The Frequent Items Problem in Online Streaming under Various Performance Measures[⋆]

Joan Boyar, Kim S. Larsen, and Abyayananda Maiti

University of Southern Denmark, Odense, Denmark
{joan,kslarsen,abyaym}@imada.sdu.dk

Abstract. In this paper, we strengthen the competitive analysis results obtained for a fundamental online streaming problem, the Frequent Items Problem. Additionally, we contribute with a more detailed analysis of this problem, using alternative performance measures, supplementing the insight gained from competitive analysis. The results also contribute to the general study of performance measures for online algorithms. It has long been known that competitive analysis suffers from drawbacks in certain situations, and many alternative measures have been proposed. However, more systematic comparative studies of performance measures have been initiated recently, and we continue this work, using competitive analysis, relative interval analysis, and relative worst order analysis on the Frequent Items Problem.

1 Introduction

The analysis of problems and algorithms for streaming applications, treating them as online problems, was started in [2]. In online streaming, the items must be processed one at a time by the algorithm, making some irrevocable decision with each item. A fixed amount of resources is assumed. In the frequent items problem [12], an algorithm must store an item, or more generally a number of items, in a buffer, and the objective is to store the items appearing most frequently in the entire stream. This problem has been studied in [15]. In addition to probabilistic considerations, they analyzed deterministic algorithms using competitive analysis. We analyze the frequent items problem using relative interval analysis [14] and relative worst order analysis [4]. In addition, we tighten the competitive analysis [17, 16] results from [15].

It has been known since the start of the area that competitive analysis does not always give good results [17] and many alternatives have been proposed. However, as a general rule, these alternatives have been fairly problem specific and most have only been compared to competitive analysis. A more comprehensive study of a larger number of performance measures on the same problem

[⋆] Supported in part by the Danish Council for Independent Research. Part of this work was done while the authors were visitng the University of Waterloo.

scenarios was initiated in [8] and this line of work has been continued in [9, 6, 7]. With this in mind, we would like to produce complete and tight results, and for that reason, we focus on a fairly simple combinatorial problem and on simple algorithms for its solution, incorporating greediness and adaptability trade-offs to a varying extent.

Finally, we formalize a notion of competitive function, as opposed to competitive ratio, in a manner which allows us to focus on the constant in front of the high order term. These ideas are also used to generalize relative worst order analysis.

Most proofs have been omitted due to space restrictions. These can be found in the full version of the paper [10].

2 Preliminaries

This is a streaming problem, but as usual in online algorithms we use the term sequence or input sequence to refer to a stream. We denote an *input sequence* by $I = a_1, a_2, \ldots, a_n$, where the items a_i are from some universe $\mathcal{U}$, assumed to be much larger than n. We may refer to the index also as the *time step*. We consider online algorithms, which means that items are given one by one.

We consider the simplest possible frequent items problem: An algorithm has a *buffer* with space for one item. When processing an item, the algorithm can either discard the item or replace the item in the buffer by the item being processed. The objective is to keep the most frequently occurring items in the buffer, where frequency is measured over the entire input, i.e., when an algorithm must make a decision, the quality of the decision also depends on items not yet revealed to the algorithm. We define this objective function formally:

Given an online algorithm $\mathcal{A}$ for this problem, we let $s_t^{\mathcal{A}}$ denote *the item in the buffer at time step t*. We may omit the superscript when it is clear from the context which algorithm we discuss.

Given an input sequence I and an item $a \in \mathcal{U}$, the *frequency* of the item is defined as $f_I(a) = \frac{n_I(a)}{n}$, where $n_I(a) = |\{i \mid a_i = a\}|$ is the number of occurrences of a in I. The objective is to maximize the *aggregate frequency* [15], defined by $F_{\mathcal{A}}(I) = \sum_{t=1}^{n} f_I(s_t^{\mathcal{A}})$, i.e., the sum of the frequencies of the items stored in the buffer over the time.

We compare the quality of the achieved aggregate frequencies of three different deterministic online algorithms from [15]: the naive algorithm (Nai), the eager algorithm (Eag), and the majority algorithm (Maj). All three are practical streaming algorithms, being simple and using very little extra space.

Definition 1. [Nai] Nai *buffers every item as it arrives, i.e., $s_t^{\text{Nai}} = a_t$ for all $t = 1, 2, \ldots, n$.*

The algorithm Eag switches mode upon detecting a *repeated item*, an item which occurs in two consecutive time steps.

Definition 2. [Eag] *Initially,* Eag *buffers every item as it arrives. If it finds a repeated item, then it keeps that item until the end, i.e., let* $t^* = \min_{1 \le t \le n-1}\{t \mid a_t = a_{t+1}\}$*, if such a* t *exists, and otherwise* $t^* = n$*. Then* Eag *is the algorithm with* $s_t^{\text{Eag}} = a_t$ *for all* $t \le t^*$ *and* $s_t^{\text{Eag}} = a_{t^*}$ *for all* $t > t^*$*.*

The algorithm Maj keeps a counter along with the buffer. Initially, the counter is set to zero.

Definition 3. [Maj] *If the counter is zero, then* Maj *buffers the arriving item and sets the counter to one. Otherwise, if the arriving item is the same as the one currently buffered,* Maj *increments the counter by one, and otherwise decrements it by one.*

Finally, as usual in online algorithms, we let Opt denote an optimal offline algorithm. Opt is, among other things, used in competitive analysis as a reference point, since no online algorithm can do better. If $\mathcal{A}$ is an algorithm, we let $\mathcal{A}(I)$ denote the result of the algorithm, i.e., $\mathcal{A}(I) = F_{\mathcal{A}}(I)$.

In comparing these three algorithms, we repeatedly use the same two families of sequences; one where Eag performs particularly poorly and one where Maj performs particularly poorly.

Definition 4. *We define the sequences*

$$E_n = a, a, b, b, \ldots, b,$$

where there are $n-2$ *copies of* b*, and*

$$W_n = \begin{cases} a_1, a_0, a_2, a_0, \ldots, a_{\frac{n}{2}}, a_0 & \text{for even } n \\ a_1, a_0, a_2, a_0, \ldots, a_{\lfloor \frac{n}{2} \rfloor}, a_0, a_{\lceil \frac{n}{2} \rceil} & \text{for odd } n. \end{cases}$$

The four algorithms, including Opt, obtain the aggregate frequencies below on these two families of sequences. The arguments are simple, but fundamental, and also serve as an introduction to the algorithmic behavior of these algorithms.

Proposition 1. *The algorithms' results on* E_n *and* W_n *are as in Fig. 1.*

Proof. In E_n, the frequency of a is $\frac{2}{n}$ and the frequency of b is $\frac{n-2}{n}$. Thus $\text{Nai}(E_n) = 2\frac{2}{n} + (n-2)\frac{n-2}{n} = n - 4 + \frac{8}{n}$. In W_n, the frequency of a_0 is $\lfloor \frac{n}{2} \rfloor / n$, and the frequencies of all the other a_i, $1 \le i \le \lceil \frac{n}{2} \rceil$, are $\frac{1}{n}$. Thus, $\text{Nai}(W_n) = \lceil \frac{n}{2} \rceil \frac{1}{n} + \lfloor \frac{n}{2} \rfloor \frac{\lfloor \frac{n}{2} \rfloor}{n}$. Considering both even and odd n gives the required result.

When processing E_n, Eag keeps a in its buffer. Hence, $\text{Eag}(E_n) = n\frac{2}{n} = 2$. Since W_n has no repeated item, $\text{Eag}(W_n) = \text{Nai}(W_n)$.

For E_n, Maj will have a in its buffer for the first four time steps, so $\text{Maj}(E_n)$ is $4\frac{2}{n} + (n-4)\frac{n-2}{n} = n - 6 + \frac{16}{n}$. For W_n, Maj brings each a_i, $1 \le i \le n$, into its buffer and never brings a_0 into its buffer. Thus, $\text{Maj}(W_n) = n\frac{1}{n} = 1$.

With E_n, Opt is forced to perform the same as Nai. In W_n, Opt must buffer a_1 in the first time step, but it buffers a_0 for the remainder of the sequence. Thus, $\text{Opt}(W_n) = \frac{1}{n} + (n-1)\frac{\lfloor \frac{n}{2} \rfloor}{n}$. Considering both even and odd n gives the required result. □

	E_n	W_n
Nai	$n-4+\frac{8}{n}$	$\begin{cases} \frac{n}{4}+\frac{1}{2} & \text{for even } n \\ \frac{n}{4}+\frac{3}{4n} & \text{for odd } n \end{cases}$
Eag	2	as Nai
Maj	$n-6+\frac{16}{n}$	1
Opt	as Nai	$\begin{cases} \frac{n}{2}-\frac{1}{2}+\frac{1}{n} & \text{for even } n \\ \frac{n}{2}-1+\frac{3}{2n} & \text{for odd } n \end{cases}$

Fig. 1. The algorithms' aggregate frequencies on E_n and W_n.

Definition 5. *Let $\mathcal{A}$ be any online algorithm. We denote the worst aggregate frequency of $\mathcal{A}$ over all the permutations σ of I by $\mathcal{A}_W(I) = \min_\sigma \mathcal{A}(\sigma(I))$.*

It is convenient to be able to consider items in order of their frequencies. Let $D(I) = a'_1, a'_2, \ldots, a'_n$ be a sorted list of the item in I in nondecreasing order of frequencies. For example, if $I = a, b, c, a, b, a$, then $D(I) = c, b, b, a, a, a$. We will use the notation $D(I)$ throughout the paper.

Lemma 1. *For odd n, $\text{Maj}_W(I) = 2\sum_{i=1}^{\lfloor \frac{n}{2} \rfloor} f_I(a'_i) + f_I(a'_{\lceil \frac{n}{2} \rceil})$, and for even n, $\text{Maj}_W(I) = 2\sum_{i=1}^{\frac{n}{2}} f_I(a'_i)$, where the a'_i are the items of $D(I)$.*

Proof. Every time step where the counter is decremented can be paired with an earlier one where it is incremented and the same item is in the buffer. So, at least $\lceil \frac{n}{2} \rceil$ requests contribute to the aggregate frequency of the algorithm. One can order the items so that exactly the $\lceil \frac{n}{2} \rceil$ requests to that many least frequent items are buffered as follows: Assuming n is even, then the worst permutation is $a'_1, a'_n, a'_2, a'_{n-1}, \ldots a'_{\frac{n}{2}}, a'_{\frac{n}{2}+1}$. All (but the last request when n is odd) of the requests which lead to an item entering the buffer contribute twice, since they are also in the buffer for the next step. $\square$

3 Competitive Analysis

An online streaming problem was first studied from an online algorithms perspective using competitive analysis by Becchetti and Koutsoupias [2]. Competitive analysis[17, 16] evaluates an online algorithm in comparison to an optimal offline algorithm. For a maximization problem, an algorithm, $\mathcal{A}$ is called c-competitive, for some constant c, if there exists a constant α such that for all finite input sequences I, $\text{Opt}(I) \leq c \cdot \mathcal{A}(I) + \alpha$. The competitive ratio of $\mathcal{A}$ is the infimum over all c such that $\mathcal{A}$ is c-competitive. Since, for the online frequent items problem, the relative performance of algorithms depends on the length of I, we define a modified and more general version of competitive analysis, providing a formal basis for our own claims as well as claims made in earlier related work. Functions have also been considered in [13]. Here, we focus on the constant in front of the most significant term. Our definition can be adapted easily to minimization problems in the same way that the adaptations are handled for standard competitive analysis. In all these definitions, when n is not otherwise defined, we use it to denote $|I|$, the length of the sequence I. As usual, when using asymptotic notation in inequalities, notation such as $f(n) \leq g(n)+o(g(n))$ means that there exists a function $h(n) \in o(g(n))$ such that $f(n) \leq g(n)+h(n)$. Thus, we focus on the multiplicative factors that relate the online algorithm's result to the input length.

Definition 6. *An algorithm $\mathcal{A}$ is $f(n)$-*competitive *if*

$$\forall I\colon\ \textsc{Opt}(I) \leq (f(n)+o(f(n))) \cdot \mathcal{A}(I).$$

$\mathcal{A}$ has competitive function $f(n)$ *if $\mathcal{A}$ is $f(n)$-*competitive *and for any $g(n)$ such that $\mathcal{A}$ is $g(n)$-*competitive*, $\lim_{n\to\infty} \frac{f(n)}{g(n)} \leq 1$.*

If algorithm $\mathcal{A}$ has competitive function $f(n)$ *and algorithm $\mathcal{B}$ has* competitive function $f'(n)$*, then $\mathcal{A}$ is better than $\mathcal{B}$ according to competitive analysis if $\lim_{n\to\infty} \frac{f(n)}{f'(n)} < 1$.*

Thus, the concept of competitive function is an exact characterization up to the level of detail we focus on. It can be viewed as an equivalence relation, and if $\lim_{n\to\infty} \frac{f(n)}{g(n)} = 1$ for two functions $f(n)$ and $g(n)$, then they belong to (and are representatives of) the same equivalence class. For example, $\frac{\sqrt{n}}{2}$ and $\frac{\sqrt{n}}{2-\frac{1}{\sqrt{n}}}$ are considered equivalent, whereas $\frac{\sqrt{n}}{2}$ and $\frac{\sqrt{n}}{4}$ are not.

All three algorithms discussed here are non-competitive according to the original definition. However, information regarding the relative quality of these algorithms can be obtained by considering the most significant constants from the corresponding functions. Giannakopoulos et al. has proved that no randomized algorithm for the online frequent items problem, where the buffer has room for one item, can have a competitive function better than $\frac{1}{3}\sqrt{n}$ [15]. That result can be strengthened for the deterministic case, based on sequences of the form $I_n = a_1, a_2, \ldots a_{n-\sqrt{n}}, x, x, \ldots, x$, where x is a_1 or a_2.

Theorem 1. *No deterministic algorithm for the online frequent items problem can have a competitive function better than $\frac{\sqrt{n}}{2}$.*

In [15], Giannakopoulos et al. proved that for all sequences I of length n, $\textsc{Opt}(I) \leq \sqrt{n} \cdot \textsc{Nai}(I)$. Here we give a tighter result for $\textsc{Nai}$.

Theorem 2. $\textsc{Nai}$ *has competitive function $\frac{\sqrt{n}}{2}$. It is an optimal deterministic online algorithm for the frequent items problem.*

For $\textsc{Maj}$ Giannakopoulos et al. [15] proved a competitive ratio of $\Theta(n)$. We give the asymptotically tight bounds, including the multiplicative factor.

Theorem 3. $\textsc{Maj}$ *has competitive function $\frac{n}{2}$.*

Proof. For the lower bound, consider the family of sequences, W_n, from Definition 4. By Proposition 1, $\textsc{Maj}(W_n) = 1$, and

$$\textsc{Opt}(W_n) = \begin{cases} \frac{n}{2} - \frac{1}{2} + \frac{1}{n} & \text{for even } n \\ \frac{n}{2} - 1 + \frac{3}{2n} & \text{for odd } n \end{cases}$$

Consequently, $\textsc{Opt}(W_n) \geq \frac{n}{2}\textsc{Maj}(W_n) - 1$. Thus, the competitive function cannot be better than $\frac{n}{2}$.

For the upper bound, let f be the largest frequency of any item in some input sequence I of length n. Opt cannot have an aggregate frequency larger than nf.

If $f \leq \frac{1}{2}$, then, since no algorithm can have an aggregate frequency less than one in total, $\frac{\textsc{Opt}(I)}{\textsc{Maj}(I)} \leq nf \leq \frac{n}{2}$.

It remains to consider the range $\frac{1}{2} < f \leq 1$. Let a_0 denote the most frequent item in I. Note that a_0 must be in the buffer at some point since $f > \frac{1}{2}$.

Since there are $n - fn$ items different from a_0, the total length of all subsequences where a_0 is not in the buffer is at most $2(n - fn)$. This means that a_0 *is* in the buffer at least $n - 2(n - fn) = 2fn - n$ times, collecting at least $(2fn-n)f = 2nf^2-nf$. The remaining items collect at least $2(n-fn)\frac{1}{n}$. In total, this amounts to $2nf^2-nf+2-2f$. If we can prove that this quantity is at least $2f$ for large n, then asymptotically, $\frac{\textsc{Opt}(I)}{\textsc{Maj}(I)} \leq \frac{nf}{2nf^2-nf+2-2f} \leq \frac{nf}{2f} = \frac{n}{2}$ and we will be done. Now, $2nf^2 - nf + 2 - 2f \geq 2f$ if and only if $2nf^2 - (n+4)f + 2 \geq 0$. Taking the derivative of the left side shows that the left side is an increasing function of f for $n \geq 4$ and $f \geq \frac{1}{2}$. Thus, $\textsc{Opt}(I) \leq \frac{n}{2}\textsc{Maj}(I)$ holds for all f and all $n \geq 4$. This implies that Maj is $\frac{n}{2}$-competitive and, in total, that the competitive function of Maj is $\frac{n}{2}$. □

Theorem 4. *The competitive function of the algorithm* Eag *is* $\frac{n}{2}$.

4 Relative Interval Analysis

Dorrigiv et al. [14] proposed another analysis method, relative interval analysis, in the context of paging. Relative interval analysis compares two online algorithms directly, i.e., it does not use the optimal offline algorithm as the baseline of the comparison. It compares two algorithms on the basis of the rate of the outcomes over the length of the input sequence rather than their worst case behavior. Here we define this analysis for maximization problems for two algorithms $\mathcal{A}$ and $\mathcal{B}$, following [14].

Definition 7. *Define*

$$\mathit{Min}_{\mathcal{A},\mathcal{B}}(n) = \min_{|I|=n} \{\mathcal{A}(I) - \mathcal{B}(I)\} \text{ and } \mathit{Max}_{\mathcal{A},\mathcal{B}}(n) = \max_{|I|=n} \{\mathcal{A}(I) - \mathcal{B}(I)\},$$

$$\mathit{Min}(\mathcal{A},\mathcal{B}) = \liminf_{n\to\infty} \frac{\mathit{Min}_{\mathcal{A},\mathcal{B}}(n)}{n} \text{ and } \mathit{Max}(\mathcal{A},\mathcal{B}) = \limsup_{n\to\infty} \frac{\mathit{Max}_{\mathcal{A},\mathcal{B}}(n)}{n}. \quad (1)$$

The relative interval *of $\mathcal{A}$ and $\mathcal{B}$ is defined as $l(\mathcal{A},\mathcal{B}) = [\mathit{Min}(\mathcal{A},\mathcal{B}), \mathit{Max}(\mathcal{A},\mathcal{B})]$. If $\mathit{Max}(\mathcal{A},\mathcal{B}) > |\mathit{Min}(\mathcal{A},\mathcal{B})|$, then $\mathcal{A}$ is said to have* better performance *than $\mathcal{B}$ in this model.*

Note that $\mathrm{Min}(\mathcal{A},\mathcal{B}) = -\mathrm{Max}(\mathcal{B},\mathcal{A})$ and $\mathrm{Max}(\mathcal{A},\mathcal{B}) = -\mathrm{Min}(\mathcal{B},\mathcal{A})$.

For any pair of algorithms, $\mathcal{A}$ and $\mathcal{B}$, for the frequent items problem, there is a trivial upper bound on $\mathrm{Max}(\mathcal{A},\mathcal{B})$ and lower bound on $\mathrm{Min}(\mathcal{A},\mathcal{B})$.

Proposition 2. *For any pair of algorithms $\mathcal{A}$ and $\mathcal{B}$, $Max(\mathcal{A},\mathcal{B}) \leq 1$ and $Min(\mathcal{A},\mathcal{B}) \geq -1$.*

Proof. The maximum aggregate frequency any algorithm could have is for a sequence where all items are identical, giving the value n. The minimum is for a sequence where all items are different, giving the value 1. The required bounds follow since $\limsup_{n\to\infty} \frac{n-1}{n} = 1$. □

4.1 Naive vs. Eager

According to relative interval analysis, Nai has better performance than Eag.

Theorem 5. *According to relative interval analysis $l(\textsc{Nai}, \textsc{Eag}) = [-\frac{1}{4}, 1]$.*

4.2 Naive vs. Majority

Nai and Maj are equally good according to relative interval analysis.

Theorem 6. *According to relative interval analysis $l(\textsc{Nai}, \textsc{Maj}) = [-\frac{1}{4}, \frac{1}{4}]$.*

Proof. For the maximum value of $\textsc{Nai}(I) - \textsc{Maj}(I)$, it is sufficient to consider the worst permutation of I for Maj since Nai has the same output for all permutations of I. For the worst permutation, $\textsc{Maj}_W(I)$ will buffer only the first $\lceil \frac{n}{2} \rceil$ items of the distribution $D(I)$. The first $\lfloor \frac{n}{2} \rfloor$ items will be buffered twice and in case of odd n, the $\lceil \frac{n}{2} \rceil$th item will be stored once at the last time step. Let $D(I) = a'_1, a'_2, a'_3, \ldots, a'_n$. Then

$$\begin{aligned}\textsc{Nai}(I) - \textsc{Maj}_W(I) &= \sum_{i=1}^{n} f_I(a'_i) - 2\sum_{i=1}^{\lfloor \frac{n}{2} \rfloor} f_I(a'_i) - \left(\left\lceil \frac{n}{2} \right\rceil - \left\lfloor \frac{n}{2} \right\rfloor\right) f_I(a'_{\lceil \frac{n}{2} \rceil}) \\ &= \sum_{i=\lceil \frac{n+2}{2} \rceil}^{n} f_I(a'_i) - \sum_{i=1}^{\lfloor \frac{n}{2} \rfloor} f_I(a'_i). \qquad (2)\end{aligned}$$

Let p be the number of occurrences of the most frequent item in I. Then $\textsc{Nai}(I) - \textsc{Maj}_W(I)$ equals

$$\sum_{i=\lceil \frac{n+2}{2} \rceil}^{n} f_I(a'_i) - \sum_{i=1}^{\lfloor \frac{n}{2} \rfloor} f_I(a'_i) \leq \left\lfloor \frac{n}{2} \right\rfloor \frac{p}{n} - \left(p - \left\lceil \frac{n}{2} \right\rceil\right) \frac{p}{n} = p - \frac{p^2}{n}.$$

If n is even, an upper bound on the maximum difference will be achieved when $p = \frac{n}{2}$, and for odd n when $p = \frac{n+1}{2}$. This gives an upper bound on the maximum of $\textsc{Nai}(I) - \textsc{Maj}(I)$ of $\frac{n}{4}$ for even n and $\frac{n}{4} - \frac{1}{4n}$ for odd n. For a lower bound on the maximum value of $\text{NAI}(I) - \text{MAJ}(I)$, we consider the family of sequences, W_n, from Definition 4. By Proposition 1, for even n, $\text{NAI}(W_n) - \text{MAJ}(W_n) = \frac{n}{4} - \frac{1}{2}$, and for odd n, $\text{NAI}(W_n) - \text{MAJ}(W_n) = \frac{n}{4} - 1 + \frac{1}{4n}$. Thus, $\text{Max}(\text{NAI}, \text{MAJ}) \geq \limsup_{n\to\infty} \frac{\text{NAI}(W_n) - \text{MAJ}(W_n)}{n} = \frac{1}{4}$, matching the upper bound.

To derive the minimum value of $\text{NAI}(I) - \text{MAJ}(I)$, we calculate the maximum value of $\text{MAJ}(I) - \text{NAI}(I)$. For an upper bound on this, we consider the best permutation, I_B, for MAJ of an arbitrary sequence, I. For I_B, MAJ would buffer the half of the requests in the sequence with the highest frequencies. The difference, $\text{MAJ}(I_B) - \text{NAI}(I_B)$, is

$$\begin{aligned}
&2 \sum_{i=\lceil \frac{n+2}{2} \rceil}^{n} f_I(a'_i) + \left(\left\lceil \frac{n}{2} \right\rceil - \left\lfloor \frac{n}{2} \right\rfloor \right) f_I(a'_{\lceil \frac{n}{2} \rceil}) - \sum_{i=1}^{n} f_I(a'_i) \\
= &\sum_{i=\lceil \frac{n+2}{2} \rceil}^{n} f_I(a'_i) - \sum_{i=1}^{\lfloor \frac{n}{2} \rfloor} f_I(a'_i).
\end{aligned}$$

This expression is exactly the same as the expression for $\text{NAI}(I) - \text{MAJ}_W(I)$ from Eq. 2, so we get the same upper bound of $\frac{1}{4}$. Now, for a lower bound on $\text{Max}(\text{MAJ}, \text{NAI})$, we use the family of sequences, I_n defined as

$$I_n = a_0, a_0, \ldots, a_0, a_1, a_2, \ldots, a_{\lfloor \frac{n}{2} \rfloor},$$

where there are $\lceil \frac{n}{2} \rceil$ copies of a_0. Then

$$\text{NAI}(I_n) = \left\lfloor \frac{n}{2} \right\rfloor \frac{1}{n} + \left\lceil \frac{n}{2} \right\rceil \frac{\lceil \frac{n}{2} \rceil}{n} = \begin{cases} \frac{n}{4} + \frac{1}{2} & \text{for even } n \\ \frac{n}{4} + 1 + \frac{1}{4n} & \text{for odd } n \end{cases}$$

and $\text{MAJ}(I_n) = n \frac{\lceil \frac{n}{2} \rceil}{n} = \lceil \frac{n}{2} \rceil$. I_n gives a lower bound of $\frac{1}{4}$ on $\text{Max}(\text{MAJ}, \text{NAI})$, since $\text{MAJ}(I_n) - \text{NAI}(I_n) \geq \lceil \frac{n}{2} \rceil - \frac{n}{4} - 1 - \frac{1}{4n}$. It follows that, $\text{Min}(\text{NAI}, \text{MAJ}) = -\text{Max}(\text{MAJ}, \text{NAI}) = -\frac{1}{4}$, and $l(\text{NAI}, \text{MAJ}) = [-\frac{1}{4}, \frac{1}{4}]$. □

4.3 Majority vs. Eager

According to relative interval analysis, MAJ has better performance than EAG.

Theorem 7. *According to relative interval analysis* $l(\text{MAJ}, \text{EAG}) = [-\frac{1}{2}, 1]$.

5 Relative Worst Order Analysis

Relative worst order analysis [4] compares two online algorithms directly. It compares two algorithms on their worst orderings of sequences which have the same content, but possibly different order. The definition of this measure is somewhat more involved; see [5] for more intuition on the various elements. As in the case of competitive analysis, here too the relative performance of the algorithms depend on the length of the input sequence I. As in Section 3, we define a modified and more general version of relative worst order analysis. The definition is given for a maximization problem, but trivially adaptable to be used for minimization problems as well; only the decision as to when which algorithm is better would change.

The following definition is parameterized by a total ordering, $\sqsubseteq$, since we will later use it for both $\leq$ and $\geq$.

Definition 8. *f is a $(\mathcal{A}, \mathcal{B}, \sqsubseteq)$*-function *if*

$$\forall I\colon\ \mathcal{A}_W(I) \sqsubseteq (f(n) + o(f(n))) \cdot \mathcal{B}_W(I),$$

where $\mathcal{A}$ and $\mathcal{B}$ are algorithms and $\sqsubseteq$ is a total ordering. Recall from Definition 5 that the notation $\textsc{Alg}_W(I)$*, where* $\textsc{Alg}$ *is some algorithm, denotes the result of* $\textsc{Alg}$ *on its worst permutation of I.*

f is a bounding function with respect to $(\mathcal{A}, \mathcal{B}, \sqsubseteq)$ *if f is a $(\mathcal{A}, \mathcal{B}, \sqsubseteq)$-function and for any $(\mathcal{A}, \mathcal{B}, \sqsubseteq)$-function g,* $\lim_{n\to\infty} \frac{f(n)}{g(n)} \sqsubseteq 1$*.*

If f is a bounding function with respect to $(\mathcal{A}, \mathcal{B}, \leq)$ and g is a bounding function with respect to $(\mathcal{A}, \mathcal{B}, \geq)$, then $\mathcal{A}$ and $\mathcal{B}$ are said to be comparable *if* $\lim_{n\to\infty} f(n) \leq 1$ *or* $\lim_{n\to\infty} g(n) \geq 1$*.*

If $\lim_{n\to\infty} f(n) \leq 1$*, then $\mathcal{B}$ is better than $\mathcal{A}$ and $g(n)$ is a* relative worst order function of $\mathcal{A}$ and $\mathcal{B}$*, and if* $\lim_{n\to\infty} g(n) \geq 1$*, then $\mathcal{A}$ is better than $\mathcal{B}$ and $f(n)$ is a* relative worst order function of $\mathcal{A}$ and $\mathcal{B}$*.*

We use $\mathrm{WR}_{\mathcal{A},\mathcal{B}} = f(n)$ to indicate that $f(n)$ belongs to the equivalence class of *relative worst order functions of $\mathcal{A}$ and $\mathcal{B}$*.

The competitive function could also have been defined using this framework, but was defined separately as a gentle introduction to the idea.

5.1 Naive vs. Optimal

Relative worst order analysis can show the strength of the simple, but adaptive, $\textsc{Nai}$ algorithm by comparing it with the powerful $\textsc{Opt}$. $\textsc{Nai}$ is an optimal algorithm according to relative worst order analysis, in the sense that it is equivalent to $\textsc{Opt}$.

Theorem 8. *According to relative worst order analysis* $\mathit{WR}_{\textsc{Opt},\textsc{Nai}} = 1$*, so* $\textsc{Nai}$ *and* $\textsc{Opt}$ *are equivalent.*

Proof. In the aggregate frequency problem, even though $\textsc{Opt}$ knows the whole sequence in advance, it cannot store an item before it first appears in the sequence. Thus, for any input sequence I, the worst permutation for $\textsc{Opt}$ is the sorting of I according to the increasing order of the frequencies of the items, i.e., $D(I)$. On this ordering, $\textsc{Opt}$ is forced to behave like $\textsc{Nai}$. Therefore, the constant function 1 is a bounding function with respect to both $(\textsc{Opt}, \textsc{Nai}, \leq)$ and $(\textsc{Opt}, \textsc{Nai}, \geq)$, so $\mathrm{WR}_{\textsc{Opt},\textsc{Nai}} = 1$. □

5.2 Naive vs. Eager

According to relative worst order analysis, $\textsc{Nai}$ is better than $\textsc{Eag}$.

Theorem 9. *According to relative worst order analysis* $WR_{\textsc{Nai},\textsc{Eag}} = \frac{n}{2}$.

Proof. From Theorem 8, we know that for $\textsc{Opt}$'s worst permutation, I_W, of any sequence I, $\textsc{Opt}(I_W) = \textsc{Nai}(I_W)$. Any arbitrary online algorithm $\mathcal{A}$ cannot be better than $\textsc{Opt}$ on any sequence, so $\textsc{Nai}$ and $\mathcal{A}$ are comparable. For any arbitrary online algorithm $\mathcal{A}$ and a worst order, I_W, for $\mathcal{A}$ of any sequence I, $\frac{\textsc{Nai}(I_W)}{\mathcal{A}(I_W)} = \frac{\textsc{Opt}(I_W)}{\mathcal{A}(I_W)}$, so a competitive function for $\mathcal{A}$ is an upper bound on the relative worst order function of $\mathcal{A}$ and $\mathcal{B}$. By Theorem 4, $\mathrm{WR}(\textsc{Nai}, \textsc{Eag}) \leq \frac{n}{2}$. Consider the family of sequences, E_n, from Definition 4. These sequences are in the worst ordering for both $\textsc{Eag}$ and $\textsc{Opt}$. By Proposition 1, $\textsc{Nai}(E_n) = n-4+\frac{8}{n}$ and $\textsc{Eag}(E_n) = 2$. Thus, $\textsc{Nai}(E_n) = \frac{n}{2}\textsc{Eag}(E_n) - 4 + \frac{8}{n}$. Consequently, $\frac{n}{2}$ is a relative worst order function of $\textsc{Nai}$ and $\textsc{Eag}$, and $\mathrm{WR}_{\textsc{Nai},\textsc{Eag}} = \frac{n}{2}$. □

5.3 Naive vs. Majority

According to relative worst order analysis, $\textsc{Nai}$ is better than $\textsc{Maj}$, though not quite as much better as compared to $\textsc{Eag}$.

Theorem 10. *According to relative worst order analysis,* $WR_{\textsc{Nai},\textsc{Maj}} = \frac{n}{4}$.

5.4 Majority vs. Eager

Theorem 11. *According to relative worst order analysis,* $\textsc{Maj}$ *and* $\textsc{Eag}$ *are incomparable.*

Proof. First, we show that $\textsc{Maj}$ can be much better than $\textsc{Eag}$. Consider the family of sequences, E_n, from Definition 4. These sequences are in their worst orderings for both $\textsc{Maj}$ and $\textsc{Eag}$. By Proposition 1, $\textsc{Eag}(E_n) = 2$, so

$$\textsc{Maj}_W(E_n) = n - 6 + \frac{16}{n} \geq \left(\frac{n}{2} - 3 + \frac{8}{n}\right) \textsc{Eag}_W(E_n).$$

Now, we show that $\textsc{Eag}$ can be much better than $\textsc{Maj}$. Consider the family of sequences, W_n, from Definition 4. These sequences are in their worst orderings for $\textsc{Maj}$, so by Proposition 1, $\textsc{Maj}_W(W_n) = 1$. A worst ordering for $\textsc{Eag}$ is

$$W'_n = a_1, a_2, \ldots, a_{\lceil \frac{n}{2} \rceil}, a_0, a_0, \ldots, a_0,$$

where there are $\lfloor \frac{n}{2} \rfloor$ copies of a_0. $\textsc{Eag}(W'_n) = \textsc{Nai}(W_n)$, which by Proposition 1 is $\frac{n}{4} + \frac{1}{2}$ when n is even and $\frac{n}{4} + \frac{3}{4n}$ when n is odd. Thus,

$$\textsc{Eag}_W(W_n) \geq \frac{n}{4}\textsc{Maj}_W(W_n).$$

These two families of sequences show that $\textsc{Maj}$ and $\textsc{Eag}$ are incomparable under relative worst order analysis. □

6 Conclusion and Future Work

The frequent items problem for streaming was considered as an online problem. Three deterministic algorithms, Nai, Maj, and Eag were compared using three different quality measures: competitive analysis, relative worst order analysis, and relative worst order ratio. According to competitive analysis, Nai is the better algorithm and Maj and Eag are equivalent. According to relative interval analysis, Nai and Maj are equally good and both are better than Eag. According to relative worst order analysis, Nai and Opt are equally good and better than Maj and Eag, which are incomparable.

All three analysis techniques studied here are worst case measures. According to both competitive analysis and relative worst order analysis, Nai is the best possible online algorithm, and according the relative worst order analysis, it is as good as Maj and better than Eag. This is a consequence of Nai being very adaptive and, as a result, good at avoiding the extreme poor performance cases. Both Maj and Eag attempt to keep the most frequent items in the buffer for longer than their frequency would warrant. The heuristic approaches hurt these algorithms in the worst case.

Relative interval analysis compares the algorithms on the same sequence in a manner which, in addition to the worst case scenarios, also takes the algorithms' best performance into account to some extent. This makes Maj's sometimes superior performance visible, whereas Eag, not being adaptive at all, does not benefit in the same way from its best performance. In some sense, Maj's behavior can be seen as swinging around the behavior of Nai, with worse behavior on some sequences counter-acted by correspondingly better behavior on other sequences.

Our conclusion is that purely worst behavior measures do not give indicative results for this problem. Relative interval analysis does better, and should possibly be supplemented by some expected case analysis variant. To that end, natural performance measures to consider would be bijective and average analysis [1]. However, as the problem is stated in [15] and studied here, the frequent items problem has an infinite universe from which the items are drawn. Thus, these analysis techniques cannot be applied directly to the problem in any meaningful way. Depending on applications, it could be realistic to assume a finite universe. This might give different results than those obtained here, and might allow the problem to be studied using other measures, giving results dependent on the size of the universe. Another natural extension of this work is to consider multiple buffers, which also allows for a richer collection of algorithms [3], or more complicated, not necessarily discrete, objective functions [11].

References

1. S. Angelopoulos, R. Dorrigiv, and A. López-Ortiz. On the separation and equivalence of paging strategies. In *Proceedings 18th ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 229–237, 2007.

2. L. Becchetti and E. Koutsoupias. Competitive analysis of aggregate max in windowed streaming. In S. Albers, A. Marchetti-Spaccamela, Y. Matias, S.E. Nikoletseas, and W. Thomas, editors, *ICALP(1) 2009*, volume 5555 of *LNCS*, pages 156–170. Springer, Heidelberg, 2009.
3. R. Berinde, G. Cormode, P. Indyk, and M.J. Strauss. Space-optimal heavy hitters with strong error bounds. In *Proceedings 28th ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems (PODS)*, pages 157–166, 2009.
4. J. Boyar and L.M. Favrholdt. The relative worst order ratio for online algorithms. *ACM Trans. Algorithms*, 3, 2007.
5. J. Boyar, L.M. Favrholdt, and K.S. Larsen. The relative worst order ratio applied to paging. *J. Comput. System Sci.*, 73(5):818–843, 2007.
6. J. Boyar, S. Gupta, and K.S. Larsen. Access graphs results for LRU versus FIFO under relative worst order analysis. In F.V. Fomin and P. Kaski, editors, *SWAT 2012*, volume 7357 of *LNCS*, pages 328–339. Springer, Heidelberg, 2012.
7. J. Boyar, S. Gupta, and K.S. Larsen. Relative interval analysis of paging algorithms on access graphs. In *WADS 2013*, LNCS, 2013. Accepted for publication.
8. J. Boyar, S. Irani, and K.S. Larsen. A comparison of performance measures for online algorithms. In F.K.H.A. Dehne, M.L. Gavrilova, J.-R. Sack, and C.D. Tóth, editors, *WADS 2009*, volume 5664 of *LNCS*, pages 119–130. Springer, Heidelberg, 2009.
9. J. Boyar, K.S. Larsen, and A. Maiti. A comparison of performance measures via online search. In J. Snoeyink, P. Lu, K. Su, and L. Wang, editors, *FAW-AAIM 2012*, volume 7285 of *LNCS*, pages 303–314. Springer, Heidelberg, 2012.
10. J. Boyar, K.S. Larsen, and A. Maiti. The frequent items problem in online streaming under various performance measures. arXiv:1306.0771 [cs.DS], 2013.
11. E. Cohen and M.J. Strauss. Maintaining time-decaying stream aggregates. *J. Algorithms*, 59(1):19–36, 2006.
12. G. Cormode and M. Hadjieleftheriou. Finding frequent items in data streams. *Proceedings of the VLDB Endowment*, 1(2):1530–1541, 2008.
13. R. Dorrigiv and A. López-Ortiz. A survey of performance measures for on-line algorithms. *SIGACT News*, 36(3):67–81, 2005.
14. R. Dorrigiv, A. López-Ortiz, and J.I. Munro. On the relative dominance of paging algorithms. *Theoret. Comput. Sci.*, 410(38–40):3694–3701, 2009.
15. Y. Giannakopoulos and E. Koutsoupias. Competitive analysis of maintaining frequent items of a stream. In F.V. Fomin and P. Kaski, editors, *SWAT 2012*, LNCS, pages 340–351. Springer, Heidelberg, 2012.
16. A.R. Karlin, M.S. Manasse, L. Rudolph, and D.D. Sleator. Competitive snoopy caching. *Algorithmica*, 3:79–119, 1988.
17. D.D. Sleator and R.E. Tarjan. Amortized efficiency of list update and paging rules. *Commun. ACM*, 28(2):202–208, 1985.