An Optimal Algorithm for Large Frequency Moments Using $O(n^{1-2/k})$ Bits*†

Vladimir Braverman[1], Jonathan Katzman[2], Charles Seidell[3], and Gregory Vorsanger[4]

1 **Johns Hopkins University, Department of Computer Science**
vova@cs.jhu.edu
2 **Johns Hopkins University**
jkatzma2@jhu.edu
3 **Johns Hopkins University**
cseidel5@jhu.edu
4 **Johns Hopkins University, Department of Computer Science**
gregvorsanger@jhu.edu

Abstract

In this paper, we provide the first optimal algorithm for the remaining open question from the seminal paper of Alon, Matias, and Szegedy: approximating large frequency moments. Given a stream $D = \{p_1, p_2, \ldots, p_m\}$ of numbers from $\{1, \ldots, n\}$, a frequency of i is defined as $f_i = |\{j : p_j = i\}|$. The k-th *frequency moment* of D is defined as $F_k = \sum_{i=1}^n f_i^k$.

We give an upper bound on the space required to find a k-th frequency moment of $O(n^{1-2/k})$ bits that matches, up to a constant factor, the lower bound of [48] for constant ϵ and constant k. Our algorithm makes a single pass over the stream and works for any constant[1] $k > 3$. It is based upon two major technical accomplishments: first, we provide an optimal algorithm for finding the heavy elements in a stream; and second, we provide a technique using Martingale Sketches which gives an optimal reduction of the large frequency moment problem to the all heavy elements problem. Additionally, this reduction works for any function g of the form $\sum_{i=1}^n g(f_i)$ that requires sub-linear polynomial space, and it works in the more general turnstile model. As a result, we also provide a polylogarithmic improvement for frequency moments, frequency based functions, spatial data streams, and measuring independence of data sets.

1998 ACM Subject Classification F.2 Analysis of Algorithms And Problem Complexity

Keywords and phrases Streaming Algorithms, Randomized Algorithms, Frequency Moments, Heavy Hitters

Digital Object Identifier 10.4230/LIPIcs.APPROX-RANDOM.2014.531

1 Introduction

The analysis of massive data sets has become an exciting topic of theoretical algorithms research. As these datasets grow increasingly large, the need to develop new algorithms which can run using sublinear memory has become paramount. It is often convenient to view such datasets as *data streams*. In this paper we consider the following streaming model:

* This work was supported in part by DARPA grant N660001-1-2-4014. Its contents are solely the responsibility of the authors and do not represent the official view of DARPA or the Department of Defense.

† This work was supported in part by Pistritto Fellowship.

[1] We stress that our bound only holds for $k = O(1)$. In fact, the dependence on k is at least exponential. See Section 1.1.

17th Int'l Workshop on Approximation Algorithms for Combinatorial Optimization Problems (APPROX'14) /
18th Int'l Workshop on Randomization and Computation (RANDOM'14).
Editors: Klaus Jansen, José Rolim, Nikhil Devanur, and Cristopher Moore; pp. 531–544
Leibniz International Proceedings in Informatics
LIPICS Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

▶ **Definition 1.** Let m and n be positive integers. A **stream** $D = D(n, m)$ is a sequence of integers $p_1, \ldots, p_m$, where $p_i \in \{1, \ldots, n\}$. A **frequency vector** is a vector of dimensionality n with non-negative entries $f_i, i \in [n]$ defined as:

$$f_i = |\{j : 1 \leq j \leq m, p_j = i\}|$$

A k-th **frequency moment** of a stream D is defined by $F_k(D) = \sum_{i \in [n]} f_i^k$. Also, $F_\infty = \max_{i \in [n]} f_i$ and $F_0 = |\{i : f_i > 0\}|$.

In their celebrated paper, Alon, Matias, and Szegedy [1, 1] introduced the following problem:

▶ **Problem 2.** *What is the space complexity of computing a $(1 \pm \epsilon)$-approximation of F_k in one pass over D?*

In this paper we consider the case where $k > 2$. Many algorithms have been designed to solve this particular problem, and we now provide a brief overview of the upper and lower bounds provided. To begin, [1] gave a lower bound of $\Omega(n^{1-5/k})$ (for $k \geq 6$) and an upper bound of $O(\frac{1}{\epsilon^2} n^{1-1/k} \log(nm))$. Bar-Yossef, Jayram, Kumar, and Sivakumar [4] improved the lower bound and showed a bound of $\Omega(n^{1-(2+\lambda)/k})$ for their one pass algorithm where λ is a small constant. They also showed a lower bound of $\Omega(n^{1-3/k})$ for a constant number of passes. Chakrabarti, Khot, and Sun [19] showed a lower bound of $\Omega(n^{1-2/k})$ for one pass and $\Omega(n^{1-2/k}/(\log n))$ for a constant number of passes. Gronemeier [31] and Jayram [36] extended the bound of [19] from one pass to multiple passes. Woodruff and Zhang [48] gave a lower bound of $\Omega(n^{1-2/k}/(\epsilon^{(4/p)} t))$ for a t-pass algorithm. Ganguly [28] improved the result of [48] for small values of ϵ and for $t = 1$. Price and Woodruff [44] gave a lower bound on the number of linear measurements and Andoni, Nguyen, Polyanskiy, and Wu [3] showed that $\Omega(n^{1-2/k} \log n)$ linear measurements are necessary.

In terms of upper bounds, Ganguly [26] and Coppersmith and Kumar [20] simultaneously gave algorithms with space complexity[2] $\tilde{O}(n^{1-1/(k-1)})$. In their breakthrough paper, Indyk and Woodruff [33] gave the first upper bound that is optimal up to a polylogarithmic factor. Their bound was improved by a polylogarithmic factor by Bhuvanagiri, Ganguly, Kesh, and Saha [7]. Monemizadeh and Woodruff [41] gave a bound of $O(\epsilon^{-2} k^2 n^{1-2/k} \log^5(n))$ for a $\log(n)$-pass algorithm. For constant ϵ, Braverman and Ostrovsky [13] gave a bound of $O(n^{1-2/k} \log^2(n) \log^{(c)}(n))$ where $\log^{(c)}(n)$ is the iterated logarithm function. Andoni, Krauthgamer, and Onak [2] gave a bound of $O(k^2 \epsilon^{-2-6/p} n^{1-2/k} \log^2(n))$. Ganguly [27] gave a bound of
$O(k^2 \epsilon^{-2} n^{1-2/k} E(k, n) \log(n) \log(nmM) / \min(\log(n), \epsilon^{4/k-2}))$ where $E(k, n) = (1-2/k)^{-1}(1-n^{-4(1-2/k)})$. Braverman and Ostrovsky [16, 15] gave a bound of $O(n^{1-2/k} \log(n) \log^{(c)}(n))$.

1.1 Main Result

For constant ϵ and k we provide a streaming algorithm with space complexity $O(n^{1-2/k})$. Thus, our upper bound matches the lower bound of Woodruff and Zhang [48] up to a constant factor. Our algorithm makes a single pass over the stream and works for constant $k > 3$.

The main technical contribution is a new algorithm that finds heavy elements in a stream of numbers. Then, combining this result with the Martingale Sketches technique we create an algorithm to approximate F_k. In particular, we show:

[2] The standard notation $\tilde{O}$ hides factors that are polylogarithmic in terms of n, m and polynomial in terms of the error parameter ϵ.

▶ **Theorem 3.** *Let ϵ be a constant and $k \geq 7$. There exists an algorithm that outputs a $(1 \pm \epsilon)$-approximation of F_k, makes three passes over the stream, uses $O(n^{1-2/k})$ memory bits, and errs with probability at most $1/3$.*

We now present the necessary definitions and theorems.

▶ **Definition 4.** Let D be a stream and ρ be a parameter. The index $i \in [n]$ is a *ρ-heavy element* if $f_i^k \geq \rho F_k$.

▶ **Definition 5.** A randomized streaming algorithm $\mathcal{A}$ is an *Algorithm for Heavy Elements (AHE)* with parameters ρ and δ if the following is true: $\mathcal{A}$ makes three passes over stream D and outputs a sequence of indices and their frequencies such that if element i is a ρ-heavy element for F_k then i will be one of the indices returned[3]. $\mathcal{A}$ errs with probability at most δ.

▶ **Theorem 6.** *Let $k \geq 7$. There exists an absolute constant $C \leq 10$ and an AHE algorithm with parameters ρ and δ that uses*

$$O(\frac{1}{\rho^C}(F_0(D))^{1-2/k} \log \frac{1}{\delta}) \tag{1}$$

bits.

▶ **Theorem 7.** *Given Theorem 6, for any ϵ there exists an algorithm that uses*

$$O(\frac{1}{\epsilon^{2C}}(F_0(D))^{1-2/k}) \tag{2}$$

memory bits, makes three passes over D, and outputs a $(1 \pm \epsilon)$-approximation of F_k with probability at least $2/3$. Here C is the constant from Theorem 6.

From here, we see that the main theorem, Theorem 3, follows directly from Theorem 7.

After establishing the matching bound with three passes, we improve our algorithm further:

▶ **Theorem 8.** *Let ϵ be a constant and $k > 3$. Assuming that m and n are polynomially far, there exists an algorithm that outputs a $(1 \pm \epsilon)$-approximation of F_k, makes one pass over the stream, uses $O(n^{1-2/k})$ memory bits, and errs with probability at most $1/3$.*

We stress that our bound $O(n^{1-2/k})$ only holds for $k = O(1)$. In fact, the dependence on k is at least exponential. This is because we (initially) assume that for the heavy element f_1 it is true that $f_1^k \geq C \sum_{i>1} f_i^k$ for some large constant $C \geq 2^\Psi$ where $\Psi = \Omega(k)$. Later, we use a standard hashing technique to find heavy elements $f_1^k > \rho F_k$ for smaller values of ρ.

Additional Results

The previous theorems demonstrate the optimal reduction from the problem of computing frequency moments for constant $k > 2$ to the problem of finding heavy elements with constant error. The Martingale Sketches technique is an improvement over the previous method of recursive sketches [16]. Thus, our method is applicable in a general setting of approximating L_1-norms of vectors which have entries obtained by applying entry-wise functions on the frequency vector. As a result, we answer the main open question from [16] and improve several applications in [16].

[3] Indices of non-heavy elements can be reported as well.

Additionally, this reduction works for any function g of the form $\sum_{i=1}^{n} g(f_i)$ that requires sub-linear polynomial space, and it works in the more general turnstile model. As a result, we also provide a polylogarithmic improvement for frequency moments, frequency based functions, spatial data streams, and measuring independence of data sets. We will provide a detailed list of these results in the full version of the paper.

1.2 Related Work

Approximating F_k has become one of the most inspiring problems in streaming algorithms. To begin, we provide an incomplete list of papers on frequency moments [32, 25, 1, 14, 8, 11, 4, 12, 19, 5, 33, 20, 22, 24, 26, 29, 39, 13, 37, 38, 43, 46, 6, 18, 34, 27, 28, 48, 35] and references therein. These and other papers have produced many beautiful results, important applications, and new methods. Below we will mention a few of the results that provide relevant bounds. We refer a reader to [42, 47] and references therein for further details.

In [1], the authors observed that it is possible to approximate F_2 in optimal polylogarithmic space. Kane, Nelson and Woodruff [38] gave a space-optimal solution for F_0. Kane, Nelson, and Woodruff [37] gave optimal-space results for $F_k, 0 < k < 2$. In addition to the original model of [1], a variety of different models of streams have been introduced. These models include the turnstile model (that allows insertion and deletion) [32], the sliding window model [10, 23, 17], and the distributed model [30, 48, 21]. In the turnstile model, where the updates can be integers in the range $[-M, M]$, the latest bound by Ganguly [27] is

$$O(k^2\epsilon^{-2}n^{1-2/k}E(k,n)\log(n)\log(nmM)/\min(\log(n),\epsilon^{4/k-2}))$$

where $E(k,n) = (1-2/k)^{-1}(1-n^{-4(1-2/k)})$. Recently, Li and Woodruff provided a matching lower bound for $\epsilon < 1/(\log n)^{O(1)}$ [40]. The bound from [27] is roughly $O(n^{1-2/k}\log^2(n))$ for constant ϵ, k and it matches the earlier result of Andoni, Krauthgamer, and Onak [2]. For the turnstile model, the problem has been solved optimally for $\epsilon < 1/(\log n)^{O(1)}$ [27, 40]. These results combined with our result demonstrate that the turnstile model is fundamentally different from the model of Alon, Matias, and Szegedy.

2 Intuition

In this abstract we will provide a detailed but high-level explanation of the algorithms and techniques that we employ in the proof of our main theorem. All of this description, as well as the analysis required for a rigorous proof of this theorem, can be found in more detail in the full version [9].

2.1 High Level Description of the Algorithm

We present a composite algorithm to estimate frequency moments. At the absolute lowest degree of detail, we perform three steps. First, we determine the length of the stream. Second, we use a new algorithm to efficiently find heavy elements. Finally, we use a new technique to estimate the value of frequency moments from the weight of the found heavy elements. We now describe the intuition of each of these parts in detail.

2.2 The Heavy Hitter Algorithm

The key step in our algorithm for frequency moment computation is a new technique to compute the heavy hitters of a stream. In order to determine which elements are ρ-heavy in stream D, we present an algorithm that is implemented as a sequence of sub-algorithms, and in general we will refer to each of these sub-algorithms as a "game". In [15] it is shown that $O(n^{1-2/k})$ samples are sufficient to solve the problem. However, each sample requires $\log n$ bits for counting the frequency (counters) and for identifying the elements (IDs). The resulting bound is $O(n^{1-2/k} \log n)$ bits. The goal of our algorithm, therefore, is to reduce the space required for counters and IDs from $\log n$ to an amortized $O(1)$ bits, achieving the optimal bound.

First we will describe the workings of a single instance of the game, and then we will describe the sequence of games that composes our heavy hitter algorithm. Each game in the sequence will be run in parallel, and the cost of the sequence of games will form a geometric series, which when evaluated will yield a total cost of $O(n^{1-2/k})$ bits. The crucial observation is that a heavy element in the stream will be returned by at least one of these games with constant probability, and will be sufficiently frequent to stand out from the other returned values as the true heavy hitter.

2.3 The Initial Algorithm

We will begin our solution by designing an algorithm for finding heavy elements in a stream which conforms to several assumptions. The key step of the algorithm is a subroutine that we call a *game*, which is described next. The algorithm will execute (in parallel) a sequence of several games with different parameters α and β.

The winner of each (α, β)-game will compete against the others, and the overall winner will be the output of the algorithm.

2.4 The Game

To find a heavy element of a stream and prove Theorem 6, we play a game using the stream as input. First we split the stream into equally sized rows as we read it in, and assemble them into a matrix M.

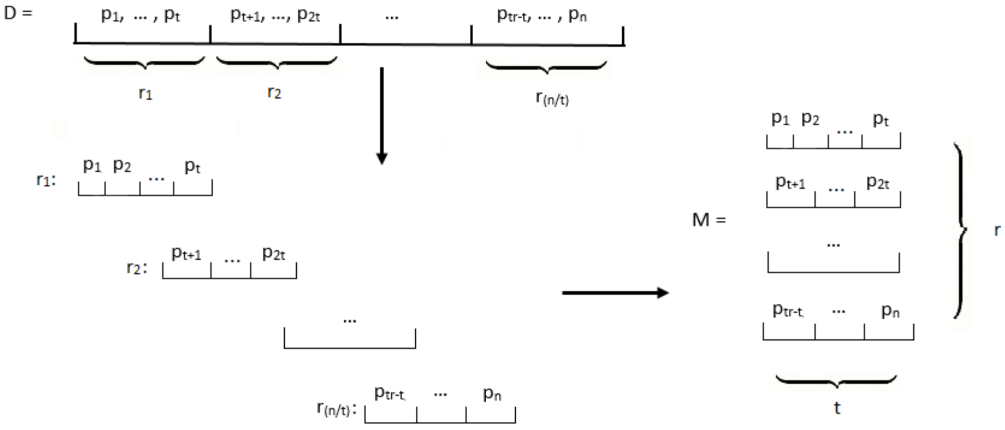

Figure 1 Transforming the stream into a matrix.

A single game is described colloquially as follows: for each row, we create a "team" that is composed of a group of $w = O(n^{1-2/k})$ players each competing to be the winner of that game. To create these teams, we sample elements from the current row to act as the players on each team, and give each player an ID number equal to one of the sampled elements. For each player on a team, maintain a counter to track how often their ID number appears as we move through the stream. If the player's counter does not grow fast enough, that player is removed from the game.

The γ-th round is played by each team after 2^γ rows have passed since the team started playing. In each round, we divide the players of each team into groups of size 3^γ, the players compete within these groups, and there is at most one winner per group, i.e. the surviving player whose counter is highest. The winning player from each group continues to play throughout the remainder of the game, competing in further rounds. Players who are not winners withdraw from the game and do not compete in any further rounds.

At the end of the game, each team will have at most one winner. The winners from every team then compete against each other, and the player with the highest overall counter is the overall winner. Below is the general pseudocode for the initial "game" on a given stream $D = \{d_1, d_2, \ldots d_m\}$ in Algorithm 1:

Algorithm 1 The Game

```
1. Break the stream into rows, as described above.
2. For the $i$-th row, as it is read in:
   a. Sample elements from the $i$-th row to act as team $T_i$ for that row.
   b. For each player $t_i$ on team $T_i$:
      i. Get the initial count for that player by counting the rest of the occurrences of that
         element in the $i$-th row.
      ii. For each other team $T_x$ with $x < i$, update the count of each player $t_x$ on team $T_x$
          such that $t_x$ is equal to the current element.
   c. Increment each other row's rounds-since-formation.
   d. For each other row, if $2^\gamma$ rows have passed since this row's formation:
      i. Eliminate all players whose counters are less than $2^\gamma$
      ii. Divide that row into groups of $3^\gamma$ players.
      iii. Compete among the players in each group.
      iv. Eliminate all players from the row that did not win their groups.
3. At the end of the game compete among all remaining elements to determine the winner.
```

To illustrate the analysis, assume that $F_k = O(n)$ and that 1 is a heavy element that appears among every $O(n^{1-1/k})$ elements. We can make two observations. First, the counter of the player who samples 1 requires only $O(\gamma)$ bits after seeing $2^\gamma n^{1-1/k}$ elements of the stream. Also, this counter will have a nice property of linear growth: after seeing 2^γ intervals the counter will be at least 2^γ.

Second, we can observe that the sum of the frequencies of every element that is not 1 has frequency larger than λ is at most $\frac{G_k}{\lambda^{k-1}}$, where $G_k = F_k - f_1^k$.

Thus, we can bound the number of intervals with many such elements. For example, fix some constant C and let an element l be "γ-bad" if $f_l \geq 2^\gamma$ and consider an interval to be a "γ-bad" interval if it contains more than $\frac{n^{1-1/k}}{2^{C\gamma}}$ distinct bad elements. There are at most $\frac{G_k}{2^{(k-1)\gamma}} \frac{2^{C\gamma}}{n^{1-1/k}}$ such intervals.

Under the assumption that $F_k = O(n)$, and for sufficiently large k, this number is bounded by $\frac{n^{1/k}}{2^{C\gamma}}$. We conclude that the majority of the intervals (e.g 0.95%) will not be γ-bad for any γ. Let an interval that is not *bad* be called a *good* interval. Fix one such good interval nd assume, w.l.o.g, that the first player samples the heavy element, 1. In the γ-th round there are at most $3^\gamma - 1$ players that can compete with the first player. Then, because the interval is *good* the probability to sample any element with frequency $\geq 2^\gamma$ is at most $\frac{3^\gamma}{2^{C\gamma}}$. Summing over all γ, we conclude that, with a constant probability, the heavy hitter will not compete with other players since they will expire at some earlier point in the game.

Unfortunately, the above observations are not true in general. First, the distribution of the heavy element throughout the stream can be arbitrary. For example, half of the appearances of the heavy element may occur in a single row and thus we need $\log n$ bits at the time each player starts playing the game. Second, it is possible that G_k is much larger than n in which case the number of bad intervals can be larger. It is possible that there exist intervals with the number of 1s being 2^i for every $i = 0, 1, \ldots, \lfloor \log(n) \rfloor$ and they comprise an equal percentage of the total frequency.

To overcome these problems we show that there exists a β such that there are a sufficiently large number of intervals where the number of 1s in each interval is in the range $[2^\beta, 2^{\beta+1}]$.

In general, our goal is to show that for any distribution of the heavy element in the stream there exists some β such that

1. $O\left(\frac{n^{1-2/k}}{2^{\mu\beta}}\right)$ samples from each interval are needed to sample the heavy element with a constant probability, where μ is a small constant,
2. Players that may compete with the heavy element will expire with high probability before the competition.

The space bound implies that the problem can be solved without knowledge of the value of β. To address the case when $F_k > n$ we play an additional set of games to search for a parameter α. This parameter α is used to define the number of columns in the matrix we play the game on, specifically $2^{-\alpha} n^{1-1/k}$.

2.5 A Sequence of Games

An exhaustive search of the range of α, β, which we show in the full version of the paper, [9], is at most logarithmic, and yields the sequence of games that eventually constitutes our heavy hitter algorithm. The total cost of playing all of the games is still $O(n^{1-2/k})$. As stated before, we will prove that the cost for these games is geometric and, after some slight modifications discussed in this paper, yields the desired overall cost.

Our proof of correctness will rely heavily on what we term the "Noisy Lemma" (see full version of paper, [9]). In this lemma we aim to show that at least one event in a collection of "good" events will come to pass with at least a certain probability, even if each event is impeded by a number of "noisy" events that can prevent the event from occurring. This lemma can then be applied to show that at least one player corresponding to the actual heavy hitter will win overall, even if there is a chance that other players with large but not heavy elements will win some games.

Having established this algorithm, we show we can eliminate the *log* factor associated with storing counters. This is because we store $O(\gamma^{O(1)})$ bits per counter. It remains to show that we can store the ID of each player in sufficiently small space to achieve our desired bound. In order to do this, we will transfer the duty of tracking the identity of each player from a deterministic ID to a hashed signature.

2.6 Signatures instead of IDs

Given a new element of the stream, our algorithm needs to be able to differentiate elements for the following reasons:

- If the new element has the same ID as one of the samples, then the stored counter of the sample should be incremented.

- If the new element has been chosen as a new sample for one of the players, it is necessary to compare the IDs of the new elements and the current sample. If they are the same, we increment the counter; if they are different, we have to replace the sample.

Since there are n possible elements, $\log n$ bits are required to identify all of the IDs deterministically. Note that after $O(\log\log n)$ rounds a team with initially w active players will only have $\frac{w}{\log^{\Omega(1)} n}$ active players. Thus, $O(n^{1-2/k})$ bits are sufficient to store all IDs of sampled elements in all tables for all old rows for which at least $O(\log\log n)$ rounds have passed.

Therefore, we only need to take care of the first $\log\log n$ rounds each row plays. Our goal is to reach $O(\gamma^{O(1)})$ bits per signature. To achieve this goal, we use random signatures. Unfortunately, if we simply hash $[n]$ into a range of $[2^{\gamma^{O(1)}}]$, the number of collisions per row will still be polynomial in n for small γ's.

In contrast, a small (constant) probability of collision can be shown for sets of small cardinalities. Thus, to use signatures we have to reduce the cardinality of the set. We do so by implementing a sampling procedure with an additional independent hash function. We choose the function carefully so that in a game with parameter β the probability to sample the heavy element for a row is preserved. First, we hash elements into a range $g : [n] \mapsto [t]$, where t is the number of columns in our matrix, and allow only those elements with values smaller than 2^β to be sampled. We then compute signatures only for the elements in the "pool" Γ of all elements that pass the g filter. With constant probability, $|\Gamma| = O(2^\beta)$ and no element will have the same signature as the heavy element (See the next section for more detail).

The same argument will work for any 2^γ rows if the length of the signature is $\Omega(\gamma)$. Thus, we can use γ bits to represent all of the IDs. Then, after $\log\log n$ rounds, the cardinality will be small enough such that we are able to switch our method and use the real ID of a given player's element. We implement this as follows: after $\log\log n$ rounds, we take the full ID of the first element that can be sampled and has a matching signature to the player, and assign it as the new ID of the player. We then count the frequency of elements based on this new ID. With constant probability this will be the same heavy element that we used to generate the signature to begin with.

While this technique reduces the space required, the downside is that there will be collisions for many of the w players and as a result we need to overcome two technical issues. First, due to multiple IDs being hashed to the same signature, the counters of the players can be larger than the frequency of the sampled element they are supposed to be counting. Second, if the heavy element is sampled from row i it can now be incorrectly compared with many non-heavy elements from rows $\{i, \dots, i + 2^\gamma\}$ that collide with another value initially sampled in row i. Intuitively, this can cause the counter for a given signature to be large due to many non-heavy elements hashing to the same signature. Because much of the analysis on the correctness of the algorithm is based on the counters of players who have sampled non-heavy elements, this difference must be addressed as well. We overcome both of these problems as follows.

First, after we have progressed far enough to assign the real ID in addition to the signature, we will add a new counter. We will stop incrementing the old counter, and the new counter will count only the frequency of elements with the chosen ID. Thus, we will no longer be counting based on the signature, and we will separate the values counted by the signature from the values counted by the ID. Then, after more rounds, we will switch to using only the new counter and thus the first problem will be fixed. This change creates an additional problem: some appearances of the heavy element might be discarded. We will ensure that the new counter will be polynomially larger than the old counter at the time when it will be discarded. Thus, the change is negligible and will not affect the correctness.

Second, we prove in the full version of the paper [9], that the probability that the counter of a non-heavy competitor will not increase by enough to alter the outcome of the game. We thus show collisions between non-heavy elements does not affect the correctness of the algorithm.

Therefore, by adding the use of hashed signatures to the way we differentiate elements that we can bound the amount of bits used to store all ID numbers and all signatures by $O(n^{1-2/k})$.

2.7 Two Level Sampling

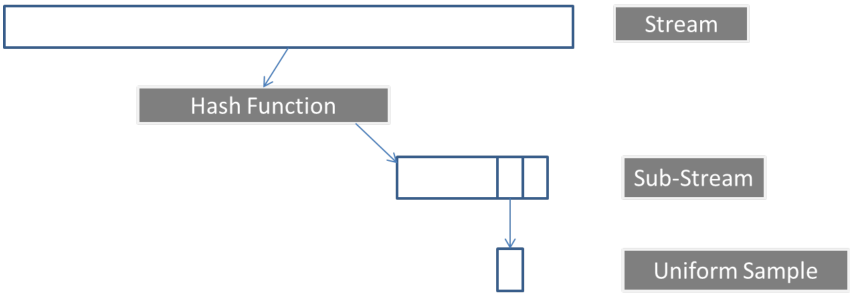

Figure 2 Two-Level Sampling.

In order to successfully use signatures instead of IDs, we also require a new method of sampling. This new method, illustrated above, combines hashing and uniform random sampling from the stream. Informally, we hash the stream and then sample uniformly from the resulting substream. Formally, the algorithm is as follows:

Algorithm 2 Two-Level Sampling(Q, λ)

```
1. Generate pairwise independent hash function $h : [n] \mapsto \{0,1\}$ such that $P(h(i) = 1) = \lambda$.
2. For every element of $p \in Q$: if $h(q) = 1$ then add $q$ to a "pool" $\Gamma$.
3. In parallel, maintain a uniform random sample $L$ from $\Gamma$ using reservoir sampling [45].
4. Return $L$.
```

2.8 An Illustrative Example

In this section we will demonstrate the main steps of our method by considering a simplified problem. Let D be a stream with the following promise: all non-zero frequencies are equal to 1 with the exception of a single element i such that the frequency of i is $f_i \geq n^{1/k}$. Furthermore, $m = \Theta(n)$ and if we split D into intervals of length $O(n^{1-1/k})$ then i appears once in each interval. Clearly, i is the heavy element and the goal of the algorithm will be to find the value of i. This simplified case is interesting because the same promise problem is used for the lower bound in [19] and in many other papers. We will thus illustrate the capability of our method by showing that a bound $O(n^{1-2/k})$ is achievable in this case.

We will assume without loss of generality that $i = 1$. This assumption does not change the analysis but simplifies our notation. In [15] it is shown that $O(n^{1-2/k})$ samples are sufficient to solve the problem. However, each sample requires $\log n$ bits for identification (we will use a notion of "ID" to identify the value of $i \in [n]$.) As well, any known algorithm stores information about the frequency of the heavy element. This can be done by storing a sketch or an explicit approximate counter. In the most direct implementation, $\log m$ bits are required to store the counter. In this example we will assume that $\log n = \Theta(\log m)$ and we will use a single parameter $\log n$.

If $n^{1-2/k}$ independent samples are sampled from each interval then the probability to sample 1 is a constant. Next, observe that most of the time only $O(1)$ bits are needed for the counters since all frequencies except $i = 1$ are either zero or one. Thus, it is sufficient to reduce the bits for IDs.

The key idea is to replace IDs with signatures and uniform sampling with (appropriately chosen) hashing. Combining signatures of constant length with hashing ensures that the number of false positives is relatively small. Specifically, consider a hash function[4] $g : [n] \mapsto [n^{1-1/k}]$ and let the z-th sample of the i-th interval be defined as follows. Let

$$\Gamma_{i,z} = \{j : g(p_j) = z\} \tag{3}$$

where p_j are elements from the i-th interval. To obtain the final sample, we sample one element uniformly at random from $\Gamma_{i,z}$. This sampling schema is *two-level sampling* as described in the previous section. The probability that 1 is sampled using the new sampling method is still a constant. Now consider the case that each sample is represented using a signature of length $O(1)$. Suppose that we store signature SIG for the z-th sample in the i-th interval. The comparison of the sample with another element q of the stream will be defined by the following procedure. We say that they are equal if $g(q) = z$ and the signature of q is equal to SIG. Consider the case when we sample the heavy element. In this case the consecutive appearances of 1 will always be declared equal to the sample. Then, consider another case when l has been sampled and when $f_l = 1$. The probability that there will be any collision in the next interval is at most $2^{-|SIG|}$. Therefore we can exploit the probability gap between these two cases.

Specifically, deleting samples with a small number of collisions allows for increasing signatures for the remaining samples in the future intervals. After 2^γ intervals, it is possible to increase the signature by $O(1)$ bits for $\gamma = 1, 2, \ldots$. In the full version of the paper, [9], we show that the heavy element will never be discarded and that the number of active samples decreases exponentially with γ. Thus, the total expected space for storing the data is $O\left(\frac{n^{1-2/k}\gamma}{2^{\Omega(\gamma)}}\right)$. The aforementioned procedure is called the γ-th round for the i-th interval. At any moment there are at most 2^γ intervals in the γ-th round and the total space is $O(n^{1-2/k})$. For $\gamma = \Omega(\log \log n)$ storing IDs instead of signatures implies that if the heavy element is not discarded then the correct answer is produced. The algorithm works in one pass and uses $O(n^{1-2/k})$ bits.

2.9 Martingale Sketches

Having established a streaming algorithm which can efficiently compute the heavy hitters of a stream, we present a reduction of the problem of frequency moment approximation to that of finding heavy hitters. In general, this analysis will show that the problem of approximating the sum of am implicit vector is the same problem as finding the heavy

[4] It is possible to show that g can be pairwise independent.

elements of that vector, up to a constant factor. As a direct corollary, and using our new heavy hitter algorithm, we obtain a new lowest bound on space required for this problem.

The intuition behind this method is as follows: consider a vector where the sum of its elements cannot be computed directly. If the elements of the vector vary in magnitude, then some elements will have a larger impact on the sum than others. Now consider a second vector made from including or excluding each element of the first by the repeated flip of a fair coin, and then doubling the value of every included element. The expected difference between the sums of the two vectors is 0. But because of the disproportionate contribution of heavy hitters, the actual difference will most likely not be 0. If we can find the heavy hitters of the first vector, we can examine which ones were included and which were excluded in the second vector. Intuitively, the excluded ones will increase the difference between the vector sums, while the included heavy hitters will decrease it (because of the scaling up by a factor of 2, and their already large contribution to the total sum). This allows us to approximate the difference between the two vector sums. If we repeat this process for the second vector and a new vector made from including or excluding each of its elements (with the included elements having their values doubled), and so on, then the repeated differences along with the sum of the final vector can be used together to accurately approximate the sum of the first vector. Thus, finding a frequency moment is reducible to finding the heavy hitters of a series of vectors.

While the overall idea of reducing a vector sum to its heavy hitters is not new, what our algorithm provides is a cost function that is geometric by the nature of the given reduction. Thus, the total space cost required for these computations matches the lower bound for frequency computation, up to a constant factor.

3 Putting It All Together

At this point, we have provided an algorithm that finds heavy hitters in a stream which conforms to certain assumptions. Due to the lack of space, we omit many technical details from the main body of the paper in this abstract. In the full version of this paper,[9], we provide detailed analysis showing that with only several small adjustments the slightly modified version of our original algorithm will work on general streams. With these slight modifications, the technical details of which are included in the appendix, we have succeeded in providing an algorithm which proves Theorem 3.

We also explain how the first and third passes can be removed, and that we can get the algorithm to work for any $k > 3$. Consequently, the reduction proven in the Martingale Sketches section coupled with the AHE algorithm proves our main result, Theorem 8, by providing an algorithm that computes frequency moments with $k > 3$ using $O(n^{1-2/k})$ bits of memory.

References

1 Noga Alon, Yossi Matias, and Mario Szegedy. The space complexity of approximating the frequency moments. *J. Comput. Syst. Sci.*, 58(1):137–147, 1999.

2 Alexandr Andoni, Robert Krauthgamer, and Krzysztof Onak. Streaming algorithms via precision sampling. In *Proceedings of the 2011 IEEE 52nd Annual Symposium on Foundations of Computer Science*, FOCS'11, pages 363–372, Washington, DC, USA, 2011. IEEE Computer Society.

3 Alexandr Andoni, Huy L. Nguyen, Yury Polyanskiy, and Yihong Wu. Tight lower bound for linear sketches of moments. In *Proceedings of the 40th International Conference on*

Automata, Languages, and Programming – Volume Part I, ICALP'13, pages 25–32, Berlin, Heidelberg, 2013. Springer-Verlag.

4 Ziv Bar-Yossef, T. S. Jayram, Ravi Kumar, and D. Sivakumar. An information statistics approach to data stream and communication complexity. In *Proceedings of the 43rd Symposium on Foundations of Computer Science*, FOCS'02, pages 209–218, Washington, DC, USA, 2002. IEEE Computer Society.

5 Ziv Bar-Yossef, T. S. Jayram, Ravi Kumar, D. Sivakumar, and Luca Trevisan. Counting distinct elements in a data stream. In *Proceedings of the 6th International Workshop on Randomization and Approximation Techniques*, RANDOM'02, pages 1–10, London, UK, UK, 2002. Springer-Verlag.

6 Paul Beame, T. S. Jayram, and Atri Rudra. Lower bounds for randomized read/write stream algorithms. In *Proceedings of the thirty-ninth annual ACM symposium on Theory of computing*, STOC'07, pages 689–698, New York, NY, USA, 2007. ACM.

7 Lakshminath Bhuvanagiri, Sumit Ganguly, Deepanjan Kesh, and Chandan Saha. Simpler algorithm for estimating frequency moments of data streams. In *Proceedings of the seventeenth annual ACM-SIAM symposium on Discrete algorithm*, SODA'06, pages 708–713, New York, NY, USA, 2006. ACM.

8 Vladimir Braverman, Ran Gelles, and Rafail Ostrovsky. How to catch l 2-heavy-hitters on sliding windows. In Ding-Zhu Du and Guochuan Zhang, editors, *Computing and Combinatorics*, volume 7936 of *Lecture Notes in Computer Science*, pages 638–650. Springer Berlin Heidelberg, 2013.

9 Vladimir Braverman, Jonathan Katzman, Charles Seidell, and Gregory Vorsanger. Approximating Large Frequency Moments with $O(n^{1-2/k})$ Bits. *CoRR*, abs/1401.1763, 2014.

10 Vladimir Braverman and Rafail Ostrovsky. Smooth histograms for sliding windows. In *Proceedings of the 48th Annual IEEE Symposium on Foundations of Computer Science*, FOCS'07, pages 283–293, Washington, DC, USA, 2007. IEEE Computer Society.

11 Vladimir Braverman and Rafail Ostrovsky. Effective computations on sliding windows. *SIAM J. Comput.*, 39(6):2113–2131, March 2010.

12 Vladimir Braverman and Rafail Ostrovsky. Measuring independence of datasets. In *Proceedings of the 42nd ACM symposium on Theory of computing*, STOC'10, pages 271–280, New York, NY, USA, 2010. ACM.

13 Vladimir Braverman and Rafail Ostrovsky. Recursive sketching for frequency moments. *CoRR*, abs/1011.2571, 2010.

14 Vladimir Braverman and Rafail Ostrovsky. Zero-one frequency laws. In *Proceedings of the 42nd ACM symposium on Theory of computing*, STOC'10, pages 281–290, New York, NY, USA, 2010. ACM.

15 Vladimir Braverman and Rafail Ostrovsky. Approximating large frequency moments with pick-and-drop sampling. Accepted to the 16th. International Workshop on Approximation Algorithms for Combinatorial Optimization Problems (APPROX'2013)., 2013.

16 Vladimir Braverman and Rafail Ostrovsky. Generalizing the layering method of Indyk and Woodruff: Recursive sketches for frequency-based vectors on streams. Accepted to the 16th. International Workshop on Approximation Algorithms for Combinatorial Optimization Problems (APPROX'2013)., 2013.

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

18 Amit Chakrabarti, Graham Cormode, and Andrew McGregor. Robust lower bounds for communication and stream computation. In *Proceedings of the 40th annual ACM symposium on Theory of computing*, STOC'08, pages 641–650, New York, NY, USA, 2008. ACM.

19 Amit Chakrabarti, Subhash Khot, and Xiaodong Sun. Near-optimal lower bounds on the multi-party communication complexity of set disjointness. In *IEEE Conference on Computational Complexity*, pages 107–117, 2003.

20 Don Coppersmith and Ravi Kumar. An improved data stream algorithm for frequency moments. In *Proceedings of the fifteenth annual ACM-SIAM symposium on Discrete algorithms*, SODA'04, pages 151–156, Philadelphia, PA, USA, 2004. Society for Industrial and Applied Mathematics.

21 Graham Cormode. Continuous distributed monitoring: a short survey. In *Proceedings of the First International Workshop on Algorithms and Models for Distributed Event Processing*, AlMoDEP'11, pages 1–10, New York, NY, USA, 2011. ACM.

22 Graham Cormode, Mayur Datar, Piotr Indyk, and S. Muthukrishnan. Comparing data streams using hamming norms (how to zero in). *IEEE Trans. on Knowl. and Data Eng.*, 15(3):529–540, 2003.

23 Mayur Datar, Aristides Gionis, Piotr Indyk, and Rajeev Motwani. Maintaining stream statistics over sliding windows. *SIAM J. Comput.*, 31(6):1794–1813, 2002.

24 J. Feigenbaum, S. Kannan, M. Strauss, and M. Viswanathan. An approximate l1-difference algorithm for massive data streams. In *Proceedings of the 40th Annual Symposium on Foundations of Computer Science*, FOCS'99, pages 501–, Washington, DC, USA, 1999. IEEE Computer Society.

25 Philippe Flajolet and G. Nigel Martin. Probabilistic counting algorithms for data base applications. *J. Comput. Syst. Sci.*, 31(2):182–209, September 1985.

26 Sumit Ganguly. Estimating frequency moments of data streams using random linear combinations. In *APPROX-RANDOM*, pages 369–380, 2004.

27 Sumit Ganguly. Polynomial estimators for high frequency moments. *CoRR*, abs/1104.4552, 2011.

28 Sumit Ganguly. A lower bound for estimating high moments of a data stream. *CoRR*, abs/1201.0253, 2012.

29 Sumit Ganguly and Graham Cormode. On estimating frequency moments of data streams. In *Proceedings of the 10th International Workshop on Approximation and the 11th International Workshop on Randomization, and Combinatorial Optimization. Algorithms and Techniques*, APPROX'07/RANDOM'07, pages 479–493, Berlin, Heidelberg, 2007. Springer-Verlag.

30 Phillip B. Gibbons and Srikanta Tirthapura. Distributed streams algorithms for sliding windows. In *Proceedings of the fourteenth annual ACM symposium on Parallel algorithms and architectures*, SPAA'02, pages 63–72, New York, NY, USA, 2002. ACM.

31 Andre Gronemeier. Asymptotically optimal lower bounds on the nih-multi-party information. *CoRR*, abs/0902.1609, 2009.

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

33 Piotr Indyk and David Woodruff. Optimal approximations of the frequency moments of data streams. In *STOC'05: Proceedings of the thirty-seventh annual ACM symposium on Theory of computing*, pages 202–208, New York, NY, USA, 2005. ACM.

34 T. S. Jayram, Andrew McGregor, S. Muthukrishnan, and Erik Vee. Estimating statistical aggregates on probabilistic data streams. In *PODS'07: Proceedings of the twenty-sixth ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems*, pages 243–252, New York, NY, USA, 2007. ACM.

35 T. S. Jayram and David Woodruff. Optimal bounds for johnson-lindenstrauss transforms and streaming problems with sub-constant error. In *Proceedings of the Twenty-Second Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA'11, pages 1–10. SIAM, 2011.

36 T.S. Jayram. Hellinger strikes back: A note on the multi-party information complexity of and. In Irit Dinur, Klaus Jansen, Joseph Naor, and José Rolim, editors, *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques*, volume 5687 of *Lecture Notes in Computer Science*, pages 562–573. Springer Berlin Heidelberg, 2009.

37 Daniel M. Kane, Jelani Nelson, and David P. Woodruff. 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'10, pages 1161–1178, Philadelphia, PA, USA, 2010. Society for Industrial and Applied Mathematics.

38 Daniel M. Kane, Jelani Nelson, and David P. Woodruff. An optimal algorithm for the distinct elements problem. In *Proceedings of the twenty-ninth ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems*, PODS'10, pages 41–52, New York, NY, USA, 2010. ACM.

39 Ping Li. Compressed counting. In *Proceedings of the twentieth Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA'09, pages 412–421, Philadelphia, PA, USA, 2009. Society for Industrial and Applied Mathematics.

40 Yi Li and DavidP. Woodruff. A tight lower bound for high frequency moment estimation with small error. In Prasad Raghavendra, Sofya Raskhodnikova, Klaus Jansen, and JoséD.P. Rolim, editors, *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques*, volume 8096 of *Lecture Notes in Computer Science*, pages 623–638. Springer Berlin Heidelberg, 2013.

41 Morteza Monemizadeh and David P. Woodruff. 1passs relative-error lp-sampling with applications. In *Proceedings of the Twenty-First Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA'10, pages 1143–1160, Philadelphia, PA, USA, 2010. Society for Industrial and Applied Mathematics.

42 S. Muthukrishnan. Data streams: algorithms and applications. *Found. Trends Theor. Comput. Sci.*, 1(2):117–236, 2005.

43 Jelani Nelson and David P. Woodruff. Fast Manhattan sketches in data streams. In *PODS'10: Proceedings of the twenty-ninth ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems of data*, pages 99–110, New York, NY, USA, 2010. ACM.

44 Eric Price and David P. Woodruff. Applications of the shannon-hartley theorem to data streams and sparse recovery. In *ISIT*, pages 2446–2450, 2012.

45 J. S. Vitter. *Random sampling with a reservoir.* ACM Transactions on Mathematical Software, v.11 n.1, pp.37–57, 1985.

46 David Woodruff. Optimal space lower bounds for all frequency moments. In *Proceedings of the fifteenth annual ACM-SIAM symposium on Discrete algorithms*, SODA'04, pages 167–175, Philadelphia, PA, USA, 2004. Society for Industrial and Applied Mathematics.

47 David P. Woodruff. Frequency moments. In *Encyclopedia of Database Systems*, pages 1169–1170. Springer, 2009.

48 David P. Woodruff and Qin Zhang. Tight bounds for distributed functional monitoring. In *Proceedings of the 44th symposium on Theory of Computing*, STOC'12, pages 941–960, New York, NY, USA, 2012. ACM.