Preemptively Guessing the Center*

Christian Konrad[1] and Tigran Tonoyan[2]

[1] Department of Computer Science, Centre for Discrete Mathematics and its Applications (DIMAP), University of Warwick, Coventry, UK
c.konrad@warwick.ac.uk

[2] ICE-TCS, School of Computer Science, Reykjavik University, Iceland
ttonoyan@gmail.com

Abstract. An online algorithm is typically unaware of the length of the input request sequence that it is called upon. Consequently, it cannot determine whether it has already processed most of its input or whether the bulk of work is still ahead.

In this paper, we are interested in whether some sort of orientation within the request sequence is nevertheless possible. Our objective is to preemptively guess the center of a request sequence of unknown length n: While processing the input, the online algorithm maintains a guess for the central position $n/2$ and is only allowed to update its guess to the position of the current element under investigation. We show that there is a randomized algorithm that in expectation places the guess at a distance of $0.172n$ from the central position $n/2$, and we prove that this is best possible. We also give upper and lower bounds for a natural extension to weighted sequences.

This problem has an application to preemptively partitioning integer sequences and is connected to the online bidding problem.

1 Introduction

Online Algorithms. Online algorithms process their inputs item by item in a linear fashion. They are characterized by the fact that the algorithm's decision as to how to process the current input item is irrevocable. A key difficulty in the design of online algorithms is that they are typically unaware of the length of the input request sequence[3]. Indeed, for many online problems (e.g. problems with a rent or buy flavor such as the ski rental problem [2]), knowing the input length would allow the algorithm to solve the problem optimally. Without knowing the input length, online algorithms are unable to determine the position of the current element within the request sequence.

* C. Konrad is supported by the Centre for Discrete Mathematics and its Applications (DIMAP) at Warwick University and by EPSRC award EP/N011163/1. T. Tonoyan is supported by grants no. 152679-05 and 174484-05 from the Icelandic Research Fund.

[3] An exception are online algorithms with advice, where the online algorithm receives additional input bits prior to the processing of the request sequence. These bits can be used to encode the input length. See [1] for a recent survey.

Guessing the Center. In this paper, we ask whether we can nevertheless obtain some sort of orientation within the request sequence. We study the natural task of guessing the central position $n/2$ within a request sequence of unknown length n in an online fashion. In this problem, the online algorithm maintains a guess of the central position while processing the input request sequence. The algorithm is only allowed to update its guess to the position of the current element under investigation. It may thus potentially update the guess many times, however, each update bears the risk that the input sequence may end very soon and the guess is thus far from the center. Such an algorithm follows the following scheme:

Algorithm 1 Scheme for Preemptively Guessing the Center

```
$p \leftarrow 0$ {initialization of our guess}
**for** each request $j = 1, 2, \ldots, n$ **do** {$n$ is unknown}
  **if** *TODO: add condition here* **then** {update guess}
    $p \leftarrow j$
**return** $p$
```

We also study a generalization of this problem to weighted requests. This is best modelled as follows. The online algorithm processes a sequence $X = w_1, w_2, \ldots, w_n$ of positive integers. Let $W = \sum_{i=1}^{n} w_i$ be the total weight of the sequence. We assume that there exists an index m with $1 \leq m \leq n$, such that $\sum_{i=1}^{m} w_i = \sum_{i=m+1}^{n} w_i$, i.e., the sequence can be split into two parts of equal weight. This assumption is necessary for a meaningful problem definition as we will discuss in Section 4.1 in more detail. While processing X, an online algorithm $\mathcal{A}$ maintains a guess p for the position m as in the unweighted case. The objective is to minimize the weight between the guess p and the position m of the central weight, that is, the *deviation*

$$\Delta_{\mathcal{A}}^{X} := \sum_{i=\min\{p,m\}+1}^{\max\{p,m\}} w_i ,$$

is to be minimized, where $\mathcal{A}$ refers to the employed algorithm and X is the input sequence. Note that the unweighted version of this problem is obtained by setting $w_i = 1$, for every $1 \leq i \leq n$. One property of this definition is that we only consider unweighted sequences of even length, since sequences of odd lengths cannot be split into two parts of equal weight. This is only for convenience; a meaningful problem statement with similar results for unweighted sequences of odd lengths can easily be derived from this work. For unweighted sequences we write $\Delta_{\mathcal{A}}^{n}$ instead of $\Delta_{\mathcal{A}}^{X}$, where n denotes the input length.

Results. For unweighted request sequences, we give an optimal randomized preemptive online algorithm for guessing the center. Our algorithm has expected deviation $0.172n$ from the central position $n/2$ (**Theorem 1**). Our main result is a lower bound, which shows that this is best possible (**Theorem 3**). We further give a barely random algorithm that uses only a single random bit and reports a position with expected deviation $0.25n$. This is also proved to be best possible for the class of algorithms that only use a single random bit. For weighted sequences, we give a randomized preemptive online algorithm that reports a position with expected deviation $0.313W$, where W is the total weight of the input sequence (**Theorem 4**). This is complemented by a lower bound of $0.25W$ (**Theorem 5**). Closing this gap proves challenging and is left as an open problem.

Techniques. Both our algorithms for unweighted and weighted sequences employ the doubling method with a random seed. In the unweighted case, our algorithm updates its guess to the current position j if $j \in \{\lceil x^{i\delta} \rceil \mid i \in \mathbb{N}\}$ (this condition is slightly different in the weighted case), where $x > 2$ is an optimized parameter that determines the step size between the guesses (this parameter is different for weighted and unweighted sequences), and $\delta \in (0, 1)$ is a seed that is chosen uniformly at random. This technique is well known and has previously been applied for various problems, see for example [3]. While our algorithms are extremely simple, their analyses require careful case distinctions.

Our main result is a lower bound for unweighted sequences, which proves that the doubling method is optimal. The argument relies on Yao's Minimax principle [4]. We define a hard input distribution where the probability of a specific input length is inversely proportional to its length. We then argue that a deterministic guessing algorithm, which can be identified by a sequence of increasing positions at which it updates its guess, will in expectation (over the hard input distribution) have a deviation of $0.172n$ from the central position. By Yao's Minimax principle, this implies that our algorithm for unweighted sequences is best possible. This argument is the most technical contribution of the paper. The lower bound for weighted sequences follows the same line, however, it uses a sequence of exponentially increasing weights.

Further Related Work. Preemptively guessing the center is strongly related to the online bidding problem [5]. In online bidding, the objective is to guess an unknown target value. The algorithm submits increasing guesses until a guess that is at least as large as the target value is submitted. For this problem, the usual cost function is the sum of the submitted guesses, which is very different from our cost function. However, similarly to the problem of guessing the center, an optimal randomized strategy can be obtained by using a sequence of exponentially increasing guesses.

Guessing the center is a special case of the problem of partitioning integer sequences. In this problem, an integer array A of length n and an integer $p \geq 2$ is given, and the goal is to find $(p-1)$ separator positions $s_1, s_2, \ldots, s_{p-1}$ with $1 = s_0 \leq s_1 \leq s_2 \leq \cdots \leq s_{p-1} \leq s_p = n+1$ such that $\max\{\sum_{i=s_j}^{s_{j+1}-1} A_i \mid 0 \leq j < p\}$ is minimized. This load balancing task has a long history in the offline setting (e.g. [6,7,8,9]) and has recently been studied in the context of data streams [10] and online algorithms [11] by the authors of this paper. In the preemptive online model, an algorithm is only allowed to insert a new partition separator at the current position, and, once all separators have been placed, previously inserted separators can be removed and then reinserted again. As shown in [11], a 2-approximation algorithm for arbitrary values of p can be obtained. The special case $p = 2$ boils down to determining the central position of an integer sequence using a preemptive guessing scheme. The problem studied in this paper thus correspond to preemptively partitioning an integer sequence of length n into two parts of equal weights.

Outline. We give our algorithm for unweighted sequences in Section 2 and our lower bound for unweighted sequences in Section 3. In Section 4, we address extensions to weighted sequences. We conclude with an open problem in Section 5. Due to space restrictions, we only sketch the proofs of Theorems 4 and 5 are defer the full proofs to the complete version of this paper.

2 Algorithm For Guessing the Center

Our algorithm, denoted $\mathcal{A}_x$, is parametrized by a real number $x > 2$. It employs a well-known doubling technique with randomized seeding. We first pick a seed $\delta \in (0, 1)$ uniformly at random. The parameter x determines the distance between two consecutive guesses and will be optimized later. The algorithm updates our guess for the central position whenever we process requests $\lceil x^{1+\delta} \rceil$, $\lceil x^{2+\delta} \rceil$, $\lceil x^{3+\delta} \rceil$, ... This is depicted in Algorithm 2.

Algorithm 2 Algorithm $\mathcal{A}_x$ for guessing the center

```
Choose uniform random $\delta \in (0,1)$, $i \leftarrow 0$, $p \leftarrow 0$ {initialization}
for each request $j = 1, 2, \ldots, n$ do {$n$ is unknown}
  if $j = \lceil x^{i+\delta} \rceil$ then {update guess}
    $p \leftarrow j$
    $i \leftarrow i + 1$
return $p$
```

While the suggested doubling strategy is fairly standard, the analysis requires a very careful case distinction. Moreover, this algorithm is optimal, which will be proved in Section 3.

One may wonder about the choice of δ to be a real-valued quantity of presumably infinite precision. This assumption is only taken for convenience in the analysis; a bit precision of $\mathrm{O}(\log n)$ is enough to provide sufficient granularity. This does not mean that n needs to be known in advance in order to determine the $\mathrm{O}(\log n)$ random bits: We can choose additional random bits for the description of δ when necessary as the algorithm proceeds.

After giving the analysis of our main algorithm, we further present an algorithm that uses only a single random bit and achieves an expected deviation of $0.25n$. We also prove that this is best possible for the class of algorithms that only use a single random bit. Observe that deterministic algorithms (i.e., using no randomness at all) fail for guessing the center: If the input sequence ends exactly when the deterministic algorithm has updated its guess, then the deviation is as large as it could be. Without randomness, this is unavoidable.

Theorem 1. *There is a constant $x \approx 3.052$ such that:*

$$\mathbb{E}\left[\Delta^n_{\mathcal{A}_x}\right] \approx 0.172n + \mathrm{O}(1) \ .$$

Proof. Let $\alpha \in [0,1)$ and $i \in \mathbb{N}$ be such that $n = 2x^{i+\alpha}$. Then the central position is $\frac{n}{2} = x^{i+\alpha}$. In order to bound the expected deviation, we conduct a case distinction for various ranges of α and δ. We distinguish two ranges for α, and within each case, we distinguish three ranges of δ.

Case 1: $\alpha > 1 - \log_x 2$ (note that we assumed $x > 2$). In order to bound $\Delta^n_{\mathcal{A}_x}$, we split the possible values of δ into three subsets:

– If $\delta \in (0, \alpha + \log_x 2 - 1]$, then we have that $x^{\delta+i+1} \leq 2x^{i+\alpha} = n$. In this case, the deviation is $\Delta^n_{\mathcal{A}_x} = x^{\delta+i+1} - n/2 = x^{\delta+i+1} - x^{i+\alpha}$.
– If $\delta \in (\alpha + \log_x 2 - 1, \alpha]$, then we have that $x^{\delta+i+1} > n$ but $x^{\delta+i} \leq \frac{n}{2}$. In this case, $\Delta^n_{\mathcal{A}_x} = x^{i+\alpha} - x^{\delta+i}$.
– If $\delta \in (\alpha, 1)$, then we have that $x^{\delta+i+1} > n$ and $x^{\delta+i} \in (\frac{n}{2}, n)$. In this case, $\Delta^n_{\mathcal{A}_x} = x^{\delta+i} - x^{i+\alpha}$.

Using these observations, we can bound the expected deviation as follows:

$$\begin{aligned}\mathbb{E}\left[\Delta^n_{\mathcal{A}_x}\right] &= \int_0^{\alpha+\log_x 2-1} (x^{\delta+i+1} - x^{i+\alpha}) d\delta + \int_{\alpha+\log_x 2-1}^{\alpha} (x^{\alpha+i} - x^{\delta+i}) d\delta \\ &\quad + \int_\alpha^1 (x^{\delta+i} - x^{\alpha+i}) d\delta \\ &= x^{i+\alpha} \cdot \left(1 - 2\log_x 2 + \frac{2}{x \ln x}\right).\end{aligned}$$

Case 2: $\alpha \leq 1 - \log_x 2$. We deal with this case similarly, but we need to group the possible values for δ in a different way:

– If $\delta \in (0, \alpha]$, then $x^{\delta+i+1} > n$ but $x^{\delta+i} \leq \frac{n}{2}$. In this case, $\Delta^n_{\mathcal{A}_x} = x^{i+\alpha} - x^{\delta+i}$.
– If $\delta \in (\alpha, \alpha + \log_x 2]$, then $x^{\delta+i} > \frac{n}{2}$ and $x^{\delta+i} \leq n$. In this case, $\Delta^n_{\mathcal{A}_x} = x^{\delta+i} - x^{i+\alpha}$.
– If $\delta \in (\alpha + \log_x 2, 1)$, then $x^{\delta+i} > n$. In this case, $\Delta^n_{\mathcal{A}_x} = x^{i+\alpha} - x^{\delta+i-1}$.

Plugging the values above in the formula for the expected value, we obtain a different sum of integrals, which however leads to the same function as above:

$$\begin{aligned}\mathbb{E}\left[\Delta^n_{\mathcal{A}_x}\right] &= \int_0^{\alpha} (x^{\alpha+i} - x^{\delta+i}) d\delta + \int_{\alpha}^{\alpha+\log_x 2} (x^{\delta+i} - x^{i+\alpha}) d\delta \\ &\quad + \int_{\alpha+\log_x 2}^1 (x^{\alpha+i} - x^{\delta+i-1}) d\delta \\ &= x^{i+\alpha} \cdot \left(1 - 2\log_x 2 + \frac{2}{x \ln x}\right).\end{aligned}$$

Moreover, the factor $1 - 2\log_x 2 + \frac{2}{x \ln x}$ above is independent of α. Thus, it remains to find a value of x that minimizes $f(x) \overset{def}{=} 1 - 2\log_x 2 + \frac{2}{x \ln x}$. Observe that $f'(x) = -\frac{2}{x^2 \ln^2 x} - \frac{2}{x^2 \ln x} + \frac{\ln 2}{x \ln^2 x}$, and $f'(x) = 0$ if and only if $x = \log_2(ex)$. With a simple transformation, the latter is equivalent to $ze^z = -\frac{\ln 2}{e}$ with $z = -x \ln 2$, so the value that minimizes $f(x)$ can be computed as $x_{\min} = -\frac{W_{-1}(-\ln 2/e)}{\ln 2} \approx 3.052$, where W_{-1} is the lower branch of Lambert's W function. The claim of the theorem follows by calculating $f(x_{\min}) \approx 0.344$. The additive $O(1)$ corresponds to the approximation of the finite range of δ by a continuous distribution. $\square$

Next, we give an algorithm that only relies on a single random bit. We prove that its expected deviation from the center is $0.25n$, which is best possible.

Algorithm 3 Single-bit algorithm $\mathcal{A}_0$

```
$i \leftarrow 0$ or 1 with probability 1/2 each, $p \leftarrow 0$ {initialization}
for each request $j = 1, 2, \ldots n$ do {$n$ is unknown}
  if $j = 2^i$ then
    $p \leftarrow j$ {update guess to current position}
    $i \leftarrow i + 2$
return $p$
```

Theorem 2. *The expected deviation of algorithm $\mathcal{A}_0$ is $\mathbb{E}\left[\Delta^n_{\mathcal{A}_0}\right] \leq 0.25n$, which is optimal for the class of algorithms that only use a single random bit.*

Proof. Let $\alpha \in [0, 1)$ and $i \in \mathbb{N}$ be such that the length of the sequence is $n = 2 \cdot 2^{i+\alpha}$. Since $2^{i+2} > n$, the algorithm reports either position 2^i or position 2^{i+1}, each with probability 1/2. In the first case, the deviation from the center is $n/2 - 2^i$, while in the second case it is $2^{i+1} - n/2$. Thus, in expectation, we have, as required, $\mathbb{E}\left[\Delta^n_{\mathcal{A}_0}\right] = \frac{1}{2} \cdot (n/2 - 2^i) + \frac{1}{2} \cdot (2^{i+1} - n/2) = 2^{i-1} \leq n/4$.

In order to see that this is best possible for the class of algorithms that only use a single random bit, first observe that a randomized algorithm that uses a single random bit is a uniform distribution over two deterministic algorithms. Note further that each deterministic algorithm is a fixed (potentially infinite) sequence of positions at which it updates its guess. Suppose that $\mathcal{B}$ is such a randomized algorithm, and let $S_1 = \{p_1, p_2, \ldots\}$ and $S_2 = \{q_1, q_2, \ldots\}$ be the corresponding sequences. Now, if the sequence has length p_i for some i, $\mathcal{B}$ would have maximal deviation if it chooses the first sequence (with probability 1/2), and may have minimal deviation 0 (with the remaining 1/2 probability) if the largest $q_j \leq p_i$ is equal to $p_i/2$. Therefore the smallest expected deviation achievable is $n/4$, which implies that our algorithm is optimal. $\square$

3 Lower Bound

We prove that no algorithm can achieve a smaller expected deviation than the one claimed in Theorem 1. The proof applies Yao's Minimax principle and uses a hard input distribution over all-ones sequences of length $n \in [n_{\min}, n_{\max}]$, for some large values of $n_{\min}$ and $n_{\max}$, where the probability that the sequence is of length n is proportional to $1/n$.

Theorem 3. *For any randomized algorithm $\mathcal{A}$, the expected deviation is*

$$\mathbb{E}\left[\Delta^n_{\mathcal{A}}\right] \geq 0.172n.$$

Proof. We will prove the theorem by using Yao's Minimax principle [4]. To this end, let us first consider an arbitrary *deterministic* algorithm $\mathcal{A}_{\text{det}}$. Assume the length of the sequence is random in the interval $X := [n_{\min}, n_{\max}]$ for large values of $n_{\max}$ and $n_{\min}$ with $n_{\max} > 2 \cdot n_{\min}$ and has the following distribution: The sequence ends at position $n \in X$ with probability p_n which is proportional to $\frac{1}{n}$, i.e., using the definition $S = \displaystyle\sum_{m=n_{\min}}^{n_{\max}} \frac{1}{m}$, we have $p_n := \mathbb{P}[\text{sequence is of length } n] = \frac{1}{n \cdot S}$.

In order to apply the Minimax principle, we will consider a *normalized* measure of the performance of an algorithm. For an algorithm $\mathcal{A}$, let $B^n_{\mathcal{A}}$ denote the larger of the two parts created by the algorithm for a sequence of length n, and let $R^n_{\mathcal{A}} = \frac{B^n_{\mathcal{A}}}{n/2} \in [1, 2]$. Then it is easily verified that

$$\Delta^n_{\mathcal{A}} = B^n_{\mathcal{A}} - \frac{n}{2} = n \cdot \frac{R^n_{\mathcal{A}} - 1}{2}.$$

We will show that for *each* deterministic algorithm $\mathcal{A}_{\text{det}}$, if the input is distributed as above, then $\mathbb{E}\left[R^n_{\mathcal{A}_{\text{det}}}\right] \geq 1.344 - \mathrm{O}(\ln^{-1} \frac{n_{\max}}{n_{\min}})$, where the expectation is taken over the distribution of n. Then, by the Minimax principle, every randomized algorithm $\mathcal{A}$ has a ratio of at least $R^n_{\mathcal{A}} \geq \mathbb{E}\left[R^n_{\mathcal{A}_{\text{det}}}\right] \geq 1.344 - \mathrm{O}(\ln^{-1} \frac{n_{\max}}{n_{\min}})$. Since the ratio $\frac{n_{\max}}{n_{\min}}$ is arbitrary, this implies the theorem.

Let J denote the set of requests at which $\mathcal{A}_{\text{det}}$ updates its guess when processing the all-ones sequence of length $n_{\max}$. Note that the positions of guess updates by $\mathcal{A}_{\text{det}}$ on sequences of shorter lengths are a subset of J. Let $I = J \cap X = \{i_1, \ldots, i_k\}$ (the i_j are ordered with increasing value).

We bound $\mathbb{E}\left[R^n_{\mathcal{A}_{\text{det}}}\right] = \sum_{n=n_{\min}}^{n_{\max}} p_n R^n_{\mathcal{A}_{\text{det}}}$ by separately bounding every partial sum in the following decomposition:

$$\mathbb{E}\left[R^n_{\mathcal{A}_{\text{det}}}\right] = E(n_{\min}, i_1) + E(i_1, i_2) + \cdots + E(i_{k-1}, i_k) + E(i_k, n_{\max}),$$

where for each $a < b$, $E(a, b) = \sum_{n=a}^{b-1} p_n R^n_{\mathcal{A}_{\text{det}}}$. The first and last terms need special care, so we will start with bounding the other terms. In the following, $H^q_p = \sum_{n=p}^{q} \frac{1}{n}$ denotes partial harmonic sums for $q \geq p \geq 1$. Observe that $S = H^{n_{\max}}_{n_{\min}}$. We proceed in three steps:

1. Consider an index $1 \leq j < k$ and let us bound the sum $E(i_j, i_{j+1})$. Let us denote $a = i_j$ and $b = i_{j+1}$. We need to consider two cases.

   Case 1: $b \leq 2a$. Then for all $n \in \{a, \ldots, b-1\}$, we have $B^n_{\mathcal{A}_{det}} = a$ (since $n/2 < a$). Then:

$$E(a, b) \geq \sum_{n=a}^{b-1} \frac{1}{nS} \cdot \frac{a}{n/2} \geq \frac{2a}{S} \sum_{n=a}^{b-1} \frac{1}{n(n+1)} = \frac{2a}{S}\left(\frac{1}{a} - \frac{1}{b}\right) > 1.4 \cdot \frac{H^{b-1}_a}{S}, \tag{1}$$

   where the last inequality can be proved as follows. First, it can be checked by direct inspection that for $\Phi(a, b) = 2(1 - \frac{a}{b})/H^{b-1}_a$ and $b > a$, it holds that $\Phi(a,b+1) < \Phi(a,b)$. Hence, recalling that $b \leq 2a$ and using standard approximations of harmonic sums, we obtain

$$\Phi(a,b) \geq \Phi(a,2a) = 1/H_a^{2a-1} \geq \frac{1}{\ln 2 + \frac{1}{a}} > 1.4,$$

where the last inequality holds for large enough $a = i_j$, e.g., $i_j \geq \frac{n_{\min}}{2} \geq 50$.
Case 2: $b > 2a$. In this case, for all $n = a, \ldots, 2a-1$, if the sequence is of length n, then $B^n_{\mathcal{A}_{det}} = a$, as in case 1. However, when $n \geq 2a$, then $n/2 \geq a$, so $B^n_{\mathcal{A}_{\text{det}}} = n - a$. Using these observations, we can bound $\mathbb{E}\left[R^n_{\mathcal{A}_{\text{det}}}\right]$:

$$\begin{aligned}
\mathbb{E}\left[R^n_{\mathcal{A}_{\text{det}}}\right] &= \sum_{n=a}^{2a-1} \frac{1}{nS}\frac{a}{n/2} + \sum_{n=2a}^{b-1} \frac{1}{nS}\frac{n-a}{n/2} \\
&\geq \frac{2a}{S}\sum_{n=a}^{2a-1}\frac{1}{n(n+1)} + \frac{2}{S}H_{2a+1}^b - \frac{2a}{S}\sum_{n=2a}^{b-1}\frac{1}{n(n+1)} \\
&= \frac{2a}{S}\left(\frac{1}{a} - \frac{1}{2a}\right) + \frac{2}{S}H_{2a+1}^b - \frac{2a}{S}\left(\frac{1}{2a} - \frac{1}{b}\right) \\
&= \frac{2a}{Sb} + \frac{2}{S}(H_a^b - H_a^{2a}) \\
&= \frac{H_a^b}{S}\cdot\left(2 + \frac{2a}{bH_a^b} - \frac{2H_a^{2a}}{H_a^b}\right),
\end{aligned}$$

where the third line is obtained by using the identity $H_{2a+1}^b = H_a^b - H_a^{2a}$. Again, using a standard approximation for harmonic sums and setting $x = \frac{b}{a}$, we can approximate:

$$\begin{aligned}
2 + \frac{2a}{bH_a^b} - \frac{2H_a^{2a}}{H_a^b} &\geq 2 + \tfrac{2a}{b\ln\frac{b}{a}} - \tfrac{\ln 2}{\ln\frac{b}{a}} - \mathrm{O}(a^{-1}) \\
&= 2 + \tfrac{2}{x\ln x} - 2\log_x 2 - \mathrm{O}(a^{-1}).
\end{aligned}$$

Note that the function $f(x) = 2 + \frac{2}{x\ln x} - 2\log_x 2$ is exactly the same that was minimized in the proof of Thm. 1, with an additional term $+1$, and achieves its minimum in $(1,\infty)$ at $x_{\min} \approx 3.052$, giving $f(x_{\min}) \approx 1.344$. Thus, we have $E(i_j, i_{j+1}) \geq \frac{H_{i_j}^{i_{j+1}-1}}{S}(1.344 - \mathrm{O}(\frac{1}{i_j}))$.

2. The term $E(i_k, n_{\max})$ can be bounded by $\frac{H_{i_k}^{n_{\max}-1}}{S}(1.344 - \mathrm{O}(\frac{1}{i_k}))$ by an identical argument as above.
3. The term $E(n_{\min}, i_1)$ needs a slightly different approach. Let i_0 denote the last position where the algorithm updates its guess before $n_{\min}$. We can assume that $i_0 \geq n_{\min}/2$, as otherwise the algorithm could only profit by updating its guess at position $n_{\min}/2$. We consider two cases. First, if $i_1 \leq 2i_0$, then we simply assume the algorithm performs optimally in the range $[n_{\min}, i_1)$:

$$E(n_{\min}, i_1) = \frac{H_{n_{\min}}^{i_1-1}}{S} \geq 1.344\cdot\frac{H_{n_{\min}}^{i_1-1}}{S} - 0.5\frac{H_{n_{\min}}^{i_1-1}}{S} > 1.344\cdot\frac{H_{n_{\min}}^{i_1-1}}{S} - 1/S,$$

since (recalling that $i_1 \le 2i_0 \le 2n_{\min}$), $H_{n_{\min}}^{i_1} < H_{n_{\min}}^{2n_{\min}} < 1$.
On the other hand, when $i_1 > 2i_0$ (and by the discussion above, $2i_0 \ge n_{\min}$), then with calculations similar to the one in Case 2 above, we obtain:

$$\begin{aligned} E(x_{\min}, i_1) &= \sum_{n=n_{\min}}^{2i_0-1} \frac{1}{nS}\frac{i_0}{n/2} + \sum_{n=2i_0}^{i_1-1} \frac{1}{nS}\frac{n-i_0}{n/2} \\ &\ge \frac{2i_0}{S}\sum_{n=n_{\min}}^{2i_0-1}\frac{1}{n(n+1)} + \frac{2}{S}H_{2i_0+1}^{i_1} - \frac{2i_0}{S}\sum_{n=2i_0}^{i_1-1}\frac{1}{n(n+1)} \\ &\ge \frac{2}{S}\left(\frac{i_0}{n_{\min}} + \frac{i_0}{i_1} + H_{n_{\min}}^{i_1} - H_{n_{\min}}^{2i_0}\right) \ge 2\cdot\frac{H_{n_{\min}}^{i_1}}{S} - \mathrm{O}(1/S), \end{aligned}$$

because $i_0 \le n_{\min}$ and thus $H_{n_{\min}}^{2i_0} < 1$.

It remains to plug the obtained estimates into Inequality 1:

$$\begin{aligned} \mathbb{E}\left[R_{\mathcal{A}_{\det}}^n\right] &= E(n_{\min}, i_1) + \sum_{j=1}^{k-1} E(i_j, i_{j+1}) + E(i_k, n_{\max}) \\ &\ge (1.344 - \mathrm{O}(1/n_{\min}))\cdot\frac{H_{n_{\min}}^{i_1} + \sum_{j=1}^{k-1} H_{i_j}^{i_{j+1}-1} + H_{i_k}^{n_{\max}}}{S} - \mathrm{O}(1/S) \\ &= 1.344 - \mathrm{O}(1/S). \end{aligned}$$

This completes the proof. □

4 Weighted Sequences

4.1 Algorithm

The algorithm for weighted instances is an adaptation of the algorithm $\mathcal{A}_x$ presented in Section 2. Namely, the guess is updated as soon as adding the current weight w_j to the weight of the prefix that has already been processed $\sum_{i=1}^{j-1} w_i$ increases the total weight to be at least $\lceil x^{i+\delta}\rceil$, i.e., if $\sum_{i=1}^{j-1} w_i < \lceil x^{i+\delta}\rceil \le \sum_{i=1}^{j} w_i$. We will keep the notation $\mathcal{A}_x$ for the modified algorithm.

Theorem 4. *There is a constant value of $x \approx 5.357$ such that*

$$\mathbb{E}\left[\Delta_{\mathcal{A}_x}^X\right] \le 0.313W + O(1)$$

holds for every weighted sequence X of total weight W.

Proof (Sketch). Let $X = w_1, w_2, \ldots, w_n$ be the input sequence of total weight W, and let m be such that $\sum_{i\le m} w_i = \frac{W}{2}$. Then, w_m is the *central weight* of the sequence. We will argue first that replacing all w_i left of w_m including w_m by a sequence of $\sum_{i\le m} w_i$ unit requests, and replacing all w_i right of w_m by a single large request of weight $\sum_{i>m} w_i$ worsens the approximation factor of the algorithm. Indeed, suppose that the algorithm attempts to update its guess at a position j that falls on an element w_i, which is located left of w_m. Then the algorithm updates its guess after w_i, bringing the guess closer to the center than if w_i were of unit weight. Similarly, suppose that the algorithm attempts to update its guess at position j that falls on an element w_i, which is located right of w_m. By replacing all weights located to the right of w_m by a single heavy element, the algorithm has to place its guess at the end of the sequence, which gives the worst possible deviation. Thus, we suppose from now on that X is of the form $X = 1 \ldots 1B$, where $B = W/2$.

After this simplification, via a case distinction as in the proof of Thm. 1 it can be shown that $E[\Delta^X_{\mathcal{A}_x}] \leq \frac{W}{2} \cdot g(x) + O(1)$, where $g(x) \overset{def}{=} 1 - \frac{1}{\ln x} + \frac{2}{x \ln x}$. It can then be shown that $x_{min} = -2W_{-1}(-\frac{1}{2e}) \approx 5.3567$ minimizes $g(x)$ and $g(x_{min}) \approx 0.627$, which implies the theorem. $\square$

Remark. Recall that we work with the assumption that the input sequence $X = w_1, \ldots, w_n$ can be split into two parts of exactly equal weight. This may seem like an artificial restriction. It is, however, necessary for a meaningful problem definition: Suppose we allowed arbitrary sequences and the goal is to minimize the distance between the guess and the most central position, i.e., the position c such that $\max\{\sum_{i=1}^{c} w_i, \sum_{i=c+1}^{n} w_i\}$ is minimized. Consider now the instance $X = 11 \ldots 1W$, where W is extremely heavy (e.g., W is a 0.99 fraction of the entire sequence). Then the most central position is the position of the last 1, while an algorithm that places a guess after W is at distance W from the most central position. We believe that this should not be penalized since such an input sequence does not have a good central position. An alternative problem formulation, which is meaningful when applied to the previously described instance, is obtained when asking for a guess that minimizes the size of the larger part as compared to the larger part in a most central split. Indeed, this formulation is coherent with the problem of partitioning integer sequences. Our algorithm for weighted sequences can be analyzed for this situation and gives a 1.628 approximation.

4.2 Lower Bound

Note that the expected deviation of $0.313W$ is tight for the algorithm $\mathcal{A}_x$ on weighted sequences. This is achieved on sequences consisting of $W/2$ unit weight elements followed by an element of weight $W/2$. However, we were not able to obtain a matching lower bound. The main difficulty in applying the Minimax principle is that in the weighted case, the deterministic algorithm may know the probability distribution of the individual weights. Instances similar to the one described above become easy if the algorithm knows their structure. Instead, we prove a lower bound of $0.25W$ using a different construction.

Theorem 5. *For every randomized algorithm $\mathcal{A}$, there is a weighted instance X of total weight W, such that the expected deviation is $\mathbb{E}\left[\Delta^X_{\mathcal{A}}\right] \geq 0.25W - O(1)$.*

Proof (Sketch). Let $X_i = 2^0, 2^0, 2^1, 2^2, \ldots, 2^{i-1}$ denote the exponentially increasing sequence of length i, and $W_i = 2^i$ denotes the weight of X_i. Note that the middle of X_i is the position before the weight 2^{i-1}. Let further $n_{\min}, n_{\max}$ be large integers with $n_{\max} \geq 2n_{\min}$, and denote $S = [n_{\min}, n_{\max}]$. We consider the performance of any deterministic algorithm $\mathcal{A}_{\text{det}}$ on the uniform input distribution over the set $\Sigma = \{X_i : i \in S\}$ of exponentially increasing sequences. As in the proof of Thm. 3, we need a normalized performance measure in order to apply the Minimax lemma. For an algorithm $\mathcal{A}$, let $B_{\mathcal{A}}^X$ denote the larger of the two parts that the algorithm creates on the input sequence X, and let $R_{\mathcal{A}}^X = \frac{B_{\mathcal{A}}^X}{W/2} \in [1, 2]$. Again, note that $\Delta_{\mathcal{A}}^X = B_{\mathcal{A}}^X - W/2 = W \cdot (R_{\mathcal{A}}^X - 1)/2$. Hence, showing that $\mathbb{E}\left[R_{\mathcal{A}_{\text{det}}}^X\right] \geq 1.5$ for every deterministic algorithm implies the corresponding bound $\mathbb{E}\left[\Delta_{\mathcal{A}}^X\right] \geq 0.25W$ for every randomized algorithm, by Yao's lemma.

Let J be the set of positions where $\mathcal{A}_{\text{det}}$ updates its guess on input $X_{n_{\max}}$. Note that the set of positions on any other input of Σ is a subset of J. Let $I = J \cap S = \{i_1, \ldots, i_k\}$ be the positions within the interval S (ordered so that $i_j < i_{j+1}$, for every j).

We bound now the expected ratio of $\mathcal{A}_{\text{det}}$, where the expectation is taken over the inputs Σ. In the formulas below, we will use the abbreviation $R^i = R_{\mathcal{A}_{\text{det}}}^{X_i}$, as we will only deal with such ratios. Then:

$$\mathbb{E}\left[R^n\right] = \frac{1}{n_{\max} - n_{\min} + 1} \cdot \sum_{n=n_{\min}}^{n_{\max}} R^n \text{ , and}$$

$$\sum_{n=n_{\min}}^{n_{\max}} R^n = \underbrace{\sum_{n=n_{\min}}^{i_1-1} R^n}_{I} + \sum_{n=i_1}^{i_2-1} R^n + \cdots + \sum_{n=i_{k-1}}^{i_k-1} R^n + \underbrace{\sum_{n=i_k}^{n_{\max}} R^n}_{II} .$$

We bound I, II, and $\sum_{n=i_j}^{i_{j+1}-1} R^n$ for every $1 \leq j \leq k-1$, separately. Recall that for every i the half-weight of X_i is $W_i/2 = 2^{i-1}$. First, observe that for every $1 \leq j \leq k-1$, we have the worst ratio $R^{i_j} = \frac{2^{i_j}}{2^{i_j-1}} = 2$ when the sequence ends at a guess update, and the best ratio $R^{i_j+1} = 1$ when it ends one item after a guess update and if $i_j + 1 < i_{j+1}$. Generally, when it ends at an intermediate position $i_j + a$ such that $a \geq 2$ and $i_j + a < i_{j+1}$, then the last guess update is after the weight i_j, while the middle is just before $i_j + a$ so we have

$$R^{i_j+a} = \frac{2^{i_j+a} - 2^{i_j-1}}{2^{i_j+a-1}} \geq 1.75.$$

These bounds together imply that $\sum_{n=i_j}^{i_{j+1}-1} R^n \geq 1.5(i_{j+1} - i_j)$.

It can be shown by similar arguments that $I \geq 1.5(i_1 - n_{\min}) - 1.25$ and $II > 1.5(n_{\max} - i_k) - 1.25$. Putting the partial bounds together, we can bound $\mathbb{E}\left[R_{\mathcal{A}_{\text{det}}}^n\right]$ by:

$$\mathbb{E}\left[R_{\mathcal{A}_{\text{det}}}^n\right] \geq \frac{1.5(n_{\max} - n_{\min} + 1) - 2 \cdot 1.25}{n_{\max} - n_{\min} + 1} = 1.5 - \mathrm{O}(n_{\max}^{-1}).$$

Since $n_{\max}$ can be chosen arbitrarily large, the latter bound on the expected performance of every deterministic algorithm then implies the claim of the theorem by applying Yao's principle, as described above. $\square$

5 Conclusion

In this paper, we gave an algorithm for preemptively guessing the center of a request sequence. It has expected deviation $0.172n$ from the central position on an instance of length n. We proved that this is optimal. We extended our algorithm to weighted sequences and showed that it has expected deviation $0.313W$, where W is the total weight of the input sequence. We also gave a lower bound showing that no algorithm achieves an expected deviation smaller than $0.25W$.

The most intriguing open problem is to close the gap between the upper and lower bounds for weighted sequences. Progress could potentially be made by combining our lower bound for unweighted sequences with an exponentially increasing sequence as it is used in our lower bound for weighted sequences. For this to be successful, a better understanding of our lower bound for unweighted sequences could be beneficial, since it relies on a non-uniform input distribution which renders it difficult to extend it.

References

1. Boyar, J., Favrholdt, L.M., Kudahl, C., Larsen, K.S., Mikkelsen, J.W.: Online algorithms with advice: A survey. SIGACT News **47**(3) (August 2016) 93–129
2. Karlin, A.R., Manasse, M.S., Rudolph, L., Sleator, D.D.: Competitive snoopy caching. Algorithmica **3**(1) (Nov 1988) 79–119
3. Chrobak, M., Kenyon-Mathieu, C.: Sigact news online algorithms column 10: Competitiveness via doubling. SIGACT News **37**(4) (December 2006) 115–126
4. Yao, A.C.C.: Probabilistic computations: Toward a unified measure of complexity. In: Proceedings of the 18th Annual Symposium on Foundations of Computer Science. SFCS '77, Washington, DC, USA, IEEE Computer Society (1977) 222–227
5. Chrobak, M., Kenyon, C., Noga, J., Young, N.E. In: Oblivious Medians Via Online Bidding. Springer Berlin Heidelberg, Berlin, Heidelberg (2006) 311–322
6. Bokhari, S.H.: Partitioning problems in parallel, pipeline, and distributed computing. IEEE Trans. Comput. **37**(1) (January 1988) 48–57
7. Hansen, P., Lih, K.W.: Improved algorithms for partitioning problems in parallel, pipelined, and distributed computing. IEEE Trans. Comput. (1992)
8. Khanna, S., Muthukrishnan, S., Skiena, S.: Efficient array partitioning. In: ICALP. Volume 1256., Springer Berlin Heidelberg (1997) 616–626
9. Miguet, S., Pierson, J.: Heuristics for 1D rectilinear partitioning as a low cost and high quality answer to dynamic load balancing. In: HPCN Europe'97. (1997) 550–564
10. Konrad, C.: Streaming partitioning of sequences and trees. In: 19th International Conference on Database Theory, ICDT 2016, Bordeaux, France, March 15-18, 2016. (2016) 13:1–13:18
11. Konrad, C., Tonoyan, T.: Preemptive online partitioning of sequences. CoRR **abs/1702.06099** (2017)