Competitive Analysis of Maintaining Frequent Items of a Stream[*]

Yiannis Giannakopoulos[†] Elias Koutsoupias[‡]

May 31, 2013

Abstract

We study the classic frequent items problem in data streams, but from a competitive analysis point of view. We consider the standard worst-case input model, as well as a weaker distributional adversarial setting. We are primarily interested in the single-slot memory case and for both models we give (asymptotically) tight bounds of $\Theta(\sqrt{N})$ and $\Theta(\sqrt[3]{N})$ respectively, achieved by very simple and natural algorithms, where N is the stream's length. We also provide lower bounds, for both models, in the more general case of arbitrary memory sizes of $k \geq 1$.

1 Introduction

The *frequent items* problem [8] is one of the most well-studied ones in the area of data streams [18, 1, 14, 12, 4]. Informally, the problem is that of observing a stream (sequence) of values and trying to discover those that appear most frequently. Many applications in packet routing, telecommunication logging and tracking keyword queries in search machines are critically based upon such routines.

More formally, in the most basic version of the classic frequent items problem, we are given a stream $a_1, a_2, \dots, a_N$ of items from some universe, as well as a frequency threshold ϕ, $0 < \phi < 1$, and we are asked to find and/or maintain all items that occur more than ϕN times throughout the stream. For real-life applications there are some restricting assumptions the algorithms need to respect: the size N of the stream, as well as the rate at which the items arrive, far exceed the computational capabilities of our devices. Consequently, we usually require streaming algorithms to use $O(polylog(N))$ memory and allow approximate solutions within a factor of ε (additive, with respect to ϕ), since exact solutions would require linear space [8], a totally unrealistic option in some domains. Furthermore, we are usually interested in algorithms that make as few passes as possible over the input stream and, in particular, *single-pass* algorithms that process the input stream in an online way, i.e. each item sequentially, making decisions on-the-fly.

So, traditionally data stream problems have been essentially approached as space complexity optimization problems: given an input stream and an approximation guarantee of ϵ, we try to minimize the memory used (see, e.g., the seminal work of [2]). However, despite their intrinsic *online* nature, these problems have not been studied within the predominant framework for studying online problems, i.e. that of competitive analysis [6].

[*]A preliminary version of this paper, not including the results in Section 5 about ephemeral frequencies, appeared in [13].

This work was supported by ERC Advanced Grant 321171 (ALGAME) and the ESF-NSRF research program Thales (ALGONOW). A significant part of it was done while the authors were at the Department of Informatics of the University of Athens.

[†]Department of Computer Science, University of Oxford. Email: `ygiannak@cs.ox.ac.uk`

[‡]Department of Computer Science, University of Oxford. Email: `elias@cs.ox.ac.uk`

Only recently, Becchetti and Koutsoupias [5] did that, using competitive analysis to study another important streaming problem, namely that of maintaining the maximum value within a sliding window [9]. Following a similar approach, we formulate an online version of the classic frequent items problem: given an input stream and a memory of at most k slots, try to optimize the online algorithm's performance with respect to that of an optimal offline algorithm that knows the entire input stream in advance. This is, in a way, the inverse of the traditional space complexity optimization objective mentioned above. We also use a similar aggregate-through-time perspective (instead of an unrealistic optimization-at-every-step requirement) upon defining our objective function (see section 1.1). As argued in [5], the competitive analysis approach combined with an aggregate objective, seem more appropriate for economic applications as well as a decision-making under uncertainty framework.

1.1 Setting

We are observing a stream $A = a_1, a_2, \dots, a_N$ of N elements drawn from some universe $\mathcal{U}$, in a sequential, *online* fashion. It is natural to assume that $|\mathcal{U}| \gg N$. We consider algorithms that, at every time step $t = 1, 2, \dots, N$, maintain in memory a set $S_t(A) \subseteq \{a_1, a_2, \dots, a_t\}$ of at most k items from the part of the input stream A observed so far[1]. Furthermore, we assume that the only way in which our algorithms can update these memory sets throughout the execution, is to make an *irrevocable* decision, at every time point t, of whether or not to store the newly arrived element a_t in memory, i.e. $S_t \subseteq S_{t-1} \cup \{a_t\}$. An *online* algorithm can base its decision just[2] on the knowledge of its current memory state S_{t-1} and the current time point t, while an offline algorithm can have access to the entire input stream A. Notice that, since $|S_t| \leq k$, if at some point t we want to store a new element $a_t \notin S_{t-1}$, then we may need to discard some previously stored item $a_j \in S_{t-1}$, $j < t$ and then $S_t \subseteq \{a_t\} \cup S_{t-1} \setminus \{a_j\}$. For the special case of $k = 1$, which will be our main concern in this paper for the most part (we will consider general memory sizes of $k \geq 1$ again in Section 4) we will denote by s_t the unique item in the algorithm's memory at time point t, i.e. $S_t = \{s_t\}$.

Given an input stream A and an element $a \in A$ we define its *frequency* as $f^A(a) = \frac{n^A(a)}{N}$, where $n_A(a) = |\{i \mid a_i = a\}|$ is the number of instances of a in the stream[3]. Intuitively, we want our algorithms, at every time, to maintain the most frequent items possible. We formalize this, by defining the *aggregate frequency* objective as the sum, across the entire execution, of the frequencies of all *distinct* items in memory, i.e. $\sum_{t=1}^{N} \sum_{a \in S_t} f(a)$. Notice here a fine point: we treat S_t as a set and *not* as a multi-set, i.e. multiple occurrences of the same element in memory can only contribute *once* towards our objective. We measure an online algorithm's performance on a given input A by comparing its total gain (i.e. the value of the aggregate frequency objective on stream A) to that of an offline algorithm that knows the entire input stream A in advance. The *competitive ratio* of the online algorithm is the maximum value of this ratio among all possible inputs,

$$\max_A \frac{\sum_{t=1}^{N} \sum_{a \in S'_t} f^A(a)}{\sum_{t=1}^{N} \sum_{a \in S_t} f^A(a)},$$

where S_t, S'_t are the memory sets of the online and optimal offline algorithm, respectively. The competitive ratio for our online frequent items problem, is the best (minimum) competitive ratio we can achieve over all online algorithms.

Finally, whenever we deal with randomized algorithms in this paper, we are always silently assuming the standard, oblivious adversary [11, 6] model, i.e. the adversary decides an input A knowing the online algorithm but not the actual results of its coin tosses.

[1] To keep notation light, we will simply use S_t instead of $S_t(A)$ whenever it is clear to which input stream we are referring to.

[2] In a different online setting, we could have also assumed that the online algorithm has complete knowledge of the past. This, though, seems as a rather unrealistic assumption to make, especially in a streaming setting.

[3] Again, we will drop the superscript A whenever this causes no confusion.

1.2 Organization of the paper and results

In this paper we are mostly interested in the special case of single-slot memories ($k = 1$). That is the case for Sections 2 and 3. We do not deal with general memory sizes of $k \geq 1$ until Section 4.

In Section 2, we prove that the competitive ratio of the online frequent items problem is $\Theta(\sqrt{N})$, by providing a lower bound proof of $\frac{1}{3}\sqrt{N}$ and showing that the most simple algorithm that myopically accepts every element that arrives achieves an (asymptotically) tight $\sqrt{N}$ competitive ratio. Furthermore, in Section 2.1, we show that the well known Majority algorithm for the classical frequent items problem performs very poorly from a competitive analysis point of view, since it has (asymptotically) the worst possible competitive ratio an online algorithm can demonstrate, namely $\Theta(N)$.

Also, we consider weaker adversarial inputs and in particular the case of the input stream being generated i.i.d. from a probability distribution, known only to the adversary. In section 3 we show how a very simple and natural algorithm, called Eager, that essentially waits until it sees some element appearing twice, achieves a competitive ratio of $O(\sqrt[3]{N})$, asymptotically matching a lower bound of $\Omega(\sqrt[3]{N})$ again providing a tight competitive ratio (of $\Theta(\sqrt[3]{N})$) for the case of $k = 1$.

Next, in Section 4 we deal with the more general case of arbitrary memory sizes of $k \geq 1$, for which we give lower bounds of $\frac{1}{3}\sqrt{N/k}$ and $\Omega(\sqrt[3]{N/k})$, respectively, for both the worst-case and weaker distributional adversarial models. In fact, the lower bounds of previous Sections 2 and 3 are derived by simply setting $k = 1$ at these more general bounds.

Finally, in section 5 we provide an extensive discussion regarding interesting possible extensions to our model and open problems.

2 Worst-case Bounds

As the following Theorem 1 demonstrates, one can not hope for online algorithms with bounded competitive ratios. In particular:

Theorem 1. *No (randomized) algorithm for the online frequent items problem can have a competitive ratio better than* $\frac{1}{3}\sqrt{N} - o(1)$.

Proof. The proof can be found in section 4 where we prove the more general Theorem 8 for arbitrary memory sizes of $k \geq 1$. □

The lower bound of Theorem 1 is (asymptotically) tight and achieved by the most simple deterministic algorithm:

Definition 1 (Algorithm Naive). The deterministic Naive algorithm accepts every element as it arrives. Formally, $s_t = a_t$ for all $t = 1, 2, \ldots, N$.

Theorem 2. *The* Naive *algorithm is* $\sqrt{N}$*-competitive.*

Proof. Fix a worst-case input stream A for the Naive algorithm, let α be an element of the stream having the highest frequency, i.e. $\alpha = \operatorname{argmax}_{a \in A} f(a)$ and let $f = f(\alpha)$. First, notice that the optimal offline gain cannot exceed Nf (by simply giving to it α from the very start).

Next, we provide two lower bounds on the online gain. First, we have a trivial lower bound of $N\frac{1}{N} = 1$, since every element stored at any time t in our memory has a frequency of at least $\frac{1}{N}$. Next, since every element $a \in A$ is being accepted as it arrives, we have it in our memory for at least $Nf(a)$ times for a gain of $f(a)$ per time, so the online gain is at least $Nf \cdot f = Nf^2$.

Consider two cases: If $f < \frac{1}{\sqrt{N}}$, then we use the first bound on the online gain to get a competitive ratio of at most $\frac{Nf}{1} < \frac{N}{\sqrt{N}} = \sqrt{N}$ and at the complimentary case of $f \geq \frac{1}{\sqrt{N}}$ we use the other bound to get, again, a competitive ratio of $\frac{Nf}{Nf^2} = \frac{1}{f} \leq \sqrt{N}$. □

2.1 The classic Majority algorithm

As was discussed in our introduction, if we demand single-pass algorithms then the traditional data streaming setting is essentially an online setting and so, some algorithms for the classic frequent items problem are valid algorithms for our online version too. This is true for the *counter-based* algorithms (see [8]) and in particular for the well-known Majority algorithm [7], for the case of $k = 1$, and its generalizations (see, e.g., the Misra-Gries algorithm [15]) for the case of $k > 1$.

The Majority algorithm keeps a single item in memory, along with a counter, initialized to 0. If the next value that arrives is the same as the one currently stored, then increase the counter by $+1$, otherwise decrease it by -1. Whenever the counter reaches 0, flush the memory and accept the next item to arrive, increasing the counter to 1. This simple and clever algorithm manages to solve the 1/2-approximation classic frequent items problem (i.e. if there is a majority element, appearing more than half the time, it is then in the algorithm's memory at the end of the execution) using just a single memory slot. However, as Theorem 3 demonstrates, from a competitive analysis point of view, its performance is disappointingly poor. The intuition behind this failure, which is the reason why other classic algorithms would also not perform efficiently for the online problem, is that the ε-approximate deviation per step has a drastic effect when aggregated across the entire execution as $N \to \infty$.

It is a simple observation that no online algorithm for the frequent items problem can perform worse than N with respect to the competitive ratio: at every given step, the offline gain can not exceed 1 and the online gain can not be less than $1/N$, since for every element α in the stream: $\frac{1}{N} \leq f(\alpha) \leq 1$.

Theorem 3. *The Majority algorithm (asymptotically) achieves the worst possible competitive ratio of $\Theta(N)$.*

Proof. Fix a stream length N, N being even, and pick $\frac{N}{2}+1$ distinct elements $\alpha, \alpha_1, \alpha_2, \dots, \alpha_{N/2} \in \mathcal{U}$. Give as input the stream

$$\alpha_1, \alpha, \alpha_2, \alpha, \dots, \alpha_{N/2}, \alpha.$$

On this input, Majority is forced to change every two steps, with $s_{2i-1} = s_{2i} = \alpha_i$ for all $i = 1, 2, \dots, \frac{N}{2}$, never having the desired element α in memory, resulting to a disappointing gain of $N \cdot \frac{1}{N} = 1$, while the offline strategy that puts element α in memory as soon as it arrives at time point $t = 2$ and keeps it until the end achieves a gain of $\frac{1+(N-1)\frac{N}{2}}{N}$, giving a competitive ratio of at least $\frac{N}{2} - \frac{1}{2}$. □

3 Weaker adversarial models

Here we consider a distributional model where the adversary, instead of explicitly selecting the input sequence stream A, now just decides on a particular probability distribution, unknown to the online algorithm, which we sample to construct the input.

Definition 2 (Distributional adversarial model)**.** In the distributional model the adversary decides the stream's length N and picks a probability distribution $\mathcal{D}$ on the universe of elements $\mathcal{U}$. The input stream A is generated by drawing N observations i.i.d. from $\mathcal{D}$. $\mathcal{D}$ is not known to the online algorithms.

It is interesting to notice that this type of adversary resembles the random order input model used in the classical secretary problems as well as recent online auctions settings (see, e.g.,[3, 10]). A random order streaming model was first proposed in the seminal paper of Munro and Paterson [17].

Notation. In this section, we will use the following notation: Let N be the length of the stream, $\mathcal{D}$ be the distribution of the input, having support $D = \{\alpha_1, \alpha_2, \alpha_3, \dots\}$ with probability p_i corresponding to element α_i. We set $q = \sum_{\alpha_i \in D} p_i^2$ and without loss of generality, assume that $p_1 \geq p_2 \geq \dots$. Since the items of the stream are selected i.i.d. from $\mathcal{D}$, the expected frequency of element α_i at the resulting stream is simply $f(\alpha_i) = p_i$, for all $\alpha_i \in D$.

We first provide a lower bound. Although obviously better than that of the worst-case model of section 2, it still remains unbounded with respect to the stream's length N:

Theorem 4. *The competitive ratio of the online frequent items problem for the adversarial model is $\Omega(\sqrt[3]{N})$.*

Proof. As with Theorem 1, the proof can be found in section 4 for the more general Theorem 9 for arbitrary memory sizes of $k \geq 1$. □

Now we turn our attention to upper bounds, and in search for an online algorithm that, used in our current model of distributional inputs, will break the $\Omega(\sqrt{N})$ bound for the worst-case model of Section 2, the first question we should deal with is whether a trivial algorithm such as the NAIVE algorithm of Definition 1, which does not even take into consideration its memory set S_{t-1} before making a decision, can achieve this. The answer is negative:

Theorem 5. *Every (randomized) online algorithm that does not take into consideration its memory set, i.e., for all time points t, its probability of accepting incoming element a_t can depend only on the current absolute execution time t and not S_{t-1}, has a competitive ratio of $\Omega(\sqrt{N})$ in the distributional adversarial model.*

Proof. Fix a stream length N with N being an even, perfect square positive integer and let $m = \sqrt{N}$. Consider a probability distribution $\mathcal{D}$ with support $\{\alpha_0, \alpha_1, \dots, \alpha_{N-m}\} \subseteq \mathcal{U}$, for which $p_0 = m/N$ and $p_j = 1/N$ for all $j = 1, 2, \dots, N-m$. Let P_t denote the probability of accepting the newly arrived element a_t at time t and $Q_t = \Pr[s_t = \alpha_0]$, the probability of having element α_0 stored in memory at time t. Then

$$Q_{t+1} = Q_t \cdot [1 - (1-p_0) P_{t+1}] + (1 - Q_t) \cdot p_0 P_{t+1} = P_{t+1}\left(\frac{m}{N} - Q_t\right) + Q_t,$$

which, by the fact that $Q_1 = m/N$ (since at the first step any online algorithm just puts item a_1 in memory), gives that $Q_t = \frac{m}{N}$ for all t. That means that the expected gain of our online algorithm is

$$\begin{aligned}\sum_{t=1}^{N} [Q_t f(\alpha_0) + (1-Q_t) f(\alpha_1)] &= \sum_{t=1}^{N} \left[Q_t \frac{m}{N} + (1-Q_t)\frac{1}{N}\right] \\ &= \frac{m^2}{N} - \frac{m}{N} + 1 = 2 - \frac{1}{\sqrt{N}}.\end{aligned}$$

The offline algorithm that waits until it sees the desired element α_0 and then stores it in memory forever has asymptotically an expected gain of at least $\frac{N}{2} f(\alpha_0) = \frac{N}{2}\frac{m}{N} = \frac{\sqrt{N}}{2}$, since the probability of having discovered α_0 until the middle of the stream is $1 - (1-p_0)^{N/2} = 1 - \left(1 - \frac{1}{\sqrt{N}}\right)^{N/2} \to 1$, as $N \to \infty$. □

Theorem 5 above tells us that we should consider only non-trivial algorithms that consult their memory before making a decision. The simplest of them is the following:

Definition 3 (Algorithm EAGER)**.** The EAGER algorithm accepts every element as it comes. If it finds the same element in two consecutive positions, then it keeps it until the end. Formally, let

$$t^* = \min_{1 \leq t \leq N-1} \{t \mid a_t = a_{t+1}\}$$

if that exists, otherwise $t^* = N$. Then EAGER is the algorithm with $s_t = a_t$ for all $t \leq t^*$ and $s_t = a_{t^*}$ for all $t > t^*$.

First, we need to bound the optimal offline gain:

Theorem 6. *The expected optimal offline gain of the distributional model is* $O(\max\{N^{1/3}, p_1 N\})$.

Proof. Let X_i be the random variable representing the number of appearances of element α_i at the resulting stream (generated i.i.d. from $\mathcal{D}$). Then, X_i follows a binomial distribution with parameters p_i, N. It is clear that the expected optimal offline gain cannot exceed $\max_i X_i$ ($\frac{\max_i X_i}{N}$ per step). Also, let $M = \max\{N^{1/3}, p_1 N\}$, $I_1 = \left\{i \mid p_i N \geq N^{1/3}\right\}$ and $I_2 = \left\{i \mid p_i N < N^{1/3}\right\}$.

We will need the following lemma:

Lemma 1. *For all* $i \in I_1$, $\Pr[X_i > 6p_i N] \leq 2p_1 p_i$ *and for all* $i \in I_2$, $\Pr\left[X_i > 6N^{1/3}\right] \leq 2p_i N^{-2/3}$.

Proof. If $i \in I_1$, then $p_1 \geq p_i \geq N^{-2/3}$ and using a Chernoff bound (see [16, Theorem 4.4]) we get

$$\Pr[X_i > 6p_i N] \leq 2^{-6p_i N} \leq 2p_1 p_i,$$

since $\frac{2^{-6p_i N}}{p_i p_1} \leq \frac{2^{-6p_i N}}{p_i^2} \leq 2^{-6N^{1/3}} N^{4/3} \leq 2$ for all N, the second inequality holding by substituting $p_i = N^{-2/3}$ at a monotonically decreasing function.

If $i \in I_2$, then $p_i < N^{-2/3}$ and

$$\begin{aligned}\Pr\left[X_i > 6N^{1/3}\right] &\leq \sum_{j=6N^{1/3}}^{N} \binom{N}{j} p_i^j (1-p_i)^{N-j} \\ &\leq p_i^{6N^{1/3}} \binom{N}{6N^{1/3}} {}_2F_1(6n^{1/3} - n, 1; 1 + 6n^{1/3}; -\frac{p_i}{1-p_i}),\end{aligned}$$

where ${}_2F_1$ is the Gaussian hypergeometric function, maximized at $p_i = N^{-2/3}$ for a value of less than 2 (for all N), giving an upper bound, for the above probability, of $2p_i^{2N^{1/3}}\binom{N}{6N^{1/3}}$. So, we need to show that $p_i^{6N^{1/3}}\binom{N}{6N^{1/3}} \leq p_i N^{-2/3}$, i.e. it is enough to show that

$$N^{2/3} p_i^{6N^{1/3}-1} \binom{N}{6N^{1/3}} \leq N^{2/3} \left(N^{-2/3}\right)^{6N^{1/3}-1} \binom{N}{6N^{1/3}} \leq 1,$$

which holds for all N, concluding the proof of the lemma. □

Now we are ready to bound the expected optimal offline gain:

$$\begin{aligned}\mathbf{E}[\max X_i] &\leq 6M \cdot \Pr[\max X_i \leq 6M] + N \cdot \Pr[\max X_i > 6M] \\ &\leq 6M + N\left(\sum_{i \in I_1} \Pr[X_i > 6M] + \sum_{i \in I_2} \Pr[X_i > 6M]\right) \\ &\leq 6M + N\left(\sum_{i \in I_1} \Pr[X_i > 6p_i N] + \sum_{i \in I_2} \Pr\left[X_i > 6N^{1/3}\right]\right),\end{aligned}$$

since $M \geq p_1 N \geq p_i N$ and $M \geq N^{1/3}$, and so by Lemma 1

$$\mathbf{E}[\max X_i] \leq 6M + N\left(\sum_{i \in I_1} 2p_1 p_i + \sum_{i \in I_2} 2N^{-2/3} p_i\right) \leq 6M + 2M \sum_i p_i = 8M.$$

□

Next, we turn our attention to analyzing the online gain of our algorithms:

Lemma 2. *The* Naive *algorithm (see Definition 1) has an expected gain of at least* $1 + q(N-1)$ *for the distributional model, where* $q = \sum p_i^2$.

Proof. Notice, that, since the NAIVE algorithm accepts every element to arrive, at every step it has element α_i stored in memory with probability p_i, for an expected (frequency) gain of $\frac{1+p_i(N-1)}{N}$ at this step, resulting to a total expected gain of $1 + q(N-1)$. □

We are ready now to give our upper bound, asymptotically matching that of Theorem 4:

Theorem 7. *The randomized algorithm that runs* NAIVE *and* EAGER *with an equal probability of* $\frac{1}{2}$ *is* $O(\sqrt[3]{N})$*–competitive for the distributional model.*

Proof. From Lemma 2 and Theorem 6 we only need to show that EAGER is $O(N^{1/3})$–competitive for the case when $p_1 N > N^{1/3}$ and $\frac{p_1 N}{1+q(N-1)} = \omega(N^{1/3})$, which give the following necessary conditions:

$$p_1 > N^{-2/3} \quad \text{and} \quad q < \frac{p_1}{N^{1/3}} \ . \tag{1}$$

Let t^* be as in Definition 3. Then, since the stream is generated i.i.d. from $\mathcal{D}$, $\Pr[t^* \geq t] = \prod_{i=1}^{t-1} \Pr[a_i \neq a_{i+1}] = \prod_{i=1}^{t-1}(1 - \Pr[a_i = a_{i+1}]) = \prod_{i=1}^{t-1}(1 - \sum_i p_i p_i) = (1-q)^{t-1}$ for $t \leq N-1$, and so the probability of EAGER "discovering" α_1 at time $t+1$ is $(1-q)^{t-1} p_1^2$, for an expected gain of at least $\frac{2+(N-2)p_1}{N}(N-t+1) \geq p_1(N-t+1)$, resulting to a total expected gain of at least

$$\sum_{t=1}^{N-1}(1-q)^{(t-1)} p_1^2 \cdot p_1(N-t+1) = p_1^3 \frac{(1-2q)(1-q)^{N-1} + q(N+1) - 1}{q^2}$$

This, together with Theorem 6 and (1), gives a competitive ratio upper bound of

$$\frac{F(q,N)}{p_1^2} \quad \text{where} \quad F(q,n) = \frac{q^2 N}{(1-2q)(1-q)^{N-1} + q(N+1) - 1} \ .$$

It is not difficult to see that $F(q,n)$ is an increasing function with respect to q, thus by (1) getting an upper bound of

$$\frac{F(p_1/N^{1/3}, N)}{p_1^2} = \frac{N^{1/3}}{\left(1 - \frac{2p_1}{n^{1/3}}\right)\left(1 - \frac{p_1}{N^{1/3}}\right)^{N-1} + p_1\left(\frac{1}{N^{1/3}} + N^{2/3}\right) - 1} \ .$$

The denominator in the above expression is an increasing function with respect to p_1 and so, by (1), it is minimized by setting $p_1 = N^{-2/3}$ for a value of $\left(1 - \frac{2}{N}\right)\left(1 - \frac{1}{N}\right)^{N-1} + \frac{1}{N} \to \frac{1}{e}$, as $N \to \infty$, giving the desired upper bound of $O(N^{1/3})$ on our competitive ratio. □

4 Lower bounds for arbitrary memory k

In this section we consider an arbitrary memory size of $k \geq 1$, generalizing the single-slot memory settings of the previous Sections 2 and 3 and we extend the lower bounds for both the worst-case and the weaker distributional adversarial models considered in these sections. Obviously, for $k = 1$ our results here match those of Theorems 1 and 4, respectively.

Theorem 8. *No (randomized) algorithm for the online frequent items problem can have a competitive ratio better than* $\frac{1}{3}\sqrt{\frac{N}{k}} - o(1)$.

Proof. We use Yao's principle [20, 6]. Fix a stream length N and set $m = \sqrt{\frac{N}{k}}$ (which we assume to be integer). Fill the stream's first $N - km$ positions, with $N - km$ distinct elements $a_1, \ldots, a_{N-km} \in \mathcal{U}$. The probabilistic input is constructed by selecting uniformly at random k distinct elements $a_{j_1}, \ldots, a_{j_k}$ from the first km positions and using m copies of each to fill the remaining last km slots of the stream.

An offline algorithm picks an element α_{j_i} as soon as it appears, an event which occurs at step km at the latest, and keeps it until the end, resulting in a total gain of at least $\frac{(N-km)\cdot(m+1)}{N} = m - \frac{1}{m}$ for each of the k positions in memory.

By the uniform way in which our input is constructed, it is easy to see that the best online deterministic strategy is to pick the first k elements, keep them until position $N - km$ and then gradually replace them with the k distinct elements of the last km positions. For a given memory slot, the probability that the online algorithm will succeed to initially pick one of the high-frequency items is at most $\frac{k}{km} = 1/m$. In this case the gain is at most $N\frac{m+1}{N} = m + 1$. Otherwise, the gain is at at most $(N - km)\frac{1}{N} + km\frac{m+1}{N} = 1 + \frac{km^2}{N} = 2$. Therefore the expected online gain per each memory slot is at most $(m + 1)\frac{1}{m} + 2(1 - \frac{1}{m}) \leq 3$. It follows that the ratio of the optimal gain over the online gain is at least $\frac{1}{3}(m - \frac{1}{m}) = \frac{1}{3}\sqrt{\frac{N}{k}} - o(1)$. □

It is easy to see why an analysis of some natural extension[4] of the Naive algorithm (see Definition 1) for the single-slot memory case can not give us a better competitive ratio than that of the simple case of $k = 1$ in Theorem 2. Simply construct an input of k consecutive distinct blocks, each block holding exactly $\sqrt{N/k}$ instances of a unique element. After these blocks, fill the remaining stream with "dummy" elements from $\mathcal{U}$ appearing only once. In such inputs where we have consecutive appearances of the same element, Naive's memory is essentially rendered useless and acts like a single-slot memory algorithm, since in our model regardless of how many times an element resides in memory at a given time instance, it can only be counted once towards the online gain.

Theorem 9. *The competitive ratio of the online frequent items problem for the distributional adversarial model is $\Omega(\sqrt[3]{N/k})$.*

Proof. We use again Yao's principle and give a distributional input $\mathcal{D}$ with k elements $\alpha_1, \alpha_2, \ldots, \alpha_k$ (not known to the online algorithm) having a probability of $p = (kN^2)^{-1/3}$ and all other elements in $\mathcal{U}$ being assigned arbitrarily small probabilities of $\varepsilon \to 0$ (we are assuming a very large universe $|\mathcal{U}| \gg k$). Notice that this is a valid probability distribution, since $kp < 1$. By this construction, it is safe to assume that no element except $\alpha_1, \alpha_2, \ldots, \alpha_k$, appears more than once in the resulting stream $a_1, a_2, \ldots, a_N$, since such an event occurs with very small, negligible probability. So, when an online algorithm observes an item that it is already in its memory (i.e. $a_t \in S_{t-1}$), it knows for sure that this is one of the "good" elements $\alpha_1, \alpha_2, \ldots, \alpha_k$ and, furthermore, until such a "marking" occurs the online algorithm can have no information on the identities of these desired elements, meaning that, until then, the probability of having α_i residing in a particular memory slot is at most p, for all $i = 1, 2, \ldots, k$ and all memory slots. Thus, the probability of discovering α_i, $i = 1, 2, \ldots, k$, at some time point t can not exceed $kp \cdot p$ (at most k slots are still "free" at time t and each is holding α_i with a probability of at most p) an event giving an expected gain of at most Np (assume that all discovered elements can be stored, that we are receiving gain from all possible multiple copies of the same element and that this gain is taken from the start $t = 1$). Also, if we don't discover any "good" element throughout the execution, we can trivially get an expected gain of at most $kp \cdot p + (1 - kp)\max\left\{\varepsilon, \frac{1}{N}\right\} = kp^2 + \frac{1-kp}{N}$ per step and per slot. Summing up, the expected total gain of every online deterministic algorithm is at most

$$N \cdot k \cdot Nkp^3 + N \cdot k \cdot \left[kp^2 + \frac{1 - kp}{N}\right] \leq 2k + N^{-1/3}k^{4/3} \leq 3k,$$

since $p = (kN^2)^{-1/3}$ and $k \leq N$.

Finally, consider the offline algorithm that waits until it sees element α_i for the first time and then stores it in the i-th memory slot forever (for an expected gain of at least p per step). Notice that the probability that an element α_i has not been stored until halfway

[4]For example, consider the algorithm that accepts every element as it arrives and replaces the items in memory in a FIFO way. Formally, $S_t = \{a_t, a_{t-1}, \ldots, a_{t-k+1}\}$ for all $t = 1, 2, \ldots, N$ (where $a_t = a_1$ for all $t \leq 0$).

the execution is $(1-p)^{N/2} \leq 1/\sqrt{e}$ (since $p \geq 1/N$), so the expected offline gain is at least $k\left(1-\frac{1}{\sqrt{e}}\right)\frac{N}{2}p = \frac{(1-e^{-1/2})}{2}N^{1/3}k^{2/3}$, giving the desired competitive ratio. $\square$

5 Extensions and future directions

The most obvious open problem based on our work is that of closing the gap, in both adversarial models, between the upper and lower bounds of the competitive ratios with respect to the memory size parameter k in the general case of arbitrary memory sizes of $k > 1$.

Throughout this paper we have used an *absolute* frequency notion, i.e. we take an element's a count $n^A(a)$ (number of appearances) with respect to the entire stream A. Alternatively, one can define an *ephemeral* frequency $f_t^A(a)$, dependent on the current time point t and refer to element's a count up to that point, i.e.

$$f_t^A(a) = \frac{|\{i \leq t \mid a_i = a\}|}{t}. \tag{2}$$

Notice that, as expected, $f(a) = f_N(a)$. Our aggregate frequency objective can be naturally extended to $\sum_{t=1}^{N}\sum_{a\in S_t} f_t(a)$.

Intuitively, one would expect the competitive ratios for ephemeral frequencies to remain essentially unchanged for the distributional adversarial model, since the expected ephemeral frequency of an element at a given time t would be equal to its expected absolute frequency (resulting to equal expected gains per step), while in the worst-case model we can hope for better performance: the "nemesis" inputs similar to that of the lower bound proof of Theorem 1 no longer apply in such a disastrous way.

Indeed, for the distributional adversarial model our results still hold within logarithmic factors:

Theorem 10. *The competitive ratio of the online* ephemeral *frequent items problem for the distributional adversarial model is $\tilde{\Omega}(\sqrt[3]{N/k})$. For the single-item case this gives a tight $\tilde{\Theta}(\sqrt[3]{N})$ competitive ratio, achieved (again) by the algorithm described in Theorem 7.*

Proof. For the lower bound, one can show a bound of $\Omega\left(\frac{\sqrt[3]{N/k}}{\ln N}\right)$ by following the proof of Theorem 9. We only have to notice that the expected gain in the case of not discovering any "good" element is now at least $k\cdot\sum_{t=1}^{N}\left[kp^2+(1-kp)\frac{1}{t}\right] \geq Nk^2p^2+(1-kp)\ln(N+1)$ and carry out the remaining calculations.

For the upper bound, the performance of the EAGER-NAIVE algorithm is now at least as good as that presented in Theorem 7: all the lower bounds on the algorithm's gain per step and per slot still hold and also we can imitate the proof of Theorem 6 to bound the optimal offline gain at a *given time step t* by $O(\max\{t^{-2/3}, p_1\})$. This leaves the remaining arguments in the proof of Theorem 7 essentially invariable since $\sum_{t=1}^{N}\frac{1}{t^{2/3}} = O(N^{1/3})$. $\square$

We will show that this "soft-O" analogy holds also for the upper bound on the worst-case adversarial model (of Theorem 2). But what would be a natural adaptation of the simple NAIVE algorithm (Definition 1) to our new model of ephemeral frequencies? It feels that the change of the denominator t at every step for our algorithms' gain (see expression (2)) inherently calls for the well-known reservoir sampling algorithm (with a reservoir size of 1) by Vitter [19]:

Definition 4 (Algorithm RESERVOIR)**.** The randomized algorithm RESERVOIR accepts the t-th element to arrive with probability $\frac{1}{t}$. Formally, $\Pr[s_t = a_t] = \frac{1}{t}$ and $\Pr[s_t = s_{t-1}] = 1-\frac{1}{t}$.

The critical property of the RESERVOIR algorithm is that the element residing in its memory at every time step is sampled *uniformly* at random from the part of the stream already observed, i.e.

$$Pr[s_t = a_i] = \frac{1}{t} \qquad \text{for all } i = 1, 2, \dots, t.$$

Theorem 11. *The* Reservoir *algorithm is* $\sqrt{N/\ln(N+1)}$*-competitive for the online* ephemeral *frequent items problem (in the worst-case adversarial model).*

Proof. Fix a worst-case input stream A for the Reservoir algorithm, and let f_t^* be the highest (ephemeral) frequency of an element of the stream up to time point t, i.e. $f_t^* = \max_{1\le i\le t} f_t(a_i)$. Then, the total optimal offline gain cannot exceed $\sum_{t=1}^N f_t^*$. Next, notice that at a given time step t the Reservoir algorithm has in its memory an element of ephemeral frequency f_t^* with probability f_t^* and, also, we have a trivial lower bound of $\frac{1}{t}$ on the gain at this step. This means that the total expected gain of the Reservoir algorithm is lower bounded by both $\sum_{t=1}^N f_t^{*2}$ and $\sum_{t=1}^N \frac{1}{t} \ge \ln(N+1)$.

Now, we have the following two cases for $\sum_{t=1}^N f_t^*$: if $\sum_{t=1}^N f_t^* \le \sqrt{N\ln(N+1)}$, then the competitive ratio is indeed bounded by

$$\frac{\sum_{t=1}^N f_t^*}{\ln(N+1)} \le \frac{\sqrt{N\ln(N+1)}}{\ln(N+1)} = \sqrt{\frac{N}{\ln(N+1)}}.$$

Otherwise, if $\sum_{t=1}^N f_t^* > \sqrt{N\ln(N+1)}$, we can use Jensen's inequality to get the bound

$$\frac{\sum_{t=1}^N f_t^*}{\sum_{t=1}^N f_t^{*2}} \le \frac{\sum_{t=1}^N f_t^*}{\frac{1}{N}\left(\sum_{t=1}^N f_t^*\right)^2} = \frac{N}{\sum_{t=1}^N f_t^*} < \frac{N}{\sqrt{N\ln(N+1)}} = \sqrt{\frac{N}{\ln(N+1)}}.$$

□

However, we have not been able to show a tight lower bound of $\tilde{\Omega}(\sqrt{N/k})$ for this ephemeral frequencies model, similar to Theorem 8 and we think this is a challenging open problem even for single slot memories. Our best result is very near to that of the lower bound of the weaker distributional adversarial setting (see Theorem 10):

Theorem 12. *The competitive ratio of the online* ephemeral *frequent items problem is* $\Omega(\sqrt[3]{N/\ln^2 N})$ *(in the worst-case adversarial model and* $k=1$*).*

Proof. We use Yao's principle, and we construct a stream of length N, consisting of $m = \sqrt[3]{N\ln N}$ consecutive "boxes", each one of length $l = \frac{N}{m}$. Pick some "good" element α (unknown to the algorithm) and throw one instance of it in every such box, uniformly at random within it. Fill the remaining positions of the stream using $(N-m)$ distinct "dummy" elements. Under this input, at every time step t belonging in the i-th box (i.e. $(i-1)\cdot\ell+1 \le t \le i\cdot\ell$) element α has an (ephemeral) frequency of

$$\frac{m}{N}\left(1-\frac{1}{i}\right) = \frac{i-1}{i\ell} \le f_t(\alpha) \le \frac{i}{(i-1)\ell+1} = \frac{m}{N}\left(1+\frac{1}{i-1}\right),$$

when $i \ge 2$ and $0 \le f_t(\alpha) \le 1$ for $i=1$. Every other "dummy" element would have a frequency of $1/t$.

An offline algorithm would discover element α until the end of the first box and place it in its memory until the end of the execution, for a total expected gain of at least $\sum_{i=2}^m \frac{m}{N}\left(1-\frac{1}{i}\right)\ell \ge m-2-\ln m = \Omega(\sqrt[3]{N\ln N})$. On the other hand, every online algorithm will either try to "discover" at every box i the desired element α, an event that happens with probability at most $1/\ell$, so as to keep it until the end for an (optimal) gain of at most $\sum_{j=i}^m \frac{m}{N}\left(1+\frac{1}{j-1}\right)\ell \le m-i+2+\ln m$ (for $i=1$ this bound is $m+\ln m+\ell$), or just accept naively every element to arrive for an expected gain of at most $\sum_{t=1}^N \frac{1}{t}$ (gain from the "dummy" elements) plus $1+\sum_{i=2}^m \frac{m}{N}\left(1+\frac{1}{i-1}\right)$ (gain from the occurrences of α). So, in the first case the total expected online gain is at most $\frac{1}{\ell}(m+\ln m+\ell) + \sum_{i=2}^m \frac{1}{\ell}(m-i+2+\ln m) = O(m^2/\ell) = O(m^3/N) = O(\ln N)$, while in the second is $O(m^2/N) = o(\ln N)$, proving the desired lower bound on the competitive ratio. □

In the light of Theorem 5 it is clear that algorithms which do not consult their current memory state S_t are not able to break the $\sqrt{N}$-bound. So, this is also the case for algorithm Reservoir (see Definition 4) and in general one can show that every algorithm that performs a *uniform* adaptive sampling of the stream throughout the execution, i.e. an element resides in its memory S_t with probability proportional to its frequency on the part of the stream so far, is doomed to this $\sqrt{N}$-performance. So what is the underlying idea that makes EAGER break this boundary and achieve optimal $\sqrt[3]{N}$-performance (see Theorem 7)? Intuitively, it seems to us that this algorithm manages to implicitly force a *non-uniform* adaptive sampling proportional to the *squares* p_i^2 of the probabilities of the underlying distribution. We believe that studying the implementability of different samplings based on various functions of the underlying probabilities (other than the simple uniform one) in such an adaptive way, with fixed size memory k, may be a challenging and very interesting problem of its own interest in the wider area of Data Streams.

Another interesting direction that the adoption of ephemeral frequencies gives us, is that of considering a measure of importance upon our time horizon, e.g. in many scenarios we are more interested in the recent past rather than remote observations. This can be modeled by using a non-decreasing, non-negative real valued *aging sequence* $q_0, q_1, q_2, \ldots$ in order to, for a given time point t, assign to past element a_{t-i} a weight of q_i, where $i = 0, 1, \ldots, t-1$. From this point of view, we need to generalize our (ephemeral) frequency definition in (2) to

$$f_t^A(a) = \frac{\sum_{i:\, a_{t-i}=a} q_i}{\sum_{i=0}^{t-1} q_i},$$

to capture this notion of time decay. Examples of such aging sequences may include an exponentially decreasing time model, where $q_i = e^{-\lambda \cdot i}$ for some real constant $\lambda > 0$, or a sliding window model of window size w, where $q_i = 1$ if $i \leq w - 1$ and $q_i = 0$ otherwise. Notice that the simple streaming model with no time-decaying which we have considered in this paper corresponds to the case of $q_i = 1$ for all i.

Finally, as stated in the introduction, to our knowledge this is just the second work that studies some data streaming problem in a competitive analysis framework, after only the Aggregate-Max paper of Becchetti and Koutsoupias [5]. We think that a lot of interesting work can further be done in this direction and that the intrinsic online nature of Data Stream settings makes competitive analysis an appropriate tool to approach such problems, especially when considering a decision-making under uncertainty and/or an economics perspective.

References

[1] Aggarwal, C. *Data Streams: Models and Algorithms*. Advances in Database Systems. Springer, 2007.

[2] Alon, N., Matias, Y., and Szegedy, M. The Space Complexity of Approximating the Frequency Moments. *Journal of Computer and System Sciences 58*, 1 (1999), 137–147.

[3] Babaioff, M., Immorlica, N., Kempe, D., and Kleinberg, R. Online auctions and generalized secretary problems. *ACM SIGecom Exchanges 7*, 2 (2008), 1–11.

[4] Becchetti, L., Chatzigiannakis, I., and Giannakopoulos, Y. Streaming techniques and data aggregation in networks of tiny artefacts. *Computer Science Review 5*, 1 (February 2011), 27–46.

[5] Becchetti, L., and Koutsoupias, E. Competitive analysis of aggregate max in windowed streaming. In *Proceedings of ICALP '09* (2009), vol. I, Springer, pp. 156–170.

[6] Borodin, A., and El-Yaniv, R. *Online Computation and Competitive Analysis*. Cambridge University Press, 1998.

[7] Boyer, R., and Moore, J. MJRTY-A Fast Majority Vote Algorithm. Tech. rep., Texas University at Austin, Institute for Computing Science and Computer Applications, 1981.

[8] Cormode, G., and Hadjieleftheriou, M. Finding frequent items in data streams. *Proceedings of the VLDB Endowment 1*, 2 (2008), 1530–1541.

[9] Datar, M., Gionis, A., Indyk, P., and Motwani, R. Maintaining stream statistics over sliding windows. *SIAM Journal on Computing* (2002), 635–644.

[10] Ferguson, T. S. Who solved the secretary problem? *Statistical Science 4*, 3 (1989), 282–296.

[11] Fiat, A., and Woeginger, G., Eds. *Online Algorithms: The State of the Art*. Springer, 1998.

[12] Gama, J. a., and Geber, M. M., Eds. *Learning from Data Streams: Processing Techniques in Sensor Networks*. Springer, 2007.

[13] Giannakopoulos, Y., and Koutsoupias, E. Competitive analysis of maintaining frequent items of a stream. In *Proceedings of SWAT '12* (2012), pp. 340–351.

[14] Liu, L., and Ozsu, M. T., Eds. *Encyclopedia of Database Systems*. Springer, 2009.

[15] Misra, J., and Gries, D. Finding repeated elements. *Science of Computer Programming 2*, 2 (1982), 143–152.

[16] Mitzenmacher, M., and Upfal, E. *Probability and Computing: Randomized Algorithms and Probabilistic Analysis*. Cambridge University Press, 2005.

[17] Munro, J. I., and Paterson, M. S. Selection and sorting with limited storage. In *Proceedings of FOCS '78* (1978), pp. 253–258.

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

[19] Vitter, J. S. Random sampling with a reservoir. *ACM Trans. Math. Softw. 11*, 1 (1985), 37–57.

[20] Yao, A. C.-C. Probabilistic computations: Toward a unified measure of complexity. In *Proceedings of FOCS '77* (1977), pp. 222–227.