

Definition 3.1 (Duplication). Let X denote the duplication vector where $X_{i,j} := x_i$ for all $i \in [n], j \in [n^c]$. Notice that $\|X\|_p^p = n^c \cdot \|x\|_p^p$.

It is challenging to adapt this duplication method to the $p > 2$ setting since we are not able to maintain a CountSketch table with $n^{(c+1)^{1-2/p}}$ buckets. On the other hand, we cannot implement CountSketch with a lower number of buckets or the additive error would be too high. Therefore,

Algorithm 4 Approximate L_p Sampler for $p > 2$ with fast-update

Input: Input vector $x \in \mathbb{R}^n$ in a stream, accuracy ε, discretization factor η
Output: Approximate L_p sampler with accuracy ε

1: Let $c \leftarrow \Theta(1)$ be sufficiently large
2: For each $i \in [n]$, $j \in [n^c]$, generate exponential random variables $\mathbf{e}_{i,j}^{1/p}$
3: For each $i \in [n]$, let $\mathbf{v}_i = \max_{j\in[n^c]} |x_i| \cdot \mathrm{rnd}_\eta(1/\mathbf{e}_{i,j}^{1/p})$
4: Let $\mathbf{u} \in \mathbb{R}^{n^{c+1}}$ be vector consisting of $|x_i| \cdot \mathrm{rnd}_\eta(1/\mathbf{e}_{i,j})$ for all $i \in [n]$, $j \in [n^c]$
5: Let $\bar{\mathbf{u}}$ be $\mathbf{u}$ with the entries of $\mathbf{v}$ zeroed out
6: Keep a CountSketch$_1$ with $\Theta(\log n)$ rows and $n^{1-2/p} \cdot \log \frac{1}{\varepsilon}$ buckets on $\mathbf{v}$
7: Let $v \in \mathbb{R}^n$ be the estimated frequencies by the resulting CountSketch$_1$ table
8: For each item v_i in v, add v_i to set B if its absolute value is bigger than $\frac{n^{c/p}\|x\|_p}{200 \log \frac{1}{\varepsilon}}$
9: Return FAIL if B is empty
10: Keep a CountSketch$_2$ with $O(\log n)$ rows and the first $|B|$ of $(n^{c+1})^{1-2/p}$ buckets on $\bar{\mathbf{u}}$
11: Add the entries in set B of CountSketch$_1$ to CountSketch$_2$
12: Let $y \in \mathbb{R}^n$ be the estimated frequencies by the resulting CountSketch$_2$ table
13: Let $i^* = \mathrm{argmax}_{i\in[n]} |y_i|$
14: Let $R_{\mathbf{u}}$ be an estimation of $\|\mathbf{u}\|_2$ satisfying $R \in [\frac{\|\mathbf{u}\|_2}{2}, 2\|\mathbf{u}\|_2]$
15: Let $\mu \in [\frac{1}{2}, \frac{3}{2}]$ be a uniform random variable
16: Return y_{i^*} if $|y_{D(1)}| - |y_{D(2)}| > \frac{100R}{\mu n^{(c+1)(1/2-1/p)}}$. Otherwise, return FAIL

we introduce a simulation scheme by keeping a two-stage CountSketch table. Consider an index i, we generate n^c exponential variables $\mathbf{e}_{i,j}$, we observe the maximum of the scaled vector can only be obtained from all $x_i/\mathbf{e}_{i,j^*}^{1/p}$, where $j^* = \mathrm{argmax}_{j\in[n^c]} x_i/\mathbf{e}_{i,j}^{1/p}$.

Thus, in our first-stage CountSketch, we only record the maximum item $\mathbf{w}_i = x_i/\mathbf{e}_{i,j^*}^{1/p}$ of each entry. We select a set B with roughly polylog $\frac{1}{\varepsilon}$ indices that contains the maximum of $\mathbf{w}$ with probability $1-\varepsilon$. Then, in our second stage, we maintain a CountSketch table for the scaled vector $\mathbf{z}$ with $\mathbf{w}$ zeroed out. We only record the first $|B|$ buckets, and we discard everything that hashes into other buckets. We add up the entries in the two CountSketch tables to obtain an estimate for each item in B. Then, we do a statistical test to fail the instances that do not have anti-concentration. Notice that since we hash the items uniformly randomly, this simulation will not change the distribution of our estimation. In this way, we implement the approximate sampler with a small space occupation.

However, upon each arrival in the stream, if we calculate each of the n^c duplicated scaled entries explicitly in the above simulation method, the update time would be $O(n^c)$, which is prohibitively large in practice. To speed up the update time, instead of calculating the explicit value of $\frac{X_i}{\mathbf{e}_i^{1/p}}$, we scale X_i by $\mathrm{rnd}_\eta(1/\mathbf{e}^{1/p})$, where $\mathrm{rnd}_\eta(x)$ rounds x down to the nearest power of $(1+\eta)^q$ for $q \in \mathbb{Z}$. Therefore, we maintain a different vector $\mathbf{u} \in \mathbb{R}^{n^{(c+1)}}$ in our CountSketch table, where $\mathbf{u}_i = X_i \cdot \mathrm{rnd}_\eta(1/\mathbf{e}^{1/p})$. Notice that $\mathbf{u}_i = \mathbf{z}_i \cdot (1 \pm O(\eta))$ for all $i \in [n^{c+1}]$, where $\mathbf{z}_i = X_i/\mathbf{e}^{1/p}$ is the scaled entry without round-up. Thus, this gives us an approximate L_p sampler with accuracy $O(\eta)$.

Last, we state a modification to the classic CountSketch algorithm introduced by [55], which is used to reduce the dependency on the anti-ranks. Let $A \in \mathbb{R}^{d\times l}$ be a $d\times l$ CountSketch matrix. Instead of uniformly hashing each item into a bucket in each row, we generate variables $h_{i,j,k} \in \{0,1\}$ for $(i,j,k) \in [d]\times[l]\times[n]$, where $h_{i,j,k}$ are all i.i.d. and equal to 1 with probability $1/l$. We also let

$g_{i,k} \in \{1, -1\}$ be i.i.d. Rademacher variables (1 with probability 1/2). Then $A_{i,j} = \sum_{k=1}^{n} x_k g_{i,k} h_{i,j,k}$, and the estimate y_l of x_k is given by:

$$y_k = \text{median}\left\{g_{i,k} A_{i,j} \mid h_{i,j,k} = 1\right\}$$

Thus, it performs the same as hashing the item to k random buckets in the whole CountSketch table. Note the element f_k can be hashed into multiple buckets in the same row of A, or even be hashed into none of the buckets in a given row. We remark that the error bound of the original CountSketch algorithm can be applied as usual (see Section A.1 of [55] for detailed analysis).

Next, we demonstrate the correctness of our sampler. We present a sketch of the proof here due to limited space. For the whole proof, one can refer to our complete version.

Analysis of CountSketch$_1$. We recall that in CountSketch$_1$, for each index $k \in [n]$, we select the maximum $1/\mathbf{e}_{k,j^*}$ of $1/\mathbf{e}_{k,j}$, $j \in [n^c]$. Then, we implement countsketch on the resulting scaled vector $\mathbf{v}_k = x_k/\mathbf{e}_{k,j^*}$. Now, we show that the CountSketch$_1$ table recovers the maximum index of the scaled vector with high probability.

First, we state Bernstein's inequality.

Theorem 3.2 (Bernstein's inequality). *Let $Y_1, \ldots, Y_n$ be independent random variables such that $|Y_i| \le M$ for all $i \in [n]$, and $Var(\sum_i Y_i) \le \sigma^2$. Then there exist constants $C_1, C_2 > 0$ such that for all $t > 0$,*

$$\mathbf{Pr}\left[\sum_i Y_i - \mathbb{E}\left[\sum_i Y_i\right] > t\right] \le C_1\left(e^{-C_2 t^2/\sigma^2} + e^{-C_2 t/M}\right).$$

By Bernstein's inequality, we are able to bound the number of large items in the scaled vector.

Lemma 3.3. *Let C be a constant. We call an index $k \in [n]$* large *if $\mathbf{v}_k \ge \frac{n^{c/p}\|x\|_p}{C \log \frac{1}{\varepsilon}}$. Let $\mathcal{E}$ be the event that the number of large indices is at most $2C^p \log^{p+1} \frac{1}{\varepsilon}$, we have $\mathbf{Pr}[\mathcal{E}] \ge 1 - \frac{1}{\text{poly}(n)}$.*

With the above result, we upper bound the error of the first CountSketch table.

Lemma 3.4. *With probability at least $1 - \frac{1}{\text{poly}(n)}$, the error of CountSketch$_1$ with $n^{1-2/p} \cdot \log \frac{1}{\varepsilon}$ buckets and $\Theta(\log n)$ rows is at most $\frac{n^{c/p}\|x\|_p}{400 \log \frac{1}{\varepsilon}}$.*

Using Lemma 1.13, we show the following lemma, stating that the maximum entry in the scaled vector is $\frac{1}{\log \frac{1}{\varepsilon}}$-heavy compared to the p-th norm of the unscaled vector.

Lemma 3.5. *We have $\max_{i \in [n]} |\mathbf{v}_i| \ge \frac{n^{c/p}\|x\|_p}{100 \log \frac{1}{\varepsilon}}$ with probability $1 - \text{poly}(\varepsilon)$.*

Combining Lemma 3.4 and Lemma 3.5, we have the correctness of CountSketch$_1$.

Lemma 3.6. *We recover the maximum index $i^* = \text{argmax}\, \mathbf{v}_i$ in CountSketch$_1$ with probability $1 - \text{poly}(\varepsilon) - \frac{1}{\text{poly}(n)}$.*

Analysis of CountSketch$_2$. We next prove the correctness of CountSketch$_2$. As we mentioned in Section 3, CountSketch$_2$ fails the statistical test with constant probability, we need to bound the dependency of this probability on which index achieves the maximum.

The following lemma shows that the dependency of CountSketch$_2$ on the anti-ranks only causes a linear shift of the failure probability that is negligible. The proof is analogous to the analysis of Lemma 12 in [55].

Lemma 3.7. *Let ¬FAIL denote the event that CountSketch$_2$ does not fail. Suppose $\eta > n^{-c}$, we have $\mathbf{Pr}[\neg\text{FAIL} \mid D(1)] = \mathbf{Pr}[\neg\text{FAIL}] \pm O\left(\eta\sqrt{\log n}\right)$.*

The next statement upper bounds the failure probability of the second-stage CountSketch.

Lemma 3.8. *Let* ¬FAIL *be the event that* CountSketch$_2$ *does not fail. Then* $\mathbf{Pr}[\neg\text{FAIL}] = \Omega(1)$.

Based on the above results, we show that our approximate sampler has the correct sampling distribution.

Theorem 3.9. *For* $\eta = \frac{O(\varepsilon)}{\sqrt{\log n}}$, *Algorithm 4 outputs an index* $i \in [n]$ *such that for each index* $j \in [n]$, *we have*

$$\mathbf{Pr}[i = j] = \frac{|f_j|^p}{\|X\|_p^p}(1 \pm \varepsilon) \pm \frac{1}{\text{poly}(n)},$$

and outputs FAIL with probability at most $O(1) < 1$.

Proof. First, we show whenever some index i^* is reported, it satisfies $i^*, j^* = \text{argmax}_{i\in[n], j\in[n^c]} |x_i/\mathbf{e}_{i,j}|$. Due to our statistical test, we have $y_{(1)} - y_{(2)} > \frac{100\mu R}{n^{(c+1)(1/2-1/p)}}$. Then the gap between the estimations of the top two coordinates in $\mathbf{v}$ is at least 50 times the CountSketch error. This means that $\mathbf{u}_{(1)}$ is strictly larger than $\mathbf{u}_{(2)}$. Let the round-up factor $\eta < \frac{1}{10}$, we have $w_{(1)}$ is strictly larger than $w_{(2)}$, and hence we output the correct max.

Then, an index i is reported if and only if the following conditions are satisfied,

(1) i, j is the maximum coordinate in the scaled vector for some $j \in [n^c]$, denoted by $\mathcal{E}_{i,j}$.
(2) i is recovered by CountSketch$_1$, denoted by event $\mathcal{E}^*$.
(3) CountSketch$_2$ does not fail, denoted by event ¬FAIL.

Then, Algorithm 4 reports i with probability

$$\sum_{j\in[n^c]} \mathbf{Pr}\left[\mathcal{E}^*, \neg\text{FAIL} \mid \mathcal{E}_{i,j}\right] \mathbf{Pr}\left[\mathcal{E}_{i,j}\right].$$

By Lemma 3.6, we have

$$\mathbf{Pr}\left[\mathcal{E}^* \mid \mathcal{E}_{i,j}\right] = 1 - \text{poly}(\varepsilon) - \frac{1}{\text{poly}(n)}.$$

By Lemma 3.7, we have

$$\mathbf{Pr}\left[\neg\text{FAIL} \mid \mathcal{E}_{i,j}\right] = \mathbf{Pr}[\neg\text{FAIL}] \pm O\left(\eta \cdot \sqrt{\log n}\right) = q \pm O\left(\eta \cdot \sqrt{\log n}\right).$$

where $q = \Omega(1)$ from Lemma 3.8. Thus, we have

$$\mathbf{Pr}\left[\mathcal{E}^*, \neg\text{FAIL} \mid \mathcal{E}_{i,j}\right] \le \mathbf{Pr}\left[\neg\text{FAIL} \mid \mathcal{E}_{i,j}\right] \ge q + O\left(\eta \cdot \sqrt{\log n}\right).$$

Moreover, by a union bound, we have

$$\begin{aligned}\mathbf{Pr}\left[\neg\mathcal{E}^*, \text{FAIL} \mid \mathcal{E}_{i,j}\right] &\le \mathbf{Pr}\left[\neg\mathcal{E}^* \mid \mathcal{E}_{i,j}\right] + \mathbf{Pr}\left[\text{FAIL} \mid \mathcal{E}_{i,j}\right] \\ &\le \text{poly}(\varepsilon) + \frac{1}{\text{poly}(n)} + 1 - q - O\left(\eta \cdot \sqrt{\log n}\right).\end{aligned}$$

Therefore, we have

$$\mathbf{Pr}\left[\mathcal{E}^*, \neg\text{FAIL} \mid \mathcal{E}_{i,j}\right] \ge q - O\left(\eta \cdot \sqrt{\log n}\right) - \text{poly}(\varepsilon) - \frac{1}{\text{poly}(n)}.$$

Due to our choice of $\eta = \frac{O(\varepsilon)}{\sqrt{\log n}}$, we have

$$\mathbf{Pr}\left[\neg\mathcal{E}^*, \text{FAIL} \mid \mathcal{E}_{i,j}\right] = q \pm \left(O(\varepsilon) - \frac{1}{\text{poly}(n)}\right).$$

Then, by Lemma 1.11, Algorithm 4 reports i with probability

$$\begin{aligned}
&\sum_{j\in[n^c]} \mathbf{Pr}\left[\mathcal{E}^*, \neg\text{FAIL} \mid \mathcal{E}_{i,j}\right] \mathbf{Pr}\left[\mathcal{E}_{i,j}\right] \\
&\quad= \sum_{j\in[n^c]} \frac{|x_i|^p}{\|X\|_p^p}\left(q \pm \left(O(\varepsilon) - \frac{1}{\text{poly}(n)}\right)\right) \\
&\quad= \frac{|x_i|^p}{\|x\|_p^p}\left(q \pm \left(O(\varepsilon) - \frac{1}{\text{poly}(n)}\right)\right).
\end{aligned}$$

Hence, given that the sampler reports some index, the probability of reporting $i \in [n]$ is

$$\frac{|x_i|^p}{\|x\|_p^p}(1\pm\varepsilon) \pm \frac{1}{\text{poly}(n)},$$

which proves the correctness of our approximate sampler. □

Space complexity. We analyze the space complexity for our approximate sampler. First, we bound the size of the set of large indices B recovered by CountSketch$_1$.

Lemma 3.10. *Recall that set B recovers the large indices in CountSketch$_1$. We have $|B| = \text{polylog}\,\frac{1}{\varepsilon}$ with probability $1 - \frac{1}{\text{poly}(n)}$.*

Proof. By Lemma 3.3, there are at most $\text{polylog}\,\frac{1}{\varepsilon}$ indices $k \in [n]$ such that it satisfies, $\mathbf{v}_k > \frac{n^{c/p}\|x\|_p}{400\log\frac{1}{\varepsilon}}$. By Lemma 3.4, the error of CountSketch$_1$ table is at most $\frac{n^{c/p}\|x\|_p}{400\log\frac{1}{\varepsilon}}$. Both events happen with high probability. Since we add an index k to B if the estimate of $\frac{n^{c/p}\|x\|_p}{200\log\frac{1}{\varepsilon}}$, there are at most $\text{polylog}\,\frac{1}{\varepsilon}$ items in B. □

With the bound on $|B|$, we can show the following space complexity.

Lemma 3.11. *Algorithm 4 uses $O\left(n^{1-2/p}\log^2 n\log\frac{1}{\varepsilon}\right)$ bits of space.*

Proof. CountSketch$_1$ table has size $[n^{1-2/p}\log\frac{1}{\varepsilon}] \times [\Theta(\log n)]$. By Lemma 3.10, we only need to maintain $\text{polylog}\,\frac{1}{\varepsilon}$ buckets in each row of CountSketch$_2$. So, we need

$$O\left(n^{1-2/p}\log^2 n\log\frac{1}{\varepsilon} + \log n\,\text{polylog}\left(\frac{1}{\varepsilon}\right)\right),$$

bits of space in total. Suppose that $\text{polylog}\left(\frac{1}{\varepsilon}\right) < n^{1-2/p}\log n$, the second term is dominated by the first term. □

Now, we present the formal statement of our approximate L_p sampler. We defer the discussion on the implementation of the fast update sketch and the derandomization scheme to Section B.

Theorem 3.12. *Given a general turnstile stream x and an accuracy parameter $\varepsilon \in (0,1)$, there is an approximate sampler which outputs an index $i \in [n]$ with probability $\frac{|x_i|^p}{\|x\|_p^p}\cdot(1\pm\varepsilon)$, and outputs FAIL with probability at most 0.1. The algorithm uses $n^{1-2/p}\log^2 n\log\frac{1}{\varepsilon}\cdot\text{poly}(\log\log(n))$ bits of space to run. The update time is $\frac{1}{\varepsilon}\cdot\text{polylog}\left(n,\frac{1}{\varepsilon}\right)$. In addition, it gives a $(1+\varepsilon)$-estimation to the sampled item using extra $\frac{1}{\varepsilon^2}n^{1-2/p}\log^2 n\log\frac{1}{\varepsilon}\cdot\text{poly}(\log\log(n))$ bits of space.*

Proof. We now discuss how to retrieve an accurate approximation of the sampled item. We run a separate CountSketch with $\frac{1}{\varepsilon^2} n^{1-2/p} \log \frac{1}{\varepsilon}$ buckets and $\Theta(\log n)$ rows on the vector $\mathbf{v} \in \mathbb{R}^n$, where $\mathbf{v}_i = \max_{j \in [n^c]} |x_i| \cdot \mathrm{rnd}_\eta(1/\mathbf{e}^{1/p})$. Following the analysis of Lemma 3.4, this gives an estimate of each coordinate in $\mathbf{v}$ with additive error $\varepsilon \cdot \frac{n^{c/p}\|x\|_p}{400 \log \frac{1}{\varepsilon}}$. Then, since the sampled item must satisfy $|\mathbf{v}_i| > \frac{n^{c/p}\|x\|_p}{400 \log \frac{1}{\varepsilon}}$ to pass the threshold of our first-stage table CountSketch_1, the above additive error is smaller than $\varepsilon \cdot |\mathbf{v}_i|$, which implies a $(1+\varepsilon)$-estimation. □

References

[1] Ankit Aggarwal, Amit Deshpande, and Ravi Kannan. 2009. Adaptive Sampling for k-Means Clustering. In *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques, 12th International Workshop, APPROX and 13th International Workshop, RANDOM. Proceedings*. 15–28.

[2] Miklós Ajtai, Vladimir Braverman, T. S. Jayram, Sandeep Silwal, Alec Sun, David P. Woodruff, and Samson Zhou. 2022. The White-Box Adversarial Data Stream Model. In *PODS '22: International Conference on Management of Data*.

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

[4] Alexandr Andoni. 2017. High frequency moments via max-stability. In *IEEE International Conference on Acoustics, Speech and Signal Processing, ICASSP*. 6364–6368.

[5] Alexandr Andoni, Robert Krauthgamer, and Krzysztof Onak. 2011. Streaming Algorithms via Precision Sampling. In *IEEE 52nd Annual Symposium on Foundations of Computer Science, FOCS*. 363–372.

[6] Sepehr Assadi, Amit Chakrabarti, Prantar Ghosh, and Manuel Stoeckl. 2023. Coloring in Graph Streams via Deterministic and Adversarially Robust Algorithms. In *Proceedings of the 42nd ACM SIGMOD-SIGACT-SIGAI Symposium on Principles of Database Systems, PODS*. 141–153.

[7] Idan Attias, Edith Cohen, Moshe Shechner, and Uri Stemmer. 2021. A Framework for Adversarial Streaming via Differential Privacy and Difference Estimators. *CoRR* abs/2107.14527 (2021).

[8] Dmitrii Avdiukhin, Slobodan Mitrovic, Grigory Yaroslavtsev, and Samson Zhou. 2019. Adversarially Robust Submodular Maximization under Knapsack Constraints. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, KDD*. 148–156.

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

[10] Omri Ben-Eliezer, Talya Eden, and Krzysztof Onak. 2022. Adversarially Robust Streaming via Dense-Sparse Trade-offs. In *5th Symposium on Simplicity in Algorithms, SOSA@SODA*, Karl Bringmann and Timothy M. Chan (Eds.). 214–227.

[11] Omri Ben-Eliezer, Rajesh Jayaram, David P Woodruff, and Eylon Yogev. 2020. A Framework for Adversarially Robust Streaming Algorithms. In *Proceedings of the 39th ACM SIGMOD-SIGACT-SIGAI Symposium on Principles of Database Systems*. 63–80.

[12] Omri Ben-Eliezer and Eylon Yogev. 2020. The adversarial robustness of sampling. In *Proceedings of the 39th ACM SIGMOD-SIGACT-SIGAI Symposium on Principles of Database Systems*. 49–62.

[13] Jaroslaw Blasiok, Jian Ding, and Jelani Nelson. 2017. Continuous Monitoring of ℓ_p Norms in Data Streams. In *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques, APPROX/RANDOM*.

[14] Mark Braverman and Or Zamir. 2024. Optimality of Frequency Moment Estimation. *Electron. Colloquium Comput. Complex.* (2024), TR24–175.

[15] Vladimir Braverman, Avinatan Hassidim, Yossi Matias, Mariano Schain, Sandeep Silwal, and Samson Zhou. 2021. Adversarial Robustness of Streaming Algorithms through Importance Sampling. In *Advances in Neural Information Processing Systems 34: Annual Conference on Neural Information Processing Systems, NeurIPS*.

[16] Vladimir Braverman, Jonathan Katzman, Charles Seidell, and Gregory Vorsanger. 2014. An Optimal Algorithm for Large Frequency Moments Using $O(n^{1-2/k})$ Bits. In *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques, APPROX/RANDOM*. 531–544.

[17] Vladimir Braverman and Rafail Ostrovsky. 2013. Approximating Large Frequency Moments with Pick-and-Drop Sampling. In *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques - 16th International Workshop, APPROX, and 17th International Workshop, RANDOM. Proceedings*.

[18] Vladimir Braverman, Rafail Ostrovsky, and Carlo Zaniolo. 2012. Optimal sampling from sliding windows. *J. Comput. Syst. Sci.* 78, 1 (2012), 260–272.

[19] Vladimir Braverman, Emanuele Viola, David P. Woodruff, and Lin F. Yang. 2018. Revisiting Frequency Moment Estimation in Random Order Streams. In *45th International Colloquium on Automata, Languages, and Programming, ICALP*. 25:1–25:14.

[20] Amit Chakrabarti, Prantar Ghosh, and Manuel Stoeckl. 2022. Adversarially Robust Coloring for Graph Streams. In *13th Innovations in Theoretical Computer Science Conference, ITCS*. 37:1–37:23.

[21] Amit Chakrabarti, Subhash Khot, and Xiaodong Sun. 2003. Near-Optimal Lower Bounds on the Multi-Party Communication Complexity of Set Disjointness. In *18th Annual IEEE Conference on Computational Complexity*. 107–117.

[22] Moses Charikar, Kevin C. Chen, and Martin Farach-Colton. 2004. Finding frequent items in data streams. *Theor. Comput. Sci.* 312, 1 (2004), 3–15.

[23] Yeshwanth Cherapanamjeri, Sandeep Silwal, David P. Woodruff, Fred Zhang, Qiuyi Zhang, and Samson Zhou. 2023. Robust Algorithms on Adaptive Inputs from Bounded Adversaries. In *The Eleventh International Conference on Learning Representations, ICLR*.

[24] Edith Cohen, Nick G. Duffield, Haim Kaplan, Carsten Lund, and Mikkel Thorup. 2011. Efficient Stream Sampling for Variance-Optimal Estimation of Subset Sums. *SIAM J. Comput.* 40, 5 (2011), 1402–1431.

[25] Edith Cohen, Nick G. Duffield, Haim Kaplan, Carsten Lund, and Mikkel Thorup. 2014. Algorithms and estimators for summarization of unaggregated data streams. *J. Comput. Syst. Sci.* 80, 7 (2014), 1214–1244.

[26] Edith Cohen and Ofir Geri. 2019. Sampling Sketches for Concave Sublinear Functions of Frequencies. In *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems, NeurIPS*. 1361–1371.

[27] Edith Cohen, Rasmus Pagh, and David P. Woodruff. 2020. WOR and p's: Sketches for ℓ_p-Sampling Without Replacement. In *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems, NeurIPS*.

[28] Graham Cormode and Donatella Firmani. 2014. A unifying framework for L_0-sampling algorithms. *Distributed Parallel Databases* 32, 3 (2014), 315–335.

[29] Graham Cormode, S. Muthukrishnan, and Irina Rozenbaum. 2005. Summarizing and Mining Inverse Distributions on Data Streams via Dynamic Inverse Sampling. In *Proceedings of the 31st International Conference on Very Large Data Bases, Trondheim, Norway, August 30 - September 2, 2005*. ACM, 25–36.

[30] Graham Cormode, S. Muthukrishnan, Ke Yi, and Qin Zhang. 2012. Continuous sampling from distributed streams. *J. ACM* 59, 2 (2012), 10:1–10:25.

[31] Amit Deshpande, Luis Rademacher, Santosh S. Vempala, and Grant Wang. 2006. Matrix Approximation and Projective Clustering via Volume Sampling. *Theory of Computing* 2, 12 (2006), 225–247.

[32] Amit Deshpande and Kasturi R. Varadarajan. 2007. Sampling-based dimension reduction for subspace approximation. In *Proceedings of the 39th Annual ACM Symposium on Theory of Computing*. 641–650.

[33] Hossein Esfandiari, Praneeth Kacham, Vahab Mirrokni, David P. Woodruff, and Peilin Zhong. 2024. Optimal Communication Bounds for Classic Functions in the Coordinator Model and Beyond. In *Proceedings of the 56th Annual ACM Symposium on Theory of Computing, STOC*. 1911–1922.

[34] Cristian Estan and George Varghese. 2003. New directions in traffic measurement and accounting: Focusing on the elephants, ignoring the mice. *ACM Trans. Comput. Syst.* 21, 3 (2003), 270–313.

[35] Alan M. Frieze, Ravi Kannan, and Santosh S. Vempala. 2004. Fast monte-carlo algorithms for finding low-rank approximations. *J. ACM* 51, 6 (2004), 1025–1041.

[36] Sumit Ganguly. 2011. Polynomial Estimators for High Frequency Moments. *CoRR* abs/1104.4552 (2011).

[37] Sumit Ganguly. 2015. Taylor Polynomial Estimator for Estimating Frequency Moments. In *Proceedings of the 42nd International Colloquium on Automata, Languages, and Programming, ICALP*. 542–553.

[38] Sumit Ganguly and David P. Woodruff. 2018. High Probability Frequency Moment Sketches. In *45th International Colloquium on Automata, Languages, and Programming, ICALP*.

[39] Rainer Gemulla, Wolfgang Lehner, and Peter J. Haas. 2008. Maintaining bounded-size sample synopses of evolving datasets. *VLDB J.* 17, 2 (2008), 173–202.

[40] Phillip B. Gibbons and Yossi Matias. 1998. New Sampling-Based Summary Statistics for Improving Approximate Query Answers. In *SIGMOD, Proceedings ACM SIGMOD International Conference on Management of Data*. 331–342.

[41] Anna C Gilbert, Yannis Kotidis, S Muthukrishnan, and Martin Strauss. 2001. Quicksand: Quick summary and analysis of network data. Technical report.

[42] Elena Gribelyuk, Honghao Lin, David P. Woodruff, Huacheng Yu, and Samson Zhou. 2024. A Strong Separation for Adversarially Robust L_0 Estimation for Linear Sketches. In *65th IEEE Annual Symposium on Foundations of Computer Science, FOCS*. 2318–2343.

[43] Elena Gribelyuk, Honghao Lin, David P. Woodruff, Huacheng Yu, and Samson Zhou. 2025. Lifting Linear Sketches: Optimal Bounds and Adversarial Robustness. *CoRR* abs/2503.19629 (2025).

[44] Uffe Haagerup. 1981. The best constants in the Khintchine inequality. *Studia Mathematica* 70, 3 (1981), 231–283.

[45] Peter J. Haas. 2016. Data-Stream Sampling: Basic Techniques and Results. In *Data Stream Management - Processing High-Speed Data Streams*. Springer, 13–44.

[46] Peter J. Haas, Jeffrey F. Naughton, S. Seshadri, and Arun N. Swami. 1996. Selectivity and Cost Estimation for Joins Based on Random Sampling. *J. Comput. Syst. Sci.* 52, 3 (1996), 550–569.

[47] Peter J. Haas and Arun N. Swami. 1992. Sequential Sampling Procedures for Query Size Estimation. In *Proceedings of the ACM SIGMOD International Conference on Management of Data*. 341–350.

[48] Avinatan Hassidim, Haim Kaplan, Yishay Mansour, Yossi Matias, and Uri Stemmer. 2022. Adversarially Robust Streaming Algorithms via Differential Privacy. *J. ACM* 69, 6 (2022), 42:1–42:14.

[49] Ling Huang, XuanLong Nguyen, Minos N. Garofalakis, Joseph M. Hellerstein, Michael I. Jordan, Anthony D. Joseph, and Nina Taft. 2007. Communication-Efficient Online Detection of Network-Wide Anomalies. In *INFOCOM. 26th IEEE International Conference on Computer Communications, Joint Conference of the IEEE Computer and Communications Societies*. 134–142.

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

[51] Piotr Indyk, Sepideh Mahabadi, Shayan Oveis Gharan, and Alireza Rezaei. 2020. Composable Core-sets for Determinant Maximization Problems via Spectral Spanners. In *Proceedings of the 2020 ACM-SIAM Symposium on Discrete Algorithms, SODA*. 1675–1694.

[52] Piotr Indyk, Sepideh Mahabadi, Mohammad Mahdian, and Vahab S. Mirrokni. 2014. Composable core-sets for diversity and coverage maximization. In *Proceedings of the 33rd ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems, PODS'14, Snowbird, UT, USA, June 22-27, 2014*. ACM, 100–108.

[53] Piotr Indyk and David P. Woodruff. 2005. Optimal approximations of the frequency moments of data streams. In *Proceedings of the 37th Annual ACM Symposium on Theory of Computing*. 202–208.

[54] Rajesh Jayaram, Gokarna Sharma, Srikanta Tirthapura, and David P. Woodruff. 2019. Weighted Reservoir Sampling from Distributed Streams. In *Proceedings of the 38th ACM SIGMOD-SIGACT-SIGAI Symposium on Principles of Database Systems, PODS*. 218–235.

[55] Rajesh Jayaram and David P. Woodruff. 2018. Perfect L_p Sampling in a Data Stream. In *59th IEEE Annual Symposium on Foundations of Computer Science, FOCS*. 544–555.

[56] Rajesh Jayaram, David P. Woodruff, and Samson Zhou. 2022. Truly Perfect Samplers for Data Streams and Sliding Windows. In *PODS: International Conference on Management of Data*. 29–40.

[57] Rajesh Jayaram, David P. Woodruff, and Samson Zhou. 2024. Streaming Algorithms with Few State Changes. *Proc. ACM Manag. Data* 2, 2 (2024), 82.

[58] Hossein Jowhari, Mert Saglam, and Gábor Tardos. 2011. Tight bounds for L_p samplers, finding duplicates in streams, and related problems. In *Proceedings of the 30th ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems, PODS*. 49–58.

[59] Daniel M. Kane, Jelani Nelson, Ely Porat, and David P. Woodruff. 2011. Fast moment estimation in data streams in optimal space. In *Proceedings of the 43rd ACM Symposium on Theory of Computing, STOC*. 745–754.

[60] Daniel M. Kane, Jelani Nelson, and David P. Woodruff. 2010. On the Exact Space Complexity of Sketching and Streaming Small Norms. In *Proceedings of the Twenty-First Annual ACM-SIAM Symposium on Discrete Algorithms, SODA*. 1161–1178.

[61] Ping Li. 2008. Estimators and tail bounds for dimension reduction in l_α ($0 < \alpha \leq 2$) using stable random projections. In *Proceedings of the Nineteenth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA*, Shang-Hua Teng (Ed.). 10–19.

[62] Honghao Lin, Hoai-An Nguyen, William Swartworth, and David P. Woodruff. 2024. Unbiased Insights: Optimal Streaming Algorithms in the Forget Model and Beyond. *manuscript* (2024).

[63] Richard J. Lipton and Jeffrey F. Naughton. 1995. Query Size Estimation by Adaptive Sampling. *J. Comput. Syst. Sci.* 51, 1 (1995), 18–25.

[64] Richard J. Lipton, Jeffrey F. Naughton, and Donovan A. Schneider. 1990. Practical Selectivity Estimation through Adaptive Sampling. In *Proceedings of the ACM SIGMOD International Conference on Management of Data*. 1–11.

[65] Sepideh Mahabadi, Piotr Indyk, Shayan Oveis Gharan, and Alireza Rezaei. 2019. Composable Core-sets for Determinant Maximization: A Simple Near-Optimal Algorithm. In *Proceedings of the 36th International Conference on Machine Learning, ICML 2019*, Vol. 97. 4254–4263.

[66] Sepideh Mahabadi, Ilya P. Razenshteyn, David P. Woodruff, and Samson Zhou. 2020. Non-adaptive adaptive sampling on turnstile streams. In *Proccedings of the 52nd Annual ACM SIGACT Symposium on Theory of Computing, STOC*. 1251–1264.

[67] Jianning Mai, Chen-Nee Chuah, Ashwin Sridharan, Tao Ye, and Hui Zang. 2006. Is sampled data sufficient for anomaly detection?. In *Proceedings of the 6th ACM SIGCOMM Internet Measurement Conference, IMC*. 165–176.

[68] Gregory T. Minton and Eric Price. 2014. Improved Concentration Bounds for Count-Sketch. In *Proceedings of the Twenty-Fifth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA*. 669–686.

[69] Morteza Monemizadeh and David P. Woodruff. 2010. 1-Pass Relative-Error L_p-Sampling with Applications. In *Proceedings of the Twenty-First Annual ACM-SIAM Symposium on Discrete Algorithms, SODA*. 1143–1160.
[70] Aduri Pavan, Sourav Chakraborty, N. V. Vinodchandran, and Kuldeep S. Meel. 2024. On the Feasibility of Forgetting in Data Streams. *Proc. ACM Manag. Data* 2, 2 (2024), 102.
[71] Seth Pettie and Dingyu Wang. 2025. Universal Perfect Samplers for Incremental Streams. In *Proceedings of the 2025 Annual ACM-SIAM Symposium on Discrete Algorithms, SODA*. 3409–3422.
[72] Marina Thottan, Guanglei Liu, and Chuanyi Ji. 2010. Anomaly Detection Approaches for Communication Networks. In *Algorithms for Next Generation Networks*. Springer, 239–261.
[73] Srikanta Tirthapura and David P. Woodruff. 2011. Optimal Random Sampling from Distributed Streams Revisited. In *Distributed Computing - 25th International Symposium, DISC. Proceedings*, Vol. 6950. 283–297.
[74] Jeffrey Scott Vitter. 1985. Random Sampling with a Reservoir. *ACM Trans. Math. Softw.* 11, 1 (1985), 37–57.
[75] David P. Woodruff. 2004. Optimal space lower bounds for all frequency moments. In *Proceedings of the Fifteenth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA*. 167–175.
[76] David P. Woodruff, Fred Zhang, and Samson Zhou. 2023. On Robust Streaming for Learning with Experts: Algorithms and Lower Bounds. In *Advances in Neural Information Processing Systems 36: Annual Conference on Neural Information Processing Systems, NeurIPS*.
[77] David P. Woodruff and Peilin Zhong. 2016. Distributed low rank approximation of implicit functions of a matrix. In *32nd IEEE International Conference on Data Engineering, ICDE*. 847–858.
[78] David P. Woodruff and Samson Zhou. 2021. Separations for Estimating Large Frequency Moments on Data Streams. In *48th International Colloquium on Automata, Languages, and Programming, ICALP*. 112:1–112:21.
[79] David P. Woodruff and Samson Zhou. 2021. Tight Bounds for Adversarially Robust Streams and Sliding Windows via Difference Estimators. In *62nd Annual Symposium on Foundations of Computer Science, FOCS*. 1183–1196.
[80] David P. Woodruff and Samson Zhou. 2024. Adversarially Robust Dense-Sparse Tradeoffs via Heavy-Hitters. In *Advances in Neural Information Processing Systems 38: Annual Conference on Neural Information Processing Systems, NeurIPS*.

A Appendix

We state some missing subroutines in the approximate L_p sampler and other complementary results. Section B shows how to implement the fast update sketch in the approximate L_p sampler with derandomization. Section C provides a sketching dimension lower bound for the L_p sampler. In addition, Section D presents an application to estimate the p-norm of a post-processing subset $Q \in [n]$. Finally, Section E gives a rejection sampling framework using the L_0 perfect sampler.

B Fast Update Sketch and Derandomization

In this section, we describe how to implement the discretization in a fast update time. We modify the fast-update sketch in [55] to fit in our CountSketch algorithms with random signs. Our goal is to compute the set of duplicated exponential variables $\{\mathrm{rnd}_\eta(1/\mathbf{e}_1^{1/p}), \ldots, \mathrm{rnd}_\eta(1/\mathbf{e}_{n^c}^{1/p})\}$ for each index $i \in [n]$. Note that the support size of $\mathrm{rnd}_\eta(x)$ for $x \in [\frac{1}{\mathrm{poly}(n)}, \mathrm{poly}(n)]$ is $O\left(\frac{1}{\eta}\log n\right)$, so we can count the number of inverse exponential variables that round up to each value in the support of $\mathrm{rnd}_\eta(x)$.

We define $I_q = (1+\eta)^q$ for $q \in \mathbb{Z} \cap [-Q, Q]$ where $Q = O\left(\frac{1}{\eta}\log n\right)$. Let $\phi(x)$ be the cdf of the $1/p$-th power of the inverse exponential distribution. Then, for a standard exponential variable $\mathbf{e}$, the probability that $\mathrm{rnd}_\eta(1/\mathbf{e}^{1/p})$ equals I_q is $p_q = \phi(I_{q+1}) - \phi(I_q)$. The number D_q of such inverse exponential variables follows a binomial distribution $\mathrm{Bin}(n^c, p_q)$.

Now, upon each arrival, there are D_q updates with value I_q that need to be hashed into CountSketch$_2$. We can generate variables from multinomial distribution to compute $a_{i,j}^q$, which is the number of items in the D_q updates that are hashed to bucket (i, j) in the CountSketch table. Our next goal is to calculate $\sum_{t=1}^{a_{i,j}^q} g_t \cdot I_q$, which is the additive value to the estimator of bucket (i, j). Note that g_t's are Rademacher variables, so the sum follows the distribution $\mathrm{Bin}(a_{i,j}^q, 1/2) - a_{i,j}^q$. Therefore, it suffices to generate one binomial variable for each bucket to compute the sum.

For CountSketch$_1$, we only hash $\Theta(\log n)$ items upon each arrival. To make it consistent with CountSketch$_2$, we find the smallest q such that D_q is not zero, and we use I_q to simulate the maximum of the n^c duplications. Then, for each item hashed to CountSketch$_1$, we generate a geometric variable g_p with parameter $p = \frac{1}{L}$, and we hash it to the bucket located g_p positions after the bucket that receives the previous item.

Last, in our L_2 estimation, we use Gaussian variables to scale our vector instead of random signs. However, we can use a similar way to speed up our calculation. Consider $\sum_{t=1}^{a_{i,j}^q} \phi_t \cdot I_q$ where $\phi_t \sim \mathcal{N}(0,1)$ are the Gaussian variables. Utilizing the 2-stability of Gaussian variables, we have $\sum_{t=1}^{a_{i,j}^q} \phi_t \cdot I_q \sim g\sqrt{a_{i,j}^q} I_q$ where $g \sim \mathcal{N}(0,1)$. Thus, it suffices to generate one Gaussian variable for each bucket to compute the sum.

Next, we state the correctness of our fast-update sketch.

Lemma B.1. *Our fast update sketch results in the same distribution over the CountSketch table and the L_2-estimation scheme as the original algorithm. Upon each arrival in the stream, the update time is $\frac{1}{\varepsilon}\operatorname{polylog}(n, \frac{1}{\varepsilon})$.*

Proof. In the CountSketch$_1$ table, we hash each item to each bucket with probability $\frac{1}{L}$. The geometric variable with parameter $\frac{1}{L}$ characterizes the distribution of the number of buckets between two consecutive buckets that have the hashed item. Thus, our hashing scheme gives the same distribution. For the CountSketch$_2$ table, the multinomial variables give the correct hashing distribution by generating the number of items that are hashed to each bucket. Then, each bucket is increased by $\sum_{t=1}^{a_{i,j}^q} g_t \cdot I_q$ where g_t's are Rademacher variables by our algorithm construction. Since $\sum_{t=1}^{a_{i,j}^q} g_t \cdot I_q$ has the same distribution as $\operatorname{Bin}(a_{i,j}^q, 1/2) - a_{i,j}^q$, our fast update sketch gives precisely the same distribution as the original two-stage CountSketch. The correctness of the fast L_2 estimation follows from Lemma 6 in [55].

Now, we compute the update time. In CountSketch$_1$, we generate $O(\log n)$ geometric random variables upon each stream update. In CountSketch$_2$, we have $\log n \operatorname{polylog}\left(\frac{1}{\varepsilon}\right)$ buckets as specified in Lemma 3.11. For each bucket, it takes $O(\log n)$ time to compute the additive value when an item arrives. Due to our choice of discretization factor $\eta = \frac{\varepsilon}{\operatorname{polylog}\frac{1}{\varepsilon}}$, the update time is $\frac{1}{\varepsilon}\operatorname{polylog}(n, \frac{1}{\varepsilon})$. □

Algorithm derandomization. We now turn to the task of derandomizing our algorithm. Recall that the algorithm relies on two sources of randomness: the hash functions and random signs in CountSketch, and the exponential random variables. Following the approach of [55], we apply two separate pseudorandom generators. The first instance simulates the random variables in CountSketch and the second instance simulates the exponential random variables. Let R_c and R_e respectively denote the random bits required for CountSketch and for the exponential variables. Since there are at most $\operatorname{poly}(n)$ random variables and each have magnitude $\operatorname{poly}(n)$, then at most $\operatorname{poly}(n)$ random bits are required for both R_c and R_e.

For a fixed index $i \in [n]$, we can view the behavior of the algorithm as a Boolean tester $\mathcal{A}_i(R_c, R_e) \in \{0,1\}$, which evaluates to 1 if the L_p sampler selects coordinate i. Our goal is to show that we can replace both R_c and R_e with pseudorandom inputs generated by the PRG without significantly changing the output distribution. Specifically, we show there exist functions F_1 and F_2 such that

$$\left| \Pr_{R_c,R_e} [\mathcal{A}_i(R_c, R_e) = 1] - \mathbf{Pr}[y_1, y_2]\, \mathcal{A}_i(F_1(y_1), F_2(y_2)) = 1 \right| \leq \frac{1}{n^C},$$

for any constant $C > 0$, where y_1 and y_2 are seeds of length $n^{1-2/p}\log^2 n \log\frac{1}{\varepsilon}(\log\log n)^2$ chosen appropriately for the desired level of accuracy.

C L_p Sampler Lower Bound

In this section, we provide a sketching dimension lower bound for the L_p sampler. The analysis is based on Section D.2 in [38], which shows the sketching lower bound of the F_p-estimation protocol. Our core idea is to construct two distributions α and β, and for an arbitrary vector x, we can decide whether x is drawn from α or β with probability at least 0.6 using our L_p sampler. On the other hand, [38] lower bounds the total variation distance between α and β under the image of a "low" dimensional sketch, which implies a sketching lower bound to distinguish whether an arbitrary vector is from α or β. Thus, this translates to a sketching dimension lower bound for the L_p samplers, which can further be used to achieve bit complexity lower bounds using the techniques of [43].

We first introduce the hard distributions.

Definition C.1 (Hard distributions). We define the first distribution α as $\mathcal{N}(0, \mathbf{I}_n)$, which is the n-dimensional standard multi-variate Gaussian distribution. Let e_i be the unit basis vector, where its i-th entry is 1, and the other entries are 0. Let $x \sim \mathcal{N}(0, \mathbf{I}_n)$, let C be a sufficiently large constant, let $E_n = \mathbb{E}_{x\sim\mathcal{N}(0,\mathbf{I}_n)}[\|x\|_p]$, and let i be sampled uniformly from $[n]$. Then, let $z = x + CE_{n-1}\cdot e_i$. We define the second distribution β as the distribution of the vector z.

The next statement lower bounds the sketching dimension that distinguishes the two distributions.

Theorem C.2 (c.f. Section D.2.4 in [38]). *Fix a matrix $S \in \mathbb{R}^{r\times n}$. Let α and β be defined as in Definition C.1. Given an arbitrary x drawn from α or β, suppose that we can decide whether x is from α or β from $S\cdot x$ with probability at least 0.6, then the sketching dimension r is at least $\Omega\left(n^{1-2/p}\log n\right)$.*

With the above result, we lower bound the sketching dimension of L_p samplers by proposing a protocol that solves the problem in Theorem C.2 using L_p samplers.

Theorem C.3. *Let $x \in \mathbb{R}^n$ be a vector. Suppose that there is a linear sketch that outputs an index $i \in [n]$ with probability $\frac{|x_i|^p}{\|x\|_p^p}\cdot(1\pm 0.01)$, and outputs FAIL with probability at most 0.1. Then, its sketching dimension is at least $\Omega\left(n^{1-2/p}\log n\right)$.*

Proof. We claim that a linear sketch $S \in \mathbb{R}^{r\times n}$ that reports an approximate L_p sampler with probability at least 0.9 implies a method to distinguish α and β in Theorem C.2 with probability at least 0.6. Given an arbitrary x drawn from α or β, we take two L_p samples from the vector x using the linear sketch. We state that x is from β if both instances succeed and they sample the same coordinate. Next, we show the correctness of this protocol.

First, we suppose that x is from β, it suffices to sample the large coordinate $x_i = CE_{n-1}$. From the standard properties of the multi-variate Gaussian distribution, we know $E_n = \Theta(n^{1/p})$ and $\mathbb{E}_{x\sim\mathcal{N}(0,\mathbf{I}_n)}[\|x\|_p^p] = \Theta(n)$. Therefore, choosing a sufficiently large constant C in Definition C.1, we have $\frac{|x_i|^p}{\|x\|_p^p} \geq 0.99$ in expectation. Then, i is sampled twice with probability at least 0.9 conditioned on not failing. Thus, our protocol identifies x from β with probability at least 0.9. Next, if x is from α, we misclassify the case if some index is sampled twice. For each coordinate, it is sampled twice with probability $\frac{1}{n^2}$, so by a union bound, the probability of misclassification is at most $\frac{1}{n}$. Thus, for a large enough n, we decide whether x is from α or β with probability at least 0.6.

Then, we have $r = \Omega(n^{1-2/p}\log n)$ from the sketching lower bound in Theorem C.2. By Yao's minimax lemma, we have our desired result. □

D Application to Norm Estimation

To estimate $\|x_Q\|_p^p = \sum_{i\in Q} x_i^p$, we first use our L_p sampler to produce an index $i \in [n]$. We also produce an unbiased estimate $\widehat{F_p}$ to $\|x\|_p^p$ using existing techniques [37]. Now if $i \in Q$, which is revealed on the query Q, then we set the estimate $Y = \widehat{F_p}$, and otherwise, we set $Y = 0$. Observe that $\mathbb{E}[Y] = \|x_Q\|_p^p \pm \frac{1}{\text{poly}(n)}$ and moreover the variance is at most $\|x_Q\|_p^p \cdot \|x\|_p^p$. Thus to drive the variance small enough to obtain a $(1+\varepsilon)$-approximation, it suffices to repeat $O\left(\frac{1}{\alpha\varepsilon^2}\right)$ times, since $\|x_Q\|_p^p \geq \alpha\|x\|_p^p$ for $\alpha \in (0,1)$. We give the full algorithm in Algorithm 5.

Algorithm 5 Moment estimation for a query subset Q

Input: Vector $x \in \mathbb{R}^n$ defined by a turnstile stream, accuracy parameter $\varepsilon \in (0,1)$, post-processing set $Q \subseteq [n]$, ratio parameter $\alpha \in (0,1]$
Output: A $(1+\varepsilon)$-estimation to $\|x\|_p^p$

1: $R \leftarrow O\left(\frac{1}{\alpha\varepsilon^2}\right)$
2: **for** $r \in [R]$ **do**
3: Approximate L_p sample with distortion $\frac{\varepsilon}{4}$ an index $i_r \in [n]$
4: Let C_r be an estimate of $\|S_r\|_p^p$ ▷Ganguly's estimator, see Theorem D.1
5: At the end of the stream, process Q
6: **return** $Z = \frac{1}{R}\sum_{r:i_r\in Q} C_r$

Now, we present the following result from [37] which produces unbiased F_p estimates using optimal space.

Theorem D.1 ([37]). *For $p > 2$ and $0 < \varepsilon \leq 1$, there exists an algorithm in the turnstile stream model that returns an estimator $\widehat{F_p}$ such that $\mathbb{E}\left[\widehat{F_p}\right] = F_p$ and $Var\left[\widehat{F_p}\right] \leq \frac{F_p^2}{50}$ with probability at least 0.9. The algorithm uses $O\left(n^{1-2/p}\log^2 n\right)$ bits of space.*

The next lemma proves that Algorithm 5 gives an $(1+\varepsilon)$-estimation.

Lemma D.2. *For the output Z of Algorithm 5,*

$$\mathbf{Pr}\left[\left|Z - \|x_Q\|_p^p\right| \leq \varepsilon \cdot \|x_Q\|_p^p\right] \geq 0.99.$$

Proof. For each $r \in [R]$, we define $Z_r = C_r$ if $i_r \in Q$ and $Z_r = 0$ if $i_r \notin Q$, so that $Z = \frac{1}{R}\sum_{r\in[R]} Z_r$. We have

$$\mathbb{E}[Z_r] = \sum_{i\in Q}\left((1\pm\varepsilon)\frac{|x_i|^p}{\|x\|_p^p} + \frac{1}{\text{poly}(n)}\right)\cdot\mathbb{E}[C_r].$$

By Theorem D.1, $\mathbb{E}[C_r] = \|x\|_p^p$, so that

$$\mathbb{E}[Z_r] = \left(1\pm\frac{\varepsilon}{4}\right)\sum_{i\in Q}|x_i|^p = \left(1\pm\frac{\varepsilon}{4}\right)\|x_Q\|_p^p.$$

Moreover, we have

$$\mathbb{E}\left[Z_r^2\right] = \sum_{i\in Q}\left(\left(1\pm\frac{\varepsilon}{4}\right)\frac{|x_i|^p}{\|x\|_p^p} + \frac{1}{\text{poly}(n)}\right)\cdot\mathbb{E}\left[C_r^2\right].$$

By Theorem D.1, we have $\mathbb{E}\left[C_r^2\right] = \frac{\|x\|_p^p}{50}$, so that

$$\mathbb{E}\left[Z_r\right] \leq \frac{1}{25}\sum_{i\in Q}|x_i|^p \cdot \|x\|_p^p = \frac{1}{25}\|x_Q\|_p^p \cdot \|x\|_p^p.$$

Therefore, we have $\mathbb{E}\left[Z\right] = \left(1 \pm \frac{\varepsilon}{4}\right)\|x_Q\|_p^p$ and

$$\mathrm{Var}\left[Z\right] \leq \frac{1}{R^2}\sum_{r\in[R]}\frac{1}{25}\|x_Q\|_p^p \cdot \|x\|_p^p = \frac{1}{25R}\|x_Q\|_p^p \cdot \|x\|_p^p.$$

Since $R = O\left(\frac{1}{\alpha\varepsilon^2}\right)$ and $\|x_Q\|_p^p \geq \alpha\|x\|_p^p$ by assumption, then we have

$$\mathrm{Var}\left[Z\right] \leq O\left(\varepsilon^2\right) \cdot \|x_Q\|_p^{2p}.$$

By Chebyshev's inequality, it follows that

$$\mathbf{Pr}\left[\left|Z - \|x_Q\|_p^p\right| \leq \varepsilon \cdot \|x_Q\|_p^p\right] \geq 0.99.$$

□

Now, we state the formal theorem of norm estimation for a query subset.

Theorem D.3. *Given $p > 2$, there exists an algorithm that processes a turnstile stream defining a vector $x \in \mathbb{R}^n$ and a post-processing query set $Q \subseteq [n]$, and with probability at least 0.99, outputs a $(1+\varepsilon)$-approximation to $\|x_Q\|_p^p$. For $\|x_Q\|_p^p \geq \alpha\|x\|_p^p$, the algorithm uses $\tilde{O}\left(\frac{1}{\alpha\varepsilon^2}n^{1-2/p}\right)$ bits of space.*

Proof. Consider Algorithm 5. The justification of correctness of approximation results from Lemma D.2. Thus it remains to analyze the space complexity of Algorithm 5. We can use either a perfect L_p sampler or an approximate L_p sampler to acquire the samples i_r. By Theorem 2.9 or Theorem 3.12, these subroutines use $\tilde{O}\left(n^{1-2/p}\right)$ bits of space. By Theorem D.1, the F_p estimation algorithm for producing C_r also uses $\tilde{O}\left(n^{1-2/p}\right)$ bits of space. Since each subroutine is repeated $R = O\left(\frac{1}{\alpha\varepsilon^2}\right)$ times, then the total space complexity is $\tilde{O}\left(\frac{1}{\alpha\varepsilon^2}n^{1-2/p}\right)$ bits of space, as claimed. □

E Rejection Sampling Framework

In this section, we generalize the rejection sampling method to provide a framework that gives perfect G-samplers for a general function $G(z) \geq 0$. For both samplers, we require the following perfect L_0-sampler, which outputs a coordinate $i \in [n]$ with probability $\frac{1}{\|x\|_0} + \frac{1}{\mathrm{poly}(n)}$, along with the exact value of x_i.

Theorem E.1. *[58] There exists a perfect L_0 sampler on turnstile streams that succeeds with probability at least $1-\delta$, using $O\left(\log^2 n \log\frac{1}{\delta}\right)$ bits of space. Moreover, for the index $i \in [n]$ that it returns, it returns x_i exactly.*

Note that if we use Theorem E.1 to sample an index $i \in [n]$ with probability $\frac{1}{\|x\|_0} + \frac{1}{\mathrm{poly}(n)}$ and obtain x_i exactly, then we can output i with probability $\frac{G(x_i)}{H}$, where H is some fixed quantity that upper bounds $G(x_i)$ for all $i \in [n]$, to guarantee that the probability is well-defined. Suppose that we have access to an upper bound H to $\max_{z\in[m]} G(z)$ and a lower bound Q to $\min_{z\in[m]} G(z)$. First, we use Theorem E.1 to acquire L_0 samples and obtain the sampled item x_i exactly. Next, we output i with probability $\frac{G(x_i)}{H}$. The rejection probability is well-defined since $G(z) \leq H$ for all z, in addition, we sample from the correct distribution due to the choice of the rejection probability. Note that a L_0 sample is accepted with probability at least $\frac{Q}{H}$, so it suffices to use $O\left(\frac{H}{Q}\right)$ independent L_0 samples.

Algorithm 6 Perfect G-sampler for $G(z)$

Input: Vector $x \in \mathbb{R}^n$ defined by a turnstile stream, function $G(z)$, parameters $H \geq \max_{z\in[m]} G(z)$, $Q \leq \min_{z\in[m]} G(z)$

Output: G-sample from x for $G(z)$

1: Let m be the length of the stream, $R \leftarrow O\left(\frac{H}{Q}\right)$
2: **for** $r \in [R]$ **do**
3: Acquire a perfect L_0 sample i_r with failure probability 0.01
4: **return** i_r and abort, with probability $\frac{G(i_r)}{H}$

Now, we give our formal theorem statement for the rejection sampling framework.

Theorem E.2. *For the function $G(z)$ satisfying $Q \leq G(z) \leq H$ for all $z \in [m]$, there exists a G-sampler on turnstile streams that uses $O\left(\frac{H}{Q}\log^2 n\right)$ bits of space and succeeds with probability* 0.99.

Proof. Consider Algorithm 6. Let $N = \|x\|_0$ be the number of nonzero coordinates and let $\mathcal{S}$ be the set of nonzero coordinates, so that $N = |\mathcal{S}|$. Then for each $i \in \mathcal{S}$ and each fixed $r \in [R]$, the algorithm samples i with probability $\frac{1}{N} + \frac{1}{\text{poly}(n)}$ and then chooses to accept i with probability $\frac{G(x_i)}{H}$. Therefore, the probability the sample returns i is $\frac{G(x_i)}{NH} + \frac{1}{\text{poly}(n)}$. Hence, conditioned on some index being returned, the probability i is sampled is

$$\frac{G(x_i)}{\sum_{j\in[n]} G(x_j)} + \frac{1}{\text{poly}(n)},$$

as desired.

On the other hand, for each $r \in [R]$, the sample i_r is returned with probability $\frac{G(x_{i_r})}{H} \geq \frac{Q}{H}$. Hence, the algorithm outputs a sample with probability at least 0.99 for $R = O\left(\frac{H}{Q}\right)$. By Theorem E.1, each L_0 sampler with failure probability 0.01 uses space $O\left(\log^2 n\right)$. Therefore, Algorithm 6 uses $O\left(\frac{H}{Q} \cdot \log^2 n\right)$ bits of space in total. □

Application to the logarithmic function and the cap function. The rejection sampling framework can be utilized to give perfect G-samplers for the logarithmic function $G(z) = \log(1 + |z|)$ and the cap function $G(z) = \min(T, |z|^p)$.

Recall that we use Theorem E.1 to sample an index $i \in [n]$ with probability $\frac{1}{\|x\|_0} + \frac{1}{\text{poly}(n)}$ and obtain x_i exactly, then we output i with probability $\frac{G(x_i)}{H}$, where H upper bounds $G(x)$. For the logarithmic function $G(z) = \log(1 + |z|)$, we can choose $H = \log(1 + m)$ and for the cap function $G(z) = \min(T, |z|^p)$, we choose $H = T$. Thus it remains to acquire a sufficiently large number of independent L_0 samples to accept some sample with probability. Since a sample is accepted with probability at least $\frac{\log 2}{\log(m+1)}$ for the logarithmic function, we can use $O(\log m)$ independent L_0 samples, which use $O\left(\log^3 n\right)$ bits of space assuming $m = \text{poly}(n)$. Similarly, a sample is accepted with probability at least $\frac{1}{T}$ for the cap function, so it suffices to use $O(T)$ independent L_0 samples, which use $O\left(T \log^2 n\right)$ bits of space.

Received December 2024; revised February 2025; accepted March 2025