Online Selection with Convex Costs

Xiaoqi Tan, Siyuan Yu
University of Alberta
{xiaoqi.tan, syu3}@ualberta.ca

Raouf Boutaba,
University of Waterloo
rboutaba@uwaterloo.ca

ABSTRACT

We study a novel online optimization problem, termed online selection with convex costs (OSCC). In OSCC, there is a sequence of items, each with a value that remains unknown before its arrival. At each step when there is a new arrival, we need to make an irrevocable decision in terms of whether to accept this item and take its value, or to reject it. The crux of OSCC is that we must pay for an increasing and convex cost associated with the number of items accepted, namely, it is increasingly more difficult to accommodate additional items. The goal is to develop an online algorithm that accepts/selects a subset of items, without any prior statistical knowledge of future arrival information, to maximize the social surplus, namely, the sum of the accepted items' values minus the total cost. Our main result is the development of a threshold policy that is logistically-simple and easy to implement, but has provable optimality guarantees among all deterministic online algorithms.

1. INTRODUCTION

We consider online selection with convex cost (OSCC), a novel online optimization problem that finds many interesting applications in resource allocation, scheduling, and admission control. The basic setting of OSCC is as follows: there is a set of T items, indexed by $t \in [T]$, each with a value v_t that is unknown in advance. Items arrive one at a time in an online manner (e.g., sequential arrival of jobs submitted to a cloud computing server). Upon the arrival of item $t \in [T]$, v_t is revealed, and an online decision must be made in terms of whether to accept this item, or to discard it. We are interested in developing an online algorithm that accepts/selects a subset $\mathcal{S}$ of items, without any statistical information of future arrivals, to maximize the social surplus $v(\mathcal{S}) - f(|\mathcal{S}|)$. Here, $v(\mathcal{S})$ denotes the total value of all the selected items and $f(|\mathcal{S}|)$ represents the cost of accommodating a total of $|\mathcal{S}|$ items[1].

The key challenge for solving OSCC is to balance the *value-cost tradeoff* in the presence of *incomplete future information*, namely, how to select a subset of "worthy" items, one at a time, so that these decisions turn out to be good choices in hindsight? As the key motivation behind the proposal of OSCC, such challenges appear in many real-world resource allocation problems. For example, in Internet advertising, how to decide which advertisement (each may have a different value) to display when a user visits a website? In cloud computing, when jobs have different priorities (represented by their different values), how to decide which job to admit to maximize allocation efficiency?

Conceptually, OSCC also provides a framework that unifies a variety of classical online optimization problems. E.g.:

- *Time series search* [1] [2]. In the elementary *time series search* problem [1], a player is searching for the maximum price in a sequence that is revealed one-by-one in an online manner. At each step $t = 1, 2, \cdots, T$, the player receives price quotation v_t and must decide whether to accept this price or not. The game ends once v_t is accepted and consequently, the player's payoff is v_t. In *k-max search* [2], the player is searching for k highest prices in a sequence. Intuitively, k-max search reduces to the elementary time series search problem when $k = 1$. Our OSCC formulation further generalizes k-max search by considering a convex cost f, which in this context can be interpreted as the cost of sampling price data in the time series.
- *Online knapsack problem* [3]. OSCC can also be interpreted as a special type of online knapsack problem with packing costs [3], in which the weight of each item equals 1 (i.e., unit-weight). In this context, f can be interpreted as the cost of packing additional items into the knapsack.

(Competitive Ratio) Given an arrival instance, denoted by $\mathcal{I} = \{v_1, v_2, \cdots, v_T\}$, an online algorithm must decide, based on information revealed over time in $\mathcal{I}$, whether an item should be accepted or rejected. The performance of an online algorithm is quantified by its competitive ratio (CR):

$$\alpha \triangleq \max_{\text{all possible } \mathcal{I}} \frac{\mathsf{OPT}(\mathcal{I})}{\mathsf{ALG}(\mathcal{I})}, \tag{1}$$

where $\mathsf{ALG}(\mathcal{I})$ denotes the social surplus achieved by the online algorithm, and $\mathsf{OPT}(\mathcal{I})$ is the optimal social surplus could be achieved when $\mathcal{I}$ is known a priori. By definition, α is no smaller than 1, and the closer to 1 the better.

(Assumptions) To facilitate the design of competitive online algorithms for OSCC, we make two major assumptions.

Assumption 1 (Bounded Variability)**.** *For any $t \in [T]$, $v_t \in [v_{\min}, v_{\max}]$.*

Assumption 1 argues that the value of each item is bounded within $v_{\min}$ and $v_{\max}$. For ease of exposition, we also assume that $v_{\min}$ and $v_{\max}$ are known (this assumption can be relaxed with a more complex analysis). Intuitively, $v_{\max} \geq v_{\min} > 0$ always holds.

[1]For example, if $T = 10$ and $\mathcal{S} = \{1, 5, 9\}$, then $v(\mathcal{S}) = v_1 + v_5 + v_9$ and $f(|\mathcal{S}|) = f(3)$, leading to a social surplus of $v_1 + v_5 + v_9 - f(3)$.

Assumption 2 (Convexity)**.** *$f(x)$ is convex and monotonically increasing in $x \in [0, k]$, and can be written as*

$$f(x) = \begin{cases} \text{convex and increasing} & \text{if } x \in [0,k], \\ +\infty & \text{otherwise,} \end{cases} \tag{2}$$

where k is the maximum number of items we can accept.

Assumption 2 implies that it is increasingly more difficult to accept new items. As a special case, we allow $f(x) = 0, \forall x \in [0, k]$, in this case OSCC reduces to the standard k-max search problem [2]. It is also worth mentioning that k can be any integer ranging from 1 to ∞.

(Notations) Given the cost function f, for any $m \in [k]$, we define the cost of accepting the m-th item[2] as c_m:

$$c_m \triangleq f(m) - f(m-1), \ \forall m \in [k]. \tag{3}$$

For example, $c_1 = f(1) - f(0) = f(1)$, denoting the cost of making the first selection of one item. When the cost function is linear, e.g., $f(x) = x$, then $c_1 = c_2 = \cdots = c_k = 1$, meaning that it is equally-costly to accept additional items during the entire decision-making process.

For a cost function f given by Eq. (2), we define f^* by

$$f^*(v) \triangleq \max_{m \in \{0,1,\cdots,k\}} vm - f(m), \quad v \in [0, +\infty). \tag{4}$$

The definition of f^* is a generalization of the standard Fenchel conjugate, and the interpretation is as follows: suppose all items have the same value of v, then $f^*(v)$ equals the maximum social surplus when k items can be accepted at most.

2. MAIN RESULTS

The main result established in this paper is a threshold policy, dubbed TP_λ, that solves OSCC with the best-possible CR among all deterministic online algorithms.

2.1 Threshold Policy: TP_λ

Algorithm 1: Threshold Policy (TP_λ)

```
Inputs: λ = (λ_0, λ_1, ··· , λ_k).
Initialization: m = 0 and S = ∅.
while a new item t arrives do
    if v_t − λ_m < 0 or m > k then
        Discard item t
    else
        Accept item t, i.e., S = S ∪ {t}
        m = m + 1.
    end if
end while
```

As a threshold policy, TP_λ is simple and intuitive: for any predesigned threshold $\boldsymbol{\lambda} = (\lambda_0, \lambda_1, \cdots, \lambda_k)$, TP_λ makes decisions of accepting or rejecting items based on whether their values exceed the corresponding threshold. As an online algorithm, our goal is to design a threshold $\boldsymbol{\lambda}$ so that TP_λ achieves a constant CR that is as close to 1 as possible.

Intuitively, the threshold $\boldsymbol{\lambda}$ should never be lower than $v_{\min}$ as it suffices to make TP_λ accept any item with a threshold of $v_{\min}$. Similarly, there is no need to go beyond $v_{\max}$ since a threshold of $v_{\max} + \epsilon$ is enough to reject all items for any $\epsilon \to 0^+$. Thus, $\boldsymbol{\lambda}$ should be a sequence of $k+1$ positive real numbers within $[v_{\min}, v_{\max}]$. This leads to the definition of admission thresholds as follows.

[2]Note that $m \in [k]$ is the index for items being accepted, while $t \in [T]$ is the index for all the items.

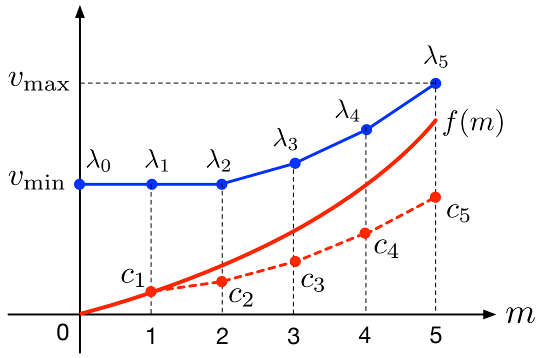

Figure 1: Illustration of an admission threshold with $\tau = 2, k = 5$, and $f(m) = m^3$.

Definition 1 (Admission Threshold)**.** *An admission threshold $\boldsymbol{\lambda} = (\lambda_0, \lambda_1, \cdots, \lambda_\tau, \cdots, \lambda_k)$ is a sequence of $k+1$ monotonically non-decreasing positive numbers such that*

$$v_{\min} = \lambda_0 = \cdots = \lambda_\tau < \lambda_{\tau+1} \le \cdots \le \lambda_{k-1} \le \lambda_k \le v_{\max},$$

where τ is an integer (to be designed) within $\{0, 1, \cdots, k-1\}$.

For any given admission threshold $\boldsymbol{\lambda}$, TP_λ always accepts the first $\tau + 1$ items since their values are guaranteed to be no less than $v_{\min}$. On the other hand, TP_λ will never accept more than k items as no item's value is higher than $v_{\max}$. For example, Fig. 1 illustrates an admission threshold with $v_{\min} = \lambda_0 = \lambda_1 = \lambda_2 < \lambda_3 < \lambda_4 < \lambda_5 = v_{\max}$. In this case, $\tau = 2$ and $k = 5$, and TP_λ always accepts the first 3 items and will never accept more than 5 items.

2.2 Uniqueness and Optimality of TP_λ

Theorem 1 below shows that there exists a *unique* admission threshold $\boldsymbol{\lambda}^*$ so that TP_{λ^*} is *optimal* among all deterministic online algorithms.

Theorem 1. *TP_{λ^*} achieves the best-possible CR of all deterministic algorithms, denoted by α^*, if and only if $\boldsymbol{\lambda}^* = \{\lambda_0^*, \lambda_1^*, \cdots, \lambda_\tau^*, \cdots, \lambda_k^*\}$ is an admission threshold such that*

- *$\lambda_0^* = \lambda_1^* = \cdots = \lambda_\tau^* = v_{\min}$ and $\lambda_k^* = v_{\max}$, where τ is the minimum integer in $\{0, 1, \cdots, k-1\}$ such that*

$$v_{\min}(\tau+1) - f(\tau+1) \ge \frac{f^*(v_{\min})}{\alpha^*}. \tag{5}$$

- *$\{\alpha^*, \lambda_{\tau+1}^*, \lambda_{\tau+2}^*, \cdots, \lambda_{k-1}^*, \lambda_k^*\}$ is the unique set of $k - \tau + 1$ positive real numbers that satisfy the system of equations:*

$$\alpha^* = \frac{f^*(\lambda_{\tau+1}^*)}{v_{\min}(\tau+1) - f(\tau+1)} = \frac{f^*(\lambda_{\tau+2}^*) - f^*(\lambda_{\tau+1}^*)}{\lambda_{\tau+1}^* - c_{\tau+2}} = \cdots = \frac{f^*(\lambda_k^*) - f^*(\lambda_{k-1}^*)}{\lambda_{k-1}^* - c_k}. \tag{6}$$

The design of τ in Eq. (5) and the system of equations characterized by Eq. (6) identify the necessary and sufficient conditions for $\boldsymbol{\lambda}^* = \{\lambda_0^*, \lambda_1^*, \cdots, \lambda_\tau^*, \cdots, \lambda_k^*\}$ that once satisfied, TP_{λ^*} achieves the optimal CR of all deterministic online algorithms. We remark that α^* does not have a closed-form expression in general – we need to numerically solve the system of equations in Eq. (6) to obtain α^*. Given that the cost function f is arbitrary, we argue that this is not surprising.

(Example: Linear Cost) In some special cases the calculation of α^* via Eq. (6) can be significantly simplified. For example, when $f(x) = \sigma x$ with some constant $\sigma \in [0, v_{\min})$, we have $c_1 = c_2 = \cdots = c_k = \sigma$ and $f^*(v) = k(v - \sigma)$. Substituting f^* into Eq. (6) leads to

$$\frac{\alpha^*}{k} = \frac{\lambda^*_{\tau+1} - \sigma}{(v_{\min} - \sigma)(\tau + 1)} = \frac{\lambda^*_{\tau+2} - \lambda^*_{\tau+1}}{\lambda^*_{\tau+1} - \sigma} = \cdots = \frac{\lambda^*_k - \lambda^*_{k-1}}{\lambda^*_{k-1} - \sigma},$$

which solves to the following analytical solution:

$$\lambda^*_m = \alpha^* \Big(1 + \frac{\alpha^*}{k}\Big)^{m-\tau-1} \cdot \frac{\tau + 1}{k} \cdot (v_{\min} - \sigma) + \sigma, \qquad (7)$$

where m varies within $\{\tau + 1, \cdots, k\}$. By Eq. (5), we have

$$\tau = \lceil \frac{k}{\alpha^*} \rceil - 1. \qquad (8)$$

Substituting τ into Eq. (7), and further using the fact that $\lambda^*_k = v_{\max}$ (the first bullet in Theorem 1), we reach to the following equation of α^*:

$$\Big(1 + \frac{\alpha^*}{k}\Big)^{k - \lceil \frac{k}{\alpha^*} \rceil} \cdot \frac{\alpha^*}{k} \cdot \lceil \frac{k}{\alpha^*} \rceil = \frac{v_{\max} - \sigma}{v_{\min} - \sigma}. \qquad (9)$$

Based on Theorem 1, Eq. (9) has a unique root in variable $\alpha^* \in [1, +\infty)$, which can be easily computed via numerical methods such as bisection. Substituting α^* back to Eq. (7) will give us the optimal threshold $\boldsymbol{\lambda}^*$.

3. PERFORMANCE EVALUATION

(**Simulation Setup**) We consider a sequence of 500 items (i.e., $T = 500$), each with a value within the range of $[25, 200]$, namely, $v_{\min} = 25$ and $v_{\max} = 200$. For the cost function, we assume $f(x) = \frac{1}{5}x^2$. To simulate different arrival scenarios, we construct the following three types of arrival instances.

- Type 1: low2high. For arrival instances that are of type low2high, the values of the first half (i.e.., 250 items) are between $v_{\min}$ and $(v_{\min} + v_{\max})/2$, and those of the second half are between $(v_{\min} + v_{\max})/2$ and $v_{\max}$. We use low2high to simulate the scenario when being aggressive at the beginning (i.e., accepting too many items in earlier stages) may be penalized as it is likely to have sufficient number of high-value items later on.
- Type 2: random. In this type of arrival instances, the values of all the 500 items are randomly sampled from $[v_{\min}, v_{\max}]$. We use random to simulate the scenario when we have completely no knowledge of future arrivals.
- Type 3: high2low. This type is configured in contrast to low2high. We use high2low to simulate the scenario when being too reserved at the beginning (i.e., rejecting too many items in earlier stages) may be penalized.

(**Performance Metric**) To empirically evaluate the competitive ratio of TP_{λ^*}, we generate N samples of arrival instances $\mathcal{I}_n$ with $n \in [N]$, and calculate the empirical ratio of TP_{λ^*} by averaging over $N = 1000$ samples of arrival instances as follows:

$$\text{Empirical Ratio} \triangleq \frac{1}{N} \sum_{n=1}^{N} \frac{\mathsf{OPT}(\mathcal{I}_n)}{\mathsf{TP}_{\lambda^*}(\mathcal{I}_n)},$$

where $\mathsf{TP}_{\lambda^*}(\mathcal{I}_n)$ denotes the social surplus achieved by our threshold policy TP_{λ^*} in the online setting, and $\mathsf{OPT}(\mathcal{I}_n)$ is the optimal performance in hindsight (or in the offline setting when $\mathcal{I}_n$ is known a priori). For each given $\mathcal{I}_n$, we compute $\mathsf{OPT}(\mathcal{I}_n)$ by solving a mixed-integer optimization problem using Gurobi[3].

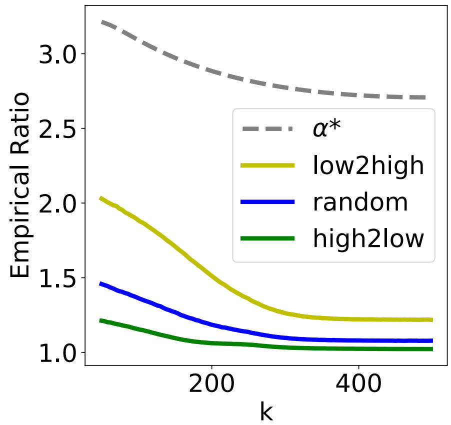

Figure 2: Performance of TP_{λ^*} under different types of arrival instances. Other than α^*, each curve shows the empirical ratio of TP_{λ^*} over 1000 arrival instances sampled by their corresponding types.

(**Numerical Results**) Fig. 2 shows that the optimal competitive ratio α^* is roughly within [2.5, 3.2] when k varies from 50 to 500. Recall that α^* denotes the worst-case scenario and is always sample independent. In comparison, the empirical ratio is highly dependent on samples. For example, the empirical ratios of TP_{λ^*} are always close to 1 when input sequences are of type high2low, but become considerably worse in face of low2high. The performance of TP_{λ^*} over random is between that of low2high and high2low, which follows our intuition. For all these three types, the empirical ratios are below α^* since by definition α^* captures the worst-possible performance. In the meanwhile, Fig. 2 shows that as k grows, all the empirical ratios decrease, so does α^*.

4. CONCLUSIONS AND FUTURE WORK

We proposed online selection with convex costs, and derived an optimal threshold policy which achieves the best-possible performance of all deterministic online algorithms.

Recall that k denotes the maximum number of items we can accept/select. Currently, it remains unclear if the optimal competitive ratio α^* asymptotically converges to some lower bound when k grows (as illustrated in Fig. 2). If α^* indeed converges, then it is interesting to study i) how to characterize and interpret this converged lower bound, and ii) whether this lower bound can be outperformed by randomized algorithms (e.g., randomized threshold policies).

5. REFERENCES

[1] R. El-Yaniv, A. Fiat, R. M. Karp, and G. Turpin, "Optimal search and one-way trading online algorithms," *Algorithmica*, vol. 30, no. 1, pp. 101–139, 2001.

[2] J. Lorenz, K. Panagiotou, and A. Steger, "Optimal algorithms for k-search with application in option pricing," *Algorithmica*, vol. 55, no. 2, pp. 311–328, 2009.

[3] X. Tan, B. Sun, A. Leon-Garcia, Y. Wu, and D. H. Tsang, "Mechanism design for online resource allocation: A unified approach," *Proc. ACM Meas. Anal. Comput. Syst. (SIGMETRICS '20)*, vol. 4, no. 2, Jun. 2020.

[3] https://www.gurobi.com/