2022

Online Algorithms with Advice for the �-search Problem

Jhoirene B. Clemente

Henry N. Adorna

Proceso L. Fernandez Jr

Philippine Journal of Science
151 (4): 1321-1331, August 2022
ISSN 0031 - 7683
Date Received: 16 Nov 2021

Online Algorithms with Advice for the k-search Problem

Jhoirene B. Clemente[1]*, **Henry N. Adorna**[1], and **Proceso L. Fernandez Jr.**[2]

[1]Department of Computer Science, University of the Philippines Diliman, Quezon City, NCR 1101 Philippines
[2]Department of Information Systems and Computer Science, Ateneo de Manila University, Quezon City, NCR 1108 Philippines

In the online search problem, a seller seeks to find the maximum price from a sequence of prices $p_1, p_2, \ldots, p_n$ that is revealed in a piece-wise manner. The bound for all prices is well known in advance with $m \leq p_i \leq M$. In the online k-search problem, the seller seeks to find the k maximum out of the n prices. In this paper, we present a tight bound of $\log_2\binom{n}{k}$ on the advice complexity of optimal online algorithms for online k-search. We also provide online algorithms with advice that use less than the required number of bits and compute the performance guarantee. Although it is natural to expect improvement due to the additional power of advice, we are interested to identify the relationship of additional information with respect to the improvement. We show that with 1 bit of advice, we can already surpass the quality of the best possible deterministic algorithm for online 2-search. We also provide a set of online algorithms, ALG$_i$, that utilizes $\log_2\binom{n}{i}$ advice bits with a competitive ratio of $\frac{kM}{iM + \sqrt[k-i+1]{(k-i)^{(k-i)}m}}$. We show that increasing the amount of advice improves the solution quality of the algorithm. Moreover, we compare the power of advice and randomization. We show that for some identified minimum number of advice bits, the lower bound on the competitive ratio of online algorithms with advice is better than any deterministic and randomized algorithm for online k-search.

Keywords: advice complexity, competitive analysis, online algorithms, online search

INTRODUCTION

Online problems are computational problems with incomplete information about the input. In this scenario, the input is given piece-wise and upon receiving the input, an algorithm must provide a piece of the solution. Algorithms solving these types of problems are called online algorithms. The lack of information about the input instance makes it difficult for any algorithm to achieve optimality, even for problems in P in the offline setting. The term "offline" is used when the whole input sequence is known in advance. Examples of well-known online problems are the secretary problem (Gardner 1995), ski-rental problem (Karlin *et al.* 1994), and time series search problem (El-Yaniv *et al.* 2001).

Competitive analysis was introduced by Sleator and Tarjan (1985) to analyze the solution quality of online algorithms. The measure used in the analysis is called the competitive ratio, which can be obtained by comparing the profit of the online algorithm to one of an optimal offline solution. Note that it is generally not possible for an online algorithm to compute the optimal offline solution in advance, because parts of the output have to be specified before the whole input is known. It is merely taken into account to analyze

*Corresponding author: jbclemente@up.edu.ph the profit that can hypothetically be obtained if the whole input is known in advance. The competitive ratio of an online algorithm is formally defined as follows.

Definition 1 (competitive ratio): Let Π be an online maximization problem, let ALG be an online algorithm for Π, and let c ≥ 1. ALG has a competitive ratio of c or c-competitive if, for every instance I of Π, we have:

$$c \cdot profit(ALG(I)) \leq profit\ (OPT(I))$$

where $profit(ALG)$ is the profit of ALG on input I, $profit\ (OPT(I))$ denotes the optimal offline profit, and OPT is an optimal offline algorithm for Π.

A complementary tool in analyzing online algorithms is the advice complexity. Dobrev *et al.* (2008) initially introduced the idea and this was later on revised by Böckenhauer *et al.* (2009), Hromkovič *et al.* (2010), and Emek *et al.* (2011) to analyze online problems. The concept is analogous to a randomized computation, where an algorithm has access to a tape. Instead of accessing random bits, online computation with advice reads information from a trusted source through an advice tape. We can think of the advice as additional information that can help the online algorithm in making "good" decisions. The advice is measured using the number of bits required to encode such information and is allowed to be any function of the entire input.

This advice complexity of an algorithm is used to measure the least amount of information necessary to be optimal or to achieve a certain competitive ratio. The advice complexity of an online problem is defined as follows.

Definition 2 (advice complexity): Let $x_1, \ldots, x_n$ be the input for an online problem Π. An online algorithm with advice, ALG, for Π computes the output sequence $y_1, \ldots, y_n$, where y_i is allowed to depend on $x_1, \ldots, x_i$ as well as on an advice string ϕ. The advice, ϕ, is written in binary on an infinite tape and is allowed to depend on the request sequence $x_1, \ldots, x_n$. The advice complexity of ALG is the largest number of advice bits it reads from ϕ over all inputs of length at most n.

An upper bound on the advice complexity of a given online problem is identified by constructing an online algorithm that reads a certain length of advice with a provable competitive ratio. On the other hand, a lower bound is derived by getting the least number of advice bits that are necessary to compute an answer sequence of the desired quality.

We use the concept of partition trees from Steffen (2014) to provide the minimum amount of advice that is needed. The general idea is to describe a set of input instances for a problem where it is difficult for any online algorithm to distinguish one from the other. Thus, an online algorithm must require a minimum amount of advice bits to specify which of the instances is the case.

Computation with advice has already been applied to several online problems in the literature, including paging (Böckenhauer *et al.* 2009; Dobrev *et al.* 2008), the k-server problem (Böckenhauer *et al.* 2009; Emek *et al.* 2011; Gupta *et al.* 2013; Renault and Rosén 2011), metrical task systems (Emek *et al.* 2011), and the online knapsack problem (Böckenhauer *et al.* 2014b). Some hardness results on the advice complexity can be achieved by a special kind of reduction (Böckenhauer *et al.* 2014a; Boyar *et al.* 2017; Emek *et al.* 2011). Moreover, advice complexity has a close and non-trivial relation to randomization (Böckenhauer *et al.* 2009; Komm and Královič 2011; Mikkelsen 2015; Steffen 2014).

Online k-search is a straightforward generalization of the online search problem previously described in the study of Clemente *et al.* (2016). Online search seeks to find the maximum price p_i, from a sequence of n prices that is revealed piece-wise. The generalization seeks to find k maximum out of n prices.

To give us intuition about online k-search, consider a scenario in which a seller would like to maximize his profit by selling $k \geq 1$ units of an asset. Every day, the market provides a price p_i and the seller must decide whether or not to sell one unit of the asset for the price. The overall profit of the seller is computed as the sum of individual selling prices of the k units. A constraint is that the seller must complete all the transactions in a limited amount of time, say n days. This implies that if the seller has j remaining unsold assets and there are only j days of trading days left, the seller must sell one unit on each of the remaining trading days regardless of the price. We may refer to n as the duration of the trading period. To help with the decision-making, the seller is aware of the minimum and maximum price offering of the asset, *i.e.* $m \leq p_i \leq M$. Formally, the online k-search is defined as follows.

Definition 3 [online k-search (Lorenz et al. 2009)]: Let $\sigma = (p_1, p_2, \ldots, p_n)$, with $0 < m \leq p_i \leq M$ for all $1 \leq i \leq n$, be a sequence of prices that arrives in an online fashion. Here, M and m are upper and lower bounds on the prices, respectively. For each day i, price p_i is revealed, and the online player has to choose whether to sell on the same day or to wait for a new price on a subsequent day. Given σ and a positive integer $k < n$, the player must decide to sell on k days of the whole duration. If the player still has j units of assets remaining immediately after day $n - j$, he must sell on the last j days of the trading period. The player's goal is to maximize the sum of the prices, *i.e.* its profit.

Here, the seller must decide immediately given a limited portion of the input, *i.e.* with the knowledge of the current price offering and all the previous prices provided by the market, the seller must decide under the uncertainty of the future prices. The decisions made by the seller are irrevocable, *i.e.* once he decides to sell, a deal is immediately done, and the seller can no longer undo any previous transaction. The assumptions in our previous of the advice complexity of online search in the study of Clemente *et al.* (2016) carry over to online k-search in the study. We assume that the price offer between a defined range [m,M] is known to the online seller before the start of the computation, and the trading duration, denoted by n, may or not be known to the online seller. In this study, we are interested in providing and analyzing online algorithms that may be used by the seller for decision-making. We also may refer to these algorithms as strategies of the seller.

RELATED WORK

This work is a continuation of our previous result (Clemente *et al.* 2016), where we study the advice complexity of the basic case of online k-search for $k = 1$. In our previous result, we provided matching upper and lower bounds of advice needed for optimality and c-competitiveness of any online algorithm for online search. Moreover, we have shown that advice is more powerful than randomization for online search, in the sense that for some constant amount of advice, we can already have an improvement over the best-randomized algorithm for online search, and as we approach $O(\log_2 n)$ bits of advice, we can have a competitive ratio that approaches 1.

Before presenting our online algorithms with advice for online k-search, we first briefly mention some preliminary work on the competitiveness of online k-search. El-Yaniv *et al.* (2001) presented a strategy called the reservation price policy. In this strategy, a seller decides to accept a certain price p_i if it is greater than or equal to some precomputed reservation price r, where $m \le r \le M$. Let us denote by $ALG[r]$ the online algorithm that follows this policy.

The existing optimal deterministic strategy for online k-search of Lorenz *et al.* (2009) is the generalization of the reservation price policy strategy from (El-Yaniv *et al.* 2001). Here we extend the notation of $ALG[r]$ to denote the algorithm that uses a sequence of reservation prices. Let $ALG[(r_1, r_2, \dots, r_k)]$ denote the online algorithm from the study of Lorenz *et al.* (2009), where $r_1, \dots, r_k$ are the reservation prices such that $r_i < r_j$ if $i < j$. In the online algorithm, the value of each r_i is precomputed based on the known parameters m and M before the start of the computation. The algorithm uses the reservation prices sequentially. It accepts the first price $p_i \ge r_1$, then another $p_j \ge r_2$, and so on. If the player still has l units to sell with only l trading days left, the algorithm is required to sell on the remaining l trading days. The optimal deterministic strategy for online k-search computes for each reservation price using:

$$r_i = m\left[1 + (c-1)\left(1 + \frac{c}{k}\right)^{i-1}\right], \qquad (1)$$

where c is the competitive ratio of $ALG[(r_1, r_2, \dots, r_k)]$ή. The equation is derived from the following worst-case scenarios:

$$\begin{aligned} c &= \frac{kr_1}{km} \\ &= \frac{kr_2}{r_1 + (k-1)m} \\ &= \frac{kr_3}{r_1 + r_2 + (k-2)m} \\ &= \frac{kM}{\sum_{i=1}^{k} r_i} \end{aligned}$$

Lorenz *et al.* (2009) presented $k+1$ possible cases, where each is characterized by the worst possible input σ_i:

$$\sigma_i = r_1, \dots, r_i, \underbrace{r_{i+1} - \varepsilon, \dots, r_{i+1} - \varepsilon}_{k}, \underbrace{m, \dots, m}_{n-(k+i)}.$$

Here, a fixed $0 < \varepsilon < 1$ prevents the deterministic strategy from obtaining the k maximum prices, which is almost $k \cdot r_{i+1}$, and the strategy is instead forced to take the last $k - i$ minimum prices.

The competitive ratio c from Equation 1 has two possible approximations depending on the values of the input parameters $\frac{M}{m}$ and k. For input instances with large fluctuation ratio $\frac{M}{m}$, the competitive ratio is approximately $\sqrt[k+1]{k^k \cdot \left(\frac{M}{m}\right)}$. On the other hand, for large c, the competitive ratio of $1 + W\left(\frac{\left(\frac{M}{m}\right) - 1}{e}\right)$ where W is W-function that is the inverse of $f(w) = w\exp(w)$. It is well-known that for suffiently large x, the function $W(x)$ behaves like $ln(x)$.

OPTIMAL ALGORITHMS WITH ADVICE FOR ONLINE k-SEARCH

In this section, we discuss the amount of advice needed by online algorithms to solve online k-search optimally.

Note that before the start of the trading period, the oracle, which provides the advice, and the online player have already engaged in an agreement as to how to interpret an encoded information ϕ on the advice tape. The whole content of the tape can be accessed only once during the start of the trading period. The answer sequence of the online player is expected to agree with the advice given by the oracle. The descriptions of the online algorithms with the advice presented in this study focus on the content of the advice tape ϕ and how the online player can use the advice to provide answers to the input sequence that is iteratively revealed.

For $k = 1$, we have an optimal online algorithm with advice that specifies the optimal trading days using $O(\log_2 n)$ bits of advice. If $k = 1$, a corresponding lower bound result from the study of Clemente *et al.* (2016) provides a tight bound for online k-search. We present the previous result in the following lemma.

Lemma 1 (Clemente et al. 2016): At least $\log_2 n$ bits of advice are necessary to obtain an optimal solution for the online search. This holds even if n is known to the algorithm.

Theorem 1: At least $\log_2 n$ bits of advice are necessary to obtain an optimal solution for online k-search, where $1 \leq k \leq n$. This holds even if n is known to the algorithm.

Proof: The proof directly follows from Lemma 1. **Q.E.D**

In the following theorem, we present a tight bound in the advice complexity of online k-search. The proof consists of a matching upper and lower bound. In the upper bound result, we present an optimal online algorithm. In the lower bound result, we used the concept of partition trees to show the minimum amount of advice needed by any online algorithm to be optimal for online k-search.

Theorem 2: Every optimal online algorithm with advice for online k-search needs at least $\log_2\binom{n}{k}$ bits of advice and this bound is tight.

Proof: To prove that the bound is tight, we need to show that the upper bound and the lower bound hold. The proof follows from the proofs of Lemmas 2 and 3 for the upper bound and lower bound results, respectively.

Lemma 2: There exists an optimal online algorithm for online k-search that uses $\log_2\binom{n}{k}$ bits of advice.

Proof: First, we prove the upper bound. We show that there exists an optimal online algorithm with advice for the online k-search problem that uses $\log_2\binom{n}{k}$ bits of advice.

We can represent the answer sequence of an online algorithm for online k-search as a sequence of the seller's decision. Let $o \in \{0,1\}^n$ be the binary representation of the answer sequence, where $o[i] = 1$ or $o[i] = 0$ corresponds to the ith day as a trading day or a non-trading day, respectively. A feasible answer sequence contains k trading and $n - k$ non-trading days. Given the parameters n and k for online k-search, one can enumerate the set containing all unique possible optimal solutions for any instance. One simple enumeration is by visiting all 2^n possible binary representations of the answer sequence and eliminating all non-feasible solutions. The number of such unique optimal solutions is $\binom{n}{k}$.

An optimal online algorithm with advice for online k-search reads the advice tape containing the encoding that specifies the index of the feasible solution in the above enumeration. The oracle can encode such information using $\log_2\binom{n}{k}$ bits.

Q.E.D.

Lemma 3: Every optimal online algorithm with advice for online k-search needs at least $\log_2\binom{n}{k}$ bits of advice.

Proof: Here, we need to show that there exists a set $\mathcal{S}_{n,k}$ of the input sequence to an online algorithm such that no online algorithm with advice will solve all instances in $\mathcal{S}_{n,k}$ optimally without needing $\log_2\binom{n}{k}$ bits of advice, for any n and $0 < k < n$.

Let $\mathcal{S}_{n,k} = \{\sigma_1, \sigma_2, \dots, \sigma\binom{n}{k}\}$ be a set of instances for online k-search. Let the corresponding set of the optimal solution for $\mathcal{S}_{n,k}$ be $O_{n,k} = \{o_1, o_2, \dots, o\binom{n}{k}\}$. Each input instance σ_i is a sequence of n prices, *i.e.* $\sigma_i = p_{i,1}, p_{i,2}, \dots, p_{i,n}$ where $m \leq p_{i,j} \leq M$ for every $1 \leq j \leq n$. The corresponding optimal solution o_i is a binary sequence of length n. The output sequence encodes the optimal seller's decision, *i.e.* 1 if it is a selling day and 0, otherwise.

Here, we show how to construct the set $\mathcal{S}_{n,k}$ for any feasible input parameters n and k.

1. Given parameters n and k, generate an ordered binary tree $T_{n,k}$ using the following recursive definition:

$$T_{n,k} = merge(T_{n-1,k-1}, T_{n-1,k})$$

The merge operation creates a tree by adding a root node and using $T_{n-1,k-1}$ and $T_{n-1,k}$ as the left and right subtrees, respectively.

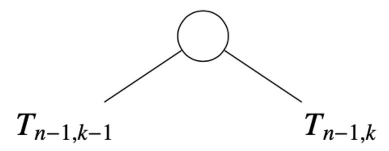
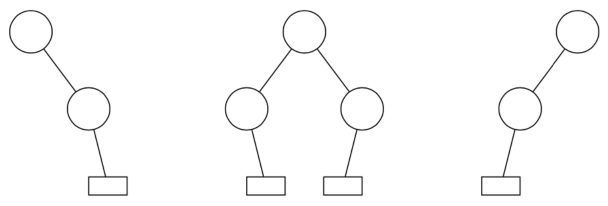


Figure 1. $T_{n,k}$.

The intention here is to let the left subtree correspond to the case that the seller decides to sell on the first day, and thus needs to later decide on which of the remaining $n-1$ days to sell the remaining $k-1$ assets. On the other hand, the right subtree corresponds to the case where the seller does not sell on the first day.

The base cases for the recursive definition are $T_{2,0}$, $T_{2,1}$ and $T_{2,2}$, as shown in the following figure:



(a) $T_{2,0}$ (b) $T_{2,1}$ (c) $T_{2,2}$

Figure 2. Three possible trees for n = 2, *i.e.* k = 0, 1, 2. The trees are $T_{2,0}$, $T_{2,1}$, and $T_{2,2}$. For illustration purposes, internal nodes are represented by circle nodes, and leaves are represented by rectangle nodes.

In $T_{n,k}$, the total number of internal nodes is $N_{n,k}$, which is computed by the following recursive definition:

$$N_{n,k} = N_{n-1,k-1} + N_{n-1,k} + 1$$

with base cases, $N_{n,0} = N_{n,n} = n$ and $N_{n,1} = n + 1$.

2. Given the bounds on the prices, *i.e.* m and M, compute for p_1, p_2, …, $p_{N_{n,k}}$ using the following definition:

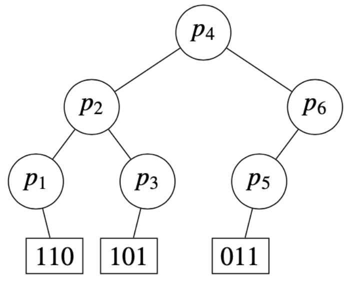
$$p_i = m + (i-1) \cdot (M-m)/(N_{n,k} - 1).$$

This construction will ensure that $m \leq p_i \leq M$ and $p_i < p_j$, for any $i,j \in \{1, \dots, N_{n,k}\}$, where $i < j$.

3. Label the internal nodes of $T_{n,k}$ using p_1, … $p_{N_{n,k}}$ in an inorder fashion. Each leaf is reachable from the root through a sequence of left and right traversals. The sequence of traversal is represented in the label of length n, The encoding is 1 if it is a left traversal and 0 if it is a right traversal. Since we followed the inorder fashion for labeling the nodes, we are guaranteed that for each node p_i, every node in the left subtree is less than p_i and every node in the right subtree is greater than p_i.

4. Generate the set $\mathcal{S}_{n,k}$ by enumerating all possible paths from the root to a leaf. The sequence of prices is the sequence of internal node labels of $T_{n,k}$. The optimal solution for σ_i is encoded in the leaf label, *i.e.* 1 if it is a selling day and 0 otherwise.

We illustrate the construction of $\mathcal{S}_{3,2}$ from $T_{3,2}$ using the four steps as described above.



(a) $T_{3,2}$

i	1	2	3
σ_1	p_4	p_2	p_1
σ_2	p_4	p_2	p_3
σ_3	p_4	p_6	p_5

(b) $\mathcal{S}_{3,2}$

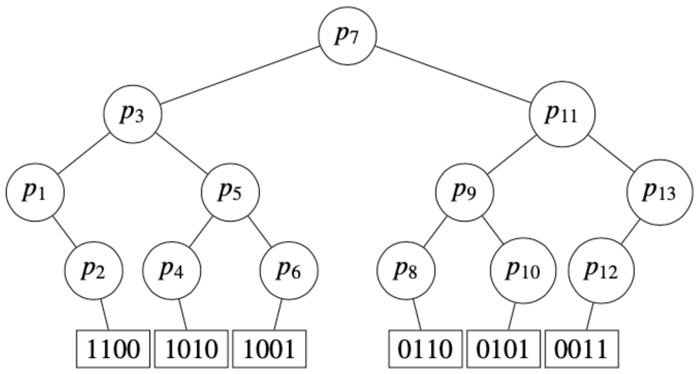
Figure 3. The ordered tree $T_{2,3}$ (a) for creating the set of input instances for online *k*-search, where *n* = 3 and *k* = 2. The leaf label encodes the optimal solution for each input instance in $\mathcal{S}_{3,2}$. (b) This table consists of the list of input instances in $\mathcal{S}_{3,2} = \{\sigma_1, \sigma_2, \sigma_3\}$. Each row in the table is an instance and the index *i* represents a trading day from 1 to *n*. The optimal selling days are highlighted in the table.

Note that, for any feasible value for the parameters n, k, M, and m, we can generate a set $\mathcal{S}_{n,k}$ with $\binom{n}{k}$ unique instances. The construction of $T_{n,k}$ assures that any pair σ_i and $\sigma_j \in \mathcal{S}_{n,k}$ has a maximal common prefix at day l, where $1 \leq l \leq n-1$, and where the optimal solution for day $l+1$ in σ_i is to sell, whereas the optimal solution for σ_j is to wait. Thus, for any pair σ_i and σ_j, it is impossible for any online algorithm to distinguish the two instances and produce an optimal solution, not unless an *a priori* advice is given to distinguish all the instances of $\mathcal{S}_{n,k}$.

Since $|\mathcal{S}_{n,k}| = \binom{n}{k}$ for any given n and k, we need at least $\log_2\binom{n}{k}$ bits of advice for any online algorithm to solve all input instances of online k-search optimally.

Q.E.D

In Figure 4, we illustrate how to create a set of instances for online k-search with $n = 4$ and $k = 2$. The set $\mathcal{S}_{4,2}$ consists of $\binom{4}{2}$ possible input instances, where each has a different optimal solution. For instance, let us consider σ_1 and σ_2, at day 2, both of them has p_3. For σ_1 selling at price p_3 is an optimal answer, whereas it is not the case for σ_2.



(a) $T(\mathcal{S}_{4,2})$

i	1	2	3	4
σ_1	p_7	p_3	p_1	p_2
σ_2	p_7	p_3	p_5	p_4
σ_3	p_7	p_2	p_5	p_6
σ_4	p_7	p_{11}	p_9	p_8
σ_5	p_7	p_{11}	p_9	p_{10}
σ_6	p_7	p_{11}	p_{13}	p_{12}

(b) $\mathcal{S}_{4,2}$

Figure 4. Each leaf in the tree is an instance in $\sigma_i \in \mathcal{S}_{4,2}$. The sequence of prices is obtained by enumerating the labels from a root to a leaf. The corresponding values (b) of each instance in $\mathcal{S}_{4,2} = \{\sigma_1, \sigma_2, \ldots \sigma_6\}$ for the whole trading day duration $1 \le i \le n$.

ONLINE ALGORITHMS WITH ADVICE: 1 BIT OF ADVICE FOR ONLINE 2-SEARCH

We explore the power of a single bit of advice in improving the competitive ratio of online algorithms for online k-search. First, we present an online algorithm with 1 bit of advice for online 2-search. Then, we show that we can generalize the algorithm for online k-search. The optimal deterministic reservation policy from the study of Lorenz *et al.* (2009) provides a tight bound on the competitive ratio of deterministic algorithms for online k-search. From Theorem 1 of Lorenz *et al.* (2009), we have the following corollary.

Corollary 1: For sufficiently large M/m and with 1 bit of advice available, there exists a deterministic online algorithm with a competitive ratio that is approximately equal to $\sqrt[3]{4 \cdot \left(\frac{M}{m}\right)}$ for online 2-search, and there exists no deterministic algorithm with a smaller competitive ratio.

The optimal deterministic strategy of Lorenz *et al.* (2009) uses two reservation prices r_1 and r_2, whose values are the solution of the following equation:

$$\frac{2r_1}{2m} = \frac{2r_2}{r_1 + m} = \frac{2M}{r_1 + r_2}$$

The deterministic approach has an exact competitive ratio of $\frac{\frac{m^2}{t} + t - m}{m}$, where:

$$t = \sqrt[3]{-m^3 + 2m^2M + 2\sqrt{m^4M^2 - m^5M}}.$$

Suppose we extend the idea of Lorenz *et al.* (2009) to provide us with an online strategy with advice for online 2-search. With 1 bit of advice, we can have two deterministic algorithms each with a set of 2 reservation prices. Let the two algorithms be $ALG[(r_{1,1}, r_{1,2})]$ and $ALG[(r_{2,1}, r_{2,2})]$ where $m < r_{1,1} < r_{1,2} < r_{2,1} < r_{2,2} < M$. Analogous to the deterministic approach, the set of reservation prices solves the following equation:

$$\frac{2r_{1,1}}{2m} = \frac{2r_{1,2}}{r_{1,1} + m} = \frac{2r_{2,1}}{r_{1,1} + r_{1,2}} = \frac{2r_{2,2}}{r_{2,1} + m} = \frac{2M}{r_{2,1} + r_{2,2}} \quad (2)$$

Let us present the intuition behind the set of equations presented. Suppose we have p_{max} as the maximum price in the given input sequence σ. We have two cases depending on the value of p_{max}. The first case is when we have $p_{max} < r_{2,1}$. Here, we have $r_{1,1}$ and $r_{1,2}$ as reservation prices, since they are between the minimum price m and $r_{2,1}$.

The second case is when we have $p_{max} < r_{2,1}$. With this condition, we are guaranteed to obtain a price of at least $r_{2,1}$. Therefore, we use $r_{2,1}$ and $r_{2,2}$ as reservation prices since they both lie between $r_{2,1}$ and M. The solution to Equation 2 would provide an optimal competitive ratio for the given approach. However, such a solution is difficult to determine. For the purpose of establishing a competitive ratio, one strategy is to fix $r_{2,1}$ to a certain value that lies between the minimum price m and the maximum price M. Say we have $r_{2,1} = \sqrt{Mm}$, as in the case of the deterministic reservation price for 1-search from (El-Yaniv *et al.* 2001), then we can compute a different competitive ratio for each deterministic algorithm. For the first algorithm, $ALG[(r_{1,1}, r_{1,2})]$, we have the following computation of the competitive ratio c_1:

$$c_1 = \frac{2r_{1,1}}{2m} = \frac{2r_{1,2}}{r_{1,1}+m} = \frac{2\sqrt{Mm}}{r_{1,1}+r_{1,2}} \leq \sqrt[3]{4\sqrt{\frac{M}{m}}}$$

On the other hand, we have $ALG[(\sqrt{Mm}, r_{2,2})]$ with competitive ratio c_2, which can be computed using the following relation:

$$c_2 = \frac{2r_{2,2}}{\sqrt{Mm}+m} = \frac{2M}{\sqrt{Mm}+r_{2,2}}$$

Solving $r_{2,2}$ in terms of m and M, we have:

$$r_{2,2} = \frac{1}{2}\left(\sqrt{4\sqrt{M^3m}+5Mm} - \sqrt{Mm}\right).$$

Using the computed value of $r_{2,2}$, we have the following computation for the competitive ratio:

$$\begin{aligned}
c_2 &= \frac{\sqrt{4\sqrt{M^3m}+5Mm} - \sqrt{Mm}}{\sqrt{Mm}+m} \\
&\leq \frac{\sqrt{4\sqrt{M^3m}+5Mm} - \sqrt{Mm}}{\sqrt{Mm}} \\
&\leq \frac{\sqrt{Mm\,(4\sqrt{\frac{M}{m}}+5)} - \sqrt{Mm}}{\sqrt{Mm}} \\
&\leq \frac{\sqrt{Mm}\sqrt{4\sqrt{\frac{M}{m}}+5} - \sqrt{Mm}}{\sqrt{Mm}} \\
&\leq \sqrt{4\sqrt{\frac{M}{m}}+5} - 1 \\
&< \sqrt{4\sqrt{\frac{M}{m}}+4} \\
&< \sqrt{4\,(\sqrt{\frac{M}{m}}+1)} \\
&< 2\sqrt{\sqrt{\frac{M}{m}}+1}
\end{aligned}$$

In summary, if we fix $r_{2,1} = \sqrt{Mm}$, we have a competitive ratio of $\sqrt[3]{4\sqrt{\frac{M}{m}}}$ for the first case and we have $2\sqrt{\sqrt{\frac{M}{m}}+1}$ for the second case. The second ratio is clearly greater than the first one, and so we adopt it as the competitive ratio for the proposed approach in order to cover all instances.

Following the above computations, we have the following theorem for online 2-search.

Theorem 3: There exists a $2\sqrt{\sqrt{\frac{M}{m}}+1}$-competitive online algorithm with 1 bit of advice for online 2-search.

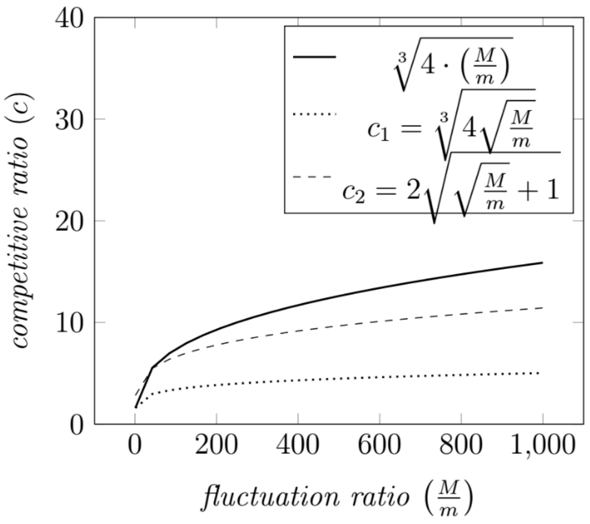


Figure 5. Comparing the competitive ratio of the existing optimal deterministic algorithm from the study of Lorenz *et al.* (2009) and the competitive ratio of the two cases (c_1 and c_2) for the online algorithm with *1* bit of advice for online 2-search as the fluctuation ratio $\frac{M}{m}$ increases.

Figure 5 shows the comparison of the competitive ratio of the optimal deterministic without advice from the study of Lorenz *et al.* (2009) and our two competitive ratios c_1 and c_2 as we increase the parameter $\frac{M}{m}$. Overall, we take c_2 as the algorithm's competitive ratio. The optimal competitive ratio for the online 2-search with 1 bit of advice will balance the two obtained ratios for the two cases, and the ratio lies between c_1 and c_2.

ONLINE ALGORITHM WITH ADVICE FOR ONLINE k-SEARCH

In this section, we present a set of online algorithms with advice for online k-search. With less than the optimal number of bits of advice, the oracle can specify a portion of the optimal solution prior to the computation. The remaining parts of the solution will be computed according to the optimal deterministic approach of Lorenz *et al.* (2009). Using $\log_2\binom{n}{i}$ bits of advice, the oracle can specify i out of k parts of the optimal solution to ALG_i. Note that the number of advice bits for ALG_i declines as the parameter k approaches n.

Theorem 4: For any integer $0 < i < k$, there exists an online algorithm ALG_i for online k-search that uses $\log_2\binom{n}{i}$ bits of advice to secure the top i of k optimal prices. For $\frac{M}{m} \to \infty$, the competitive ratio of this algorithm is approximately equal to:

$$\frac{kM}{iM + \sqrt[k-i+1]{(k-i)M^{(k-i)}m}}.$$

Proof: Let ALG_i be an online algorithm with advice for online k-search. The oracle can specify the location of i highest prices by specifying an element from the set of all possible i combinations of n. The set of all possible combinations has a cardinality of $\binom{n}{i}$. The oracle can encode such information by specifying an index of the element from the set using $\log_2\binom{n}{i}$ bits.

After reading the advice, ALG_i will sell on the identified i days and choose the remaining $(k-i)$ prices using the optimal deterministic algorithm $ALG[(r_1, r_2, \dots, r_{k-i})]$. Using the advice, ALG_i is guaranteed to be optimal in the top i out of the k highest prices. Since we use $ALG[(r_1, r_2, \dots, r_{k-i})]$ to provide the remaining parts of the solution, the solution quality of the partial solution is within a factor of the known competitive ratio of the algorithm. The profit of the online algorithm with advice is equivalent to the sum of the i highest prices and the profit of the partial solution obtained by using $ALG[(r_1, r_2, \dots, r_{k-i})]$. If $ALG[(r_1, r_2, \dots, r_{k-i})]$ is σ-competitive, the lowest profit of the algorithm is $profit(OPT)/\,\sigma \leq ((k-i)\,M)\,/\,\sigma$.

Thus, the competitive ratio of ALG_i is expressed as $(M/m)/\left(iM + \frac{(k-i)M}{\sigma}\right)$, where $\sigma = \sqrt[k-i+1]{(k-i)^{(k-i)}(M/m)}$. By performing the following algebraic manipulation, we obtain the following competitive ratio:

$$\begin{aligned} c &= \frac{kM}{iM + \frac{(k-i)M}{\sigma}} \\ &= \frac{kM}{iM + \frac{(k-i)M}{\sqrt[k-i+1]{(k-i)^{(k-i)}(M/m)}}} \\ &= \frac{kM}{iM + \frac{(k-i)M}{\left((k-i)^{(k-i)}(M/m)\right)^{1/(k-i+1)}}} \end{aligned}$$

To simplify the computation, let $d = (k-i)$:

$$\begin{aligned} c &= \frac{kM}{iM + \frac{(k-i)M}{\left((k-i)^{(k-i)}(M/m)\right)^{1/(k-i+1)}}} \\ &= \frac{kM}{iM + \frac{dM}{\left(d^d(M/m)\right)^{1/(d+1)}}} \\ &= \frac{kM}{iM + d^{(d+1-d)/(d+1)}M^{(d+1-1)/(d+1)}m^{1/(d+1)}} \\ &= \frac{kM}{iM + d^{1/(d+1)}M^{d/(d+1)}m^{1/(d+1)}} \\ &= \frac{kM}{iM + \sqrt[d+1]{dM^d m}} \\ &= \frac{kM}{iM + \sqrt[k-i+1]{(k-i)M^{(k-i)}m}} \end{aligned}$$

Q.E.D

Using the expression above for the competitive ratio, we have $(kM)/\left((k-1)M + \sqrt{Mm}\right)$ for ALG_{k-1}. If the oracle can specify the location of the $k-1$ highest prices, we are left with a portion of the optimal solution that is as good as the best deterministic online search, *i.e.* $k = 1$. On the contrary, if the oracle can only specify the location of the highest price, we have a relatively poorer competitive ratio of $kM/\left(M + \sqrt[k]{(k-1)mM^{(k-1)}}\right)$. In summary, ALG_i is a collection of online algorithms for online k-search that provides higher solution quality with higher advice requirements. The computed competitive ratio of ALG_i coincides with the optimal deterministic competitive ratio of Lorenz if i is set to 0, *i.e.* no advice is given. Moreover, if all the k highest prices are provided by the oracle ALG_k is optimal with $c = 1$.

ADVICE COMPLEXITY

Upper Bound Results

The use of randomization for improving the competitive ratio of online algorithms has been well established in the literature. Online computation has an interesting relationship to randomized computation. We highlight one observation where it is possible to transform any existing randomized algorithm into an online algorithm by simply choosing the best random string as the advice. Formally, we have the following observation from (Komm 2012).

Observation 1 (Komm 2012): If there exists a randomized online algorithm RAND that is c-competitive in expectation and that uses b random bits, there also exists a c-competitive online algorithm with advice that uses b bits of advice.

From the above observation, we use a randomized online algorithm RAND from the study of Lorenz *et al.* (2009) to provide a corresponding online algorithm with advice.

Theorem 5: There exists an online algorithm with advice for online k-search with a competitive ratio of $O\big(2\ln(M/m)\big)$ using $b = \log_2\big(\log_a(M/m)\big)$ bits of advice.

Proof: Let RAND be a randomized online algorithm for online k-search. RAND implements $ALG[ma^r]$, for some r chosen uniformly at random from $\{0, \ldots, l-1\}$. Here, parameter $l = \log_a(M/m)$, for some $1 < a < M/m$. For $M/m > 3$ and $a < 3/2$, algorithm RAND has an expected competitive ratio of $2\ln(M/m)$, as shown in Lemma 8 by Lorenz *et al.* (2009).

An online algorithm for online k-search simulates $ALG[ma^r]$ given an advice $r \in \{0, \ldots, l-1\}$. The oracle gives the best value for the parameter r for each possible input sequence. Since RAND has a competitive ratio of $2\ln(M/m)$ in expectation, the online algorithm has a competitive ratio of at most $2\ln(M/m)$. The number of advice bits follows from the encoding of the random parameter r, which can be encoded using $b = \log_2\big(\log_a(M/m)\big)$ bits of advice.

Q.E.D

Lower Bound Results

It was shown by Clemente *et al.* (2016) that there exists an online algorithm with advice that can outperform the best deterministic and the best-randomized algorithm for online search.

Lemma 4 (Clemente et al. 2016): Let ALG be an algorithm with advice for online search, which reads $b < \log_2(n)$ bits of advice. The competitive ratio of ALG is at least $(M/m)^{\frac{1}{2^b+1}}$.

We argue in this section that we can also have the same lower bound results for online k-search.

Theorem 6: Every online algorithm for Online k-Search for $0 < k < n$, which reads $b < \log_2 n$ bits of advice have a competitive ratio of at least $(M/m)^{\frac{1}{2^b+1}}$.

Proof: The proof follows directly from Lemma 4.

Q.E.D

ADVICE AND RANDOMIZATION

Since there is a close and non-trivial relationship between advice complexity and randomization (Barhum *et al.* 2014; Komm and Královič 2011), we compare the known lower bound results from the study of Lorenz *et al.* (2009) in Figure 6.

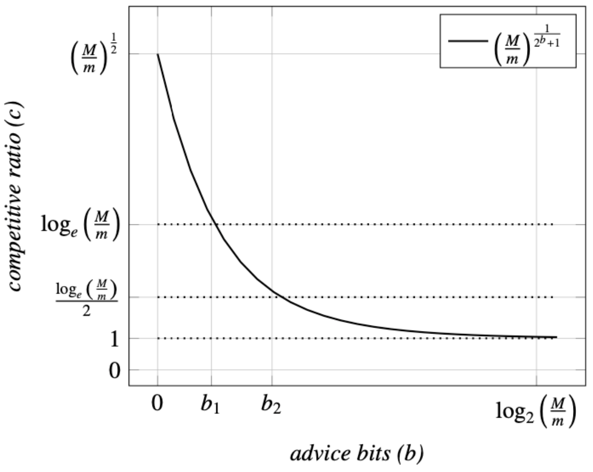


Figure 6. In this graph, we compare the lower bound results for any deterministic $log_e(M/m)$ randomized ($log_e(M/m)/2$ in expectation) and online algorithm with advice $((M/m)^{2b+1})$ for online k-search.

It was shown that no deterministic and randomized algorithm for online k-search will have a better competitive ratio of $\ln(M/m)$ and $\frac{\ln(M/m)}{2}$ (in expectation), respectively. In Figure 6, we show that for a number of advice bits greater than:

$$b_1 = \frac{\ln\left(\frac{ln(M/m)}{\ln\big(\ln(M/m)\big) - 1}\right)}{\ln(2)}$$

and:

$$b_2 = \frac{\ln\left(\dfrac{\ln(M/m)}{\ln\left(\dfrac{\ln(M/m)}{2}\right) - 1}\right)}{\ln(2)},$$

we can outperform the competitive ratio of the best deterministic and randomized algorithm for online k-search, respectively. Moreover, since the lower bound result of advice can also be used to provide a lower bound for randomized algorithms (Komm and Královič 2011), we can also have the following corollary from Theorem 6.

Corollary 2: No randomized online algorithm using b random bits will have an expected competitive ratio better than $(M/m)^{\frac{1}{2^b+1}}$.

CONCLUSION

We showed that there exists an optimal online algorithm using $log_2\binom{n}{k}$ for online k-search in Theorem 2 and that this bound is tight. We show that any optimal online algorithm needs at least $log_2\binom{n}{k}$ bits of advice. The proof in Theorem 2 uses the concept of partition trees where the lower bound on the advice is computed by counting the total number of hard instances for online k-search. The lower bound results in this paper for online k-search agrees with our results on the advice complexity for online k-search with $k = 1$, *i.e.* we need at least $O(\log_2 n)$ bits of advice to solve the problem optimally.

We also study the power of 1 bit of advice for improving the competitive ratio of the online 2-search. We presented an online algorithm for online 2-search and showed that through advice, we can outperform the lower bound of the deterministic algorithms in terms of the competitive ratio. From a competitive ratio of $\sqrt[3]{4\frac{M}{m}}$, we present an online algorithm with one bit of advice for online 2-search with a competitive ratio of $2\sqrt{\sqrt{\frac{M}{m}}+1}$. Moreover, we present a set of online algorithms ALG_i for online k-search. This set of algorithms utilizes a parameter i, where $1 \leq i \leq k$. ALG_i can read $\log_2\binom{n}{i}$ bits of advice from the oracle bearing the location of the i highest prices in the input. We showed that ALG_i has a competitive ratio that approaches *1* as $i \rightarrow k$.

A randomized algorithm is used to provide an upper-bound result for online k-search. We show that online k-search needs at least the minimum advice bits for online search to achieve certain competitiveness. From Theorem 6, any online algorithm for online k-search needs at least b bits of advice to achieve a competitive ratio of $(M/m)^{\frac{1}{2^b+1}}$. The lower bound result in Theorem 6 provides an improvement on the expected lower bound result of any randomized online algorithm for online k-search. Moreover, the lower bound result on advice in Theorem 6 was also used to provide a lower bound result for randomized algorithms in Corollary 2.

ACKNOWLEDGMENTS

This study is fully supported by the SOSE Industry 4.0 project of the Ateneo de Manila University. The authors would like to thank Dennis Komm, Christian Kudahl, and Ivy Ordanel for the initial comments and discussions on this topic.

REFERENCES

BARHUM K, BÖCKENHAUER H-J, FORIŠEK M, GEBAUER H, HROMKOVIČ J, KRUG S, SMULA J, STEFFEN B. 2014. On the power of advice and randomization for the disjoint path allocation problem. In: Proc. of the SOFSEM 2014: Theory and Practice of Computer Science. Lecture Notes in Computer Science 8327: 89–101.

BÖCKENHAUER H-J, HROMKOVIČ J, KOMM D, KRUG S, SMULA J, SPROCK A. 2014a. The string guessing problem as a method to prove lower bounds on the advice complexity. Theoretical Computer Science 554: 95–108.

BÖCKENHAUER H-J, KOMM D, KRÁLOVIČ R, ROSSMANITH P. 2014b. The online knapsack problem: advice and randomization. Theoretical Computer Science 527: 61–72.

BÖCKENHAUER H-J, KOMM D, KRÁLOVIČ R, KRÁLOVIČ R, MÖMKE T. 2009. On the advice complexity of online problems. In: Proc. of the 20th International Symposium on Algorithms and Computation. Lecture Notes in Computer Science 5878: 331–340.

BOYAR J, FAVRHOLDT LM, KUDAH C, MIKKELSEN JW. 2017. Advice complexity for a class of online problems. Theory of Computing Systems 61: 1128–1177.

CLEMENTE J, HROMKOVIČ J, KOMM D, KUDAHL C. 2016. Advice complexity of the online search problem. In: Proc. of the 27th International Workshop on

Combinatorial Algorithms. Lecture Notes in Computer Science 9843: 203–212.

DOBREV S, KRÁLOVIČ R, PARDUBSKÁ D. 2008. How much information about the future is needed? In: Proc. of the 34th Conference on Current Trends in Theory and Practice of Computer Science. Lecture Notes in Computer Science 4910: 247–258.

EL-YANIV R, FIAT A, KARP RM, TURPIN G. 2001. Optimal Search and One-way Trading Online Algorithms. Algorithmica 30(1): 010–139.

EMEK Y, FRAIGNIAUD P, KORMAN A, ROSÉN A. 2011. Online computation with advice. Theoretical Computer Science 412(24): 2642–2656.

GARDNER M. 1995. New mathematical diversions. Scientific American, Vol. 35.

GUPTA S, KAMALI S, LÓPEZ-ORTIZ A. 2013. On advice complexity of the *k*-server problem under sparse metrics. In: Proc. of the 20th International Colloquium on Structural Information and Communication Complexity. Lecture Notes in Computer Science 8179: 55–67.

HROMKOVIČ J, KRÁLOVIČ R, KRÁLOVIČ R. 2010. Information complexity of online problems. In: Proc. of the 35th International Symposium on Mathematical Foundations of Computer Science. Lecture Notes in Computer Science 6281: 24–36.

KARLIN AR, MANASSE MS, MCGEOCH-J LA, OWICKI S. 1994. Competitive Randomized Problems Algorithms for and Results. Algorithmica 11(6): 542–571.

KOMM D. 2012. Advice and Randomization in Online Computation [Ph.D. Thesis]. Swiss Federal Institute of Technology, Zürich, Switzerland (available at the ETH Library).

KOMM D, KRÁLOVIČ R. 2011. Advice complexity and barely random algorithms. RAIRO – Theoretical Informatics and Applications 45(2): 249–267.

LORENZ J, PANAGIOTOU K, STEGER A. 2009. Optimal algorithms for k-search with application in option pricing. Algorithmica 55: 311–328.

MIKKELSEN J. 2015. Randomization can be as helpful as a glimpse of the future in online computation. Leibniz International Proceedings in Informatics, Vol. 55.

RENAULT MP, ROSÉN A. 2011. On online algorithms with advice for the *k*-server problem. In: Proc. of the 9th International Workshop on Approximation and Online Algorithms. Lecture Notes in Computer Science 7164: 198–210.

SLEATOR DD, TARJAN RE. 1985. Amortized efficiency of list update and paging rules. Communications of the ACM 28(2): 202–208.

STEFFEN B. 2014. Advice Complexity of Online Graph Problems [Ph.D. Thesis]. Swiss Federal Institute of Technology, Zürich, Switzerland (available at the ETH Library).