Submodular Secretary Problem with Shortlists

Shipra Agrawal[1]
Columbia University, New York, NY, USA, 10027
sa3305@columbia.edu

Mohammad Shadravan
Columbia University, New York, NY, USA, 10027
ms4961@columbia.edu

Cliff Stein[2]
Columbia University, New York, NY, USA, 10027
cliff@ieor.columbia.edu

Abstract

In submodular k-secretary problem, the goal is to select k items in a randomly ordered input so as to maximize the expected value of a given monotone submodular function on the set of selected items. In this paper, we introduce a relaxation of this problem, which we refer to as submodular k-secretary problem with shortlists. In the proposed problem setting, the algorithm is allowed to choose more than k items as part of a shortlist. Then, after seeing the entire input, the algorithm can choose a subset of size k from the bigger set of items in the shortlist. We are interested in understanding to what extent this relaxation can improve the achievable competitive ratio for the submodular k-secretary problem. In particular, using an $O(k)$ sized shortlist, can an online algorithm achieve a competitive ratio close to the best achievable offline approximation factor for this problem? We answer this question affirmatively by giving a polynomial time algorithm that achieves a $1 - 1/e - \epsilon - O(k^{-1})$ competitive ratio for any constant $\epsilon > 0$, using a shortlist of size $\eta_\epsilon(k) = O(k)$. This is especially surprising considering that the best known competitive ratio (in polynomial time) for the submodular k-secretary problem is $(1/e - O(k^{-1/2}))(1 - 1/e)$ [20].

The proposed algorithm also has significant implications for another important problem of submodular function maximization under random order streaming model and k-cardinality constraint. We show that our algorithm can be implemented in the streaming setting using a memory buffer of size $\eta_\epsilon(k) = O(k)$ to achieve a $1 - 1/e - \epsilon - O(k^{-1})$ approximation. This result substantially improves upon [28], which achieved the previously best known approximation factor of $1/2 + 8 \times 10^{-14}$ using $O(k \log k)$ memory; and closely matches the known upper bound for this problem [24].

2012 ACM Subject Classification Mathematics of computing → Submodular optimization and polymatroids, Theory of computation → Online algorithms, Theory of computation → Streaming, sublinear and near linear time algorithms

Keywords and phrases Submodular Optimization, Secretary Problem, Streaming Algorithms

Digital Object Identifier 10.4230/LIPIcs.ITCS.2019.1

Related Version A full version of the paper is available as [1], `https://arxiv.org/abs/1809.05082`.

[1] Research supported in part by Google Faculty Research Awards 2017 and Amazon Research Awards 2017.
[2] Research supported in part by NSF grants CCF-1421161 and CCF-1714818.

10th Innovations in Theoretical Computer Science (ITCS 2019).
Editor: Avrim Blum; Article No. 1; pp. 1:1–1:19
Leibniz International Proceedings in Informatics
LIPICS Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

1 Introduction

In the classic *secretary problem*, n items appear in random order. We know n, but don't know the value of an item until it appears. Once an item arrives, we have to irrevocably and immediately decide whether or not to select it. Only one item is allowed to be selected, and the objective is to select the most valuable item, or perhaps to maximize the expected value of the selected item [11, 15, 23]. It is well known that the optimal policy is to observe the first n/e items without making any selection and then select the first item whose value is larger than the value of the best item in the first n/e items [11]. This algorithm, given by [11], is asymptotically optimal, and hires the best secretary with probability at least $1/e$. Hence it is also $1/e$-competitive for the expected value of the chosen item, and it can be shown that no algorithm can beat $1/e$-competitive ratio in expectation.

Many variants and generalizations of the secretary problem have been studied in the literature, see e.g., [3, 32, 30, 33, 21, 4]. [21, 4] introduced a multiple choice secretary problem, where the goal is to select k items in a randomly ordered input so as to maximize the *sum* of their values; and [21] gave an algorithm with an asymptotic competitive ratio of $1-O(1/\sqrt{k})$. Thus as $k \rightarrow \infty$, the competitive ratio approaches 1. Recent literature studied several generalizations of this setting to multidimensional knapsacks [26], and proposed algorithms for which the expected online solution approaches the best offline solution as the knapsack sizes become large (e.g., [13, 10, 2]).

In another variant of multiple-choice secretary problem, [6] and [16] introduce the *submodular k-secretary problem*. In this secretary problem, the algorithm again selects k items, but the value of the selected items is given by a monotone submodular function f. The algorithm has value oracle access to the function, i.e., for any given set T, an algorithm can query an oracle to find its value $f(T)$ [31]. The algorithm can select at most k items, $a_1 \cdots, a_k$, from a randomly ordered sequence of n items. The goal is to maximize $f(\{a_1, \cdots, a_k\})$. Currently, the best result for this setting is due to [20], who achieve a $1/e$-competitive ratio in exponential time, or $\frac{1}{e}(1-\frac{1}{e})$ in polynomial time. In this case, the offline problem is NP-hard and hard-to approximate beyond the factor of $1-1/e$ achieved by the greedy algorithm [27]. However, it is unclear if a competitive ratio of $1-1/e$ can be achieved by an online algorithm for the submodular k-secretary problem even when k is large.

Our model: secretary problem with shortlists

In this paper, we consider a relaxation of the secretary problem where the algorithm is allowed to select a *shortlist* of items that is larger than the number of items that ultimately need to be selected. That is, in a multiple-choice secretary problem with cardinality constraint k, the algorithm is allowed to choose more than k items as part of a shortlist. Then, after seeing the entire input, the algorithm can choose a subset of size k from the bigger set of items in the shortlist.

This new model is motivated by some practical applications of secretary problems, such as hiring (or assignment problems), where in some cases it may be possible to tentatively accept a larger number of candidates (or requests), while deferring the choice of the final k-selections to after all the candidates have been seen. Since there may be a penalty for declining candidates who were part of the shortlist, one would prefer that the shortlist is not much larger than k.

Another important motivation is theoretical: we wish to understand to what extent this relaxation of the secretary problem can improve the achievable competitive ratio. This question is in the spirit of several other methods of analysis that allow an online algorithm to have additional power, such as *resource augmentation* [18, 29].

The potential of this relaxation is illustrated by the basic secretary problem, where the aim is to select the item of maximum value among randomly ordered inputs. There, it is not difficult to show that if an algorithm picks every item that is better than the items seen so far, the true maximum will be found, while the expected number of items picked under randomly ordered inputs will be $O(\log n)$. Further, we show that this approach can be easily modified to get the maximum with $1-\epsilon$ probability while picking at most $O(\ln(1/\epsilon))$ items for any constant $\epsilon > 0$. Thus, with just a constant sized shortlist, we can break the $1/e$ barrier for the secretary problem and achieve a competitive ratio that is arbitrarily close to 1.

Motivated by this observation, we ask if a similar improvement can be achieved by relaxing the submodular k-secretary problem to allow a shortlist. That is, instead of choosing k items, the algorithm is allowed to chose $\eta(k)$ items as part of a shortlist, for some function η; and at the end of all inputs, the algorithm chooses k items from the $\eta(k)$ selected items. Then, what is the relationship between $\eta(\cdot)$ and the competitive ratio for this problem? Can we achieve a solution close to the best offline solution when $\eta(k)$ is not much bigger than k, for example when $\eta(k) = \theta(k)$?

In this paper, we answer this question affirmatively by giving a polynomial time algorithm that achieves $1 - 1/e - \epsilon - O(k^{-1})$ competitive ratio for the submodular k-secretary problem using a shortlist of size $\eta(k) = O(k)$. This is surprising since $1-1/e$ is the best achievable approximation (in polynomial time) for the offline problem. Further, for some special cases of submodular functions, we demonstrate that an $O(1)$ shortlist allows us to achieve a $1-\epsilon$ competitive ratio. These results demonstrate the power of (small) shortlists for closing the gap between online and offline (polynomial time) algorithms.

We also discuss connections of secretary problem with shortlists to the related streaming settings. While a streaming algorithm does not qualify as an online algorithm (even when a shortlist is allowed), we show that our algorithm can in fact be implemented in a streaming setting to use $\eta(k) = O(k)$ memory buffer; and our results significantly improve the available results for the submodular random order streaming problem.

1.1 Problem Definition

We now give a more formal definition. Items from a set $\mathcal{U} = \{a_1, a_2, \dots, a_n\}$ (pool of items) arrive in a uniformly random order over n sequential rounds. The set $\mathcal{U}$ is apriori fixed but unknown to the algorithm, and the total number of items n is known to the algorithm. In each round, the algorithm irrevocably decides whether to add the arriving item to a *shortlist* A or not. The algorithm's value at the end of n rounds is given by

$$\mathrm{ALG} = \mathbb{E}[\max_{S \subseteq A, |S| \leq k} f(S)]$$

where $f(\cdot)$ is a monotone submodular function. The algorithm has value oracle access to this function. The optimal offline utility is given by

$$\mathrm{OPT} := f(S^*), \text{ where } S^* = \arg\max_{S \subseteq [n], |S| \leq k} f(S).$$

We say that an algorithm for this problem achieves a competitive ratio c using shortlist of size $\eta(k)$, if at the end of n rounds, $|A| \leq \eta(k)$ and $\frac{\mathrm{ALG}}{\mathrm{OPT}} \geq c$.

Given the shortlist A, since the problem of computing the solution $\arg\max_{S \subseteq A, |S| \leq k} f(S)$ can itself be computationally intensive, our algorithm will also track and output a subset $A^* \subseteq A, |A^*| \leq k$. We will lower bound the competitive ratio by bounding $\frac{f(A^*)}{f(S^*)}$.

The above problem definition has connections to some existing problems studied in the literature. The well-studied online submodular k-secretary problem described earlier is obtained from the above definition by setting $\eta(k) = k$, i.e., it is same as the case when no extra items can be selected as part of a shortlist. Another related problem is *submodular random order streaming problem* studied in [28]. In that problem, items from a set $\mathcal{U}$ arrive online in random order and the algorithm aims to select a subset $S \subseteq \mathcal{U}, |S| \leq k$ in order to maximize $f(S)$. The streaming algorithm is allowed to maintain a *buffer* of size $\eta(k) \geq k$. However, the streaming problem is distinct from the submodular k-secretary problem with shortlists in several important ways. On one hand, since an item previously selected in the memory buffer can be discarded and replaced by a new items, a memory buffer of size $\eta(k)$ does not imply a shortlist of size at most $\eta(k)$. On the other hand, in the secretary setting, we are allowed to memorize/store more than $\eta(k)$ items without adding them to the shortlist. Thus an algorithm for submodular k-secretary problem with shortlist of size $\eta(k)$ may potentially use a buffer of size larger than $\eta(k)$. Our algorithms, as described in the paper, do use a large buffer. But we will show those algorithms can in fact be implemented to use only $\eta(k) = O(k)$ buffer, thus obtaining matching results for the streaming problem.

1.2 Our Results

Our main contributation is an online algorithm for the submodular k-secretary problem with shortlists that, for any constant $\epsilon > 0$, achieves a competitive ratio of $1 - \frac{1}{e} - \epsilon - O(\frac{1}{k})$ with $\eta(k) = O(k)$. Note that for submodular k-secretary problem there is an upper bound of $1 - 1/e$ on the achievable aproximation factor, even in the offline setting, and this upper bound applies to our problem for arbitrary size $\eta(\cdot)$ of shortlists. On the other hand for online monotone submodular k-secretary problem, i.e., when $\eta(k) = k$, the best competitive ratio achieved in the literature is $1/e - O(k^{-1/2})$ [20]. Remarkably, with only an $O(k)$ size shortlist, our online algorithm is able to achieve a competitive ratio that is arbitrarily close to the offline upper bound of $1 - 1/e$.

In the theorem statements below, big-Oh notation $O(\cdot)$ is used to represent asymptotic behavior with respect to k and n. We assume the standard value oracle model: the only access to the submodular function is through a black box returning $f(S)$ for a given set S, and each such query can be done in $O(1)$ time.

▶ **Theorem 1.** *For any constant $\epsilon > 0$, there exists an online algorithm (Algorithm 2) for the submodular k-secretary problem with shortlists that achieves a competitive ratio of $1 - \frac{1}{e} - \epsilon - O(\frac{1}{k})$, with shortlist of size $\eta_\epsilon(k) = O(k)$. Here, $\eta_\epsilon(k) = O(2^{poly(1/\epsilon)}k)$.*

Specifically, we have $\eta_\epsilon(k) = c^{\frac{\log(1/\epsilon)}{\epsilon^2}} \binom{\frac{1}{\epsilon^6}\log(1/\epsilon)}{\frac{1}{\epsilon^4}\log(1/\epsilon)}\ k$ for some constant c.

Further, we give an efficient implementation of Algorithm 2 that uses a memory buffer of size at most $\eta_\epsilon(k)$ to get the following result for the problem of *submodular random order streaming problem* described in the previous section.

▶ **Theorem 2.** *For any constant $\epsilon \in (0, 1)$, there exists an algorithm for the submodular random order streaming problem that achieves $1 - \frac{1}{e} - \epsilon - O(\frac{1}{k})$ approximation to OPT while using a memory buffer of size at most $\eta_\epsilon(k) = O(k)$. Also, the number of objective function evaluations for each item, amortized over n items, is $O(1 + \frac{k^2}{n})$.*

The above result significantly improves over the state-of-the-art results in random order streaming model [28], which are an approximation ratio of $\frac{1}{2} + 8 \times 10^{-14}$ using a memory of size $O(k \log k)$. In addition it closely matches the known upper bound for this problem [24].

In [24], the authors demonstrate the existence of a monotone sumbodular function f such that any constant-pass algorithm that finds a $(1+\epsilon)(1-1/k)^k$ approximation with probability at least 0.99 requires $\Omega(n/k^2)$ space in random order streaming model.

Also note from Theorem 2 that our algorithm can be implemented with running time linear in n, the size of the input ($O(n+k^2)$ time to be precise). This is significant as, until recently, it was not known if there exists a linear time algorithm achieving a $1-1/e-\epsilon$ approximation even for the offline monotone submodular maximization problem under cardinality constraint[25]. Another interesting aspect of our algorithm is that it is highly parallel. Even though the decision for each arriving item may take time that is exponential in $1/\epsilon$ (roughly $\eta_\epsilon(k)/k$), it can be readily parallelized among multiple (as many as $\eta_\epsilon(k)/k$) processors.

It is natural to ask whether these shortlists are, in fact, too powerful. Maybe they could actually allow us to always match the best offline algorithm. We give a negative result in this direction and show that even if we have unlimited computation power, for any function $\eta(k) = o(n)$, we can get no better than 7/8-competitive algorithm using a shortlist of size $\eta(k)$. Note that with unlimited computational power, the offline problem can be solved exactly. This result demonstrates that having a shortlist does not make the online problem too easy - even with a shortlist (of size $o(n)$) there is an information theoretic gap between the online and offline problem.

▶ **Theorem 3.** *No online algorithm (even with unlimited computational power) can achieve a competitive ratio better than $7/8+o(1)$ for the submodular k-secretary problem with shortlists, while using a shortlist of size $\eta(k) = o(n)$.*

Finally, for some special cases of monotone submodular functions, we can asymptotically approach the optimal solution. The first one is the family of functions we call m-submdular. A function f is m-submodular if it is submodular and there exists a submodular function F such that for all S:

$$f(S) = \max_{T\subseteq S, |T|\leq m} F(T) \ .$$

▶ **Theorem 4.** *If f is an m-submodular function, there exists an online algorithm for the submodular k-secretary problem with shortlists that achieves a competitive ratio of $1-\epsilon$ with shortlist of size $\eta_{\epsilon,m}(k) = O(1)$. Here, $\eta_{\epsilon,m}(k) = (2m+3)\ln(2/\epsilon)$.*

A proof of Theorem 4 along with the relevant algorithm appear in the full version [1].

Another special case is monotone submodular functions f satisfying the following property: $f(\{a_1, \cdots, a_i + \alpha, \cdots, a_k\}) \geq f(\{a_1, \cdots, a_i, \cdots, a_k\})$, for any $\alpha > 0$ and $1 \leq i \leq k$. We can show that the algorithm by [21] asymptotically approaches optimal solution for such functions, but we omit the details.

1.3 Comparison to related work

We compare our results (Theorem 1 and Theorem 2) to the best known results for *submodular k-secretary problem* and *submodular random order streaming problem*, respectively.

The best known algorithm so far for submodular k-secretary problem is by [20], with asymptotic competitive ratio of $1/e - O(k^{-1/2})$. In their algorithm, after observing each element, they use an oracle to compute optimal offline solution on the elements seen so far. Therefore it requires exponential time in n. The best competitive ratio that they can get in polynomial time is $\frac{1}{e}(1-\frac{1}{e}) - O(k^{-1/2})$. In comparison, by using a shortlist of size $O(k)$ our

Table 1 submodular k-secretary problem settings.

	#selections	Comp ratio	Running time	Comp ratio in poly(n)
[20]	k	$1/e - O(k^{-1/2})$	$exp(n)$	$\frac{1}{e}(1-1/e)$
this	$O_\epsilon(k)$	$1-1/e-\epsilon-O(1/k)$	$O_\epsilon(n)$	$1-1/e-\epsilon-O(1/k)$

Table 2 submodular random order streaming problem.

	Memory size	Approximation ratio	Running time	update time
[17]	$O(k)$	0.19	$O(n)$	O(1)
[28]	$O(k \log k)$	$1/2 + 8 \times 10^{-14}$	$O(n \log k)$	$O(\log k)$
[5]	$O(\frac{1}{\epsilon} k \log k)$	$1/2 - \epsilon$	$poly(n, k, 1/\epsilon)$	$O(\frac{1}{\epsilon} \log k)$
this	$O_\epsilon(k)$	$1-1/e-\epsilon-O(1/k)$	$O_\epsilon(n+k^2)$	amortized $O_\epsilon(1+\frac{k^2}{n})$

(polynomial time) algorithm achieves a competitive ratio of $1-\frac{1}{e}-\epsilon-O(k^{-1})$. In addition to substantially improving the above-mentioned result for submodular k-secretary problem, this closely matches the best possible offline approximation ratio of $1-1/e$ in polynomial time. Further, our algorithm is linear time. Table 1 summarizes this comparison. Here, $O_\epsilon(\cdot)$ hides the dependence on the constant ϵ. The hidden constant in $O_\epsilon(.)$ is $c^{\frac{\log(1/\epsilon)}{\epsilon^2}\left(\frac{\frac{1}{\epsilon^6}\log(1/\epsilon)}{\frac{1}{\epsilon^4}\log(1/\epsilon)}\right)}$ for some absolute constant c.

In the streaming setting, [9] provided a single pass streaming algorithm for monotone submodular function maximization under k-cardinality constraint, that achieves a 0.25 approximation under adversarial ordering of input. Their algorithm requires $O(1)$ function evaluations per arriving item and $O(k)$ memory. The currently best known approximation under adversarial order streaming model is by [5], who achieve a $1/2-\epsilon$ approximation with a memory of size $O(\frac{1}{\epsilon}k\log k)$. There is an $1/2+o(1)$ upper bound on the competitive ratio achievable by any streaming algorithm for submodular maximization that only queries the value of the submodular function on feasible sets (i.e., sets of cardinality at most k) while using $o(n)$ memory [28].

[17] initiated the study of submodular random order streaming problem. Their algorithm uses $O(k)$ memory and a total of n function evaluations to achieve 0.19 approximation. The state of the art result in the random order input model is due to [28] who achieve a $1/2+8\times 10^{-14}$ approximation, while using a memory buffer of size $O(k\log k)$.

Table 2 provides a detailed comparison of our result in Theorem 2 to the above-mentioned results for submodular random order streaming problem, showing that our algorithm substantially improves the existing results for most aspects of the problem.

There is also a line of work studying the online variant of the submodular welfare maximization problem (e.g., [22, 7, 19]). In this problem, the items arrive online, and each arriving item should be allocated to one of m agents with a submodular valuation functions $w_i(S_i)$ where S_i is the subset of items allocated to i-th agent). The goal is to partition the arriving items into m sets to be allocated to m agents, so that the sum of valuations over all agents is maximized. This setting is incomparable with the submodular k-secretary problem setting considered here.

1.4 Organization

The rest of the paper is organized as follows. Section 2 describes our main algorithm (Algorithm 2) for the submodular k-secretary problem with shortlists, and demonstrates that its shortlist size is bounded by $\eta_\epsilon(k)=O(k)$. In Section 3, we analyze the competitive ratio

Algorithm 1 Algorithm for secretary with shortlist. (finding max online)

Inputs: number of items N, items in $I = \{a_1, \ldots, a_N\}$ arriving sequentially, $\delta \in (0, 1]$.
Initialize: $A \leftarrow \emptyset$, $u = n\delta/2$, $M = -\infty$
$L \leftarrow 4\ln(2/\delta)$
for $i = 1$ to N **do**
  if $a_i > M$ **then**
    $M \leftarrow a_i$
    if $i \geq u$ and $|A| < L$ **then**
      $A \leftarrow A \cup \{a_i\}$
    end if
  end if
end for
return A, and $A^* := \max_{i \in A} a_i$

of this algorithm to prove Theorem 1. In Section 4, we provide an alternate implementation of Algorithm 2 that uses a memory buffer of size at most $\eta_\epsilon(k)$, in order to prove Theorem 2. Finally, in Section 5, we provide a proof of our impossibility result stated in Theorem 3. The proof of Theorem 4 along with the relevant algorithm can be found in the full version [1].

2 Algorithm description

Before giving our algorithm for submodular k-secretary problem with shortlists, we describe a simple technique for (classic) secretary problem with shortlists that achieves a $1 - \delta$ competitive ratio using shortlists of size logarithmic in $1/\delta$. Recall that in the secretary problem, the aim is to select an item with expected value close to the maximum among a pool of items $I = (a_1, \ldots, a_N)$ arriving sequentially in a uniformly random order. We will consider the variant with shortlists, where we now want to pick a shortlist which contains an item with expected value close to the maximum. We propose the following simple algorithm. For the first $n\delta/2$ rounds, don't add any items to the shortlist, but just keep track of the maximum value seen so far. For all subsequent rounds, for any arriving item i that has a value a_i greater than or equal to the maximum value seen so far, add it to the shortlist if number of items added so far is less than or equal to $L = 4\ln(2/\delta)$. This algorithm is summarized as Algorithm 1. Clearly, for constant δ, this algorithm uses a shortlist of size $L = O(1)$. Further, under a uniform random ordering of input, we can show that the maximum value item will be part of the shortlist with probability $1 - \delta$. (See Proposition 25 in Section 3.)

There are two main difficulties in extending this idea to the submodular k-secretary problem with shortlists. First, instead of one item, here we aim to select a set S of k items using an $O(k)$ length shortlist. Second, the contribution of each new item i to the objective value, as given by the submodular function f, depends on the set of items selected so far.

The first main concept we introduce to handle these difficulties is that of dividing the input into sequential blocks that we refer to as (α, β) windows. Below is the precise construction of (α, β) windows, for any postivie integers α and β, such that k/α is an integer.

We use a set of random variables $X_1, \ldots, X_m$ defined in the following way. Throw n balls into m bins uniformly at random. Then set X_j to be the number of balls in the jth bin. We call the resulting X_j's a (n, m)-*ball-bin random set*.

Algorithm 2 Algorithm for **submodular** k-secretary with shortlist.

Inputs: set $\bar{I} = \{\bar{a}_1, \ldots, \bar{a}_n\}$ of n items arriving sequentially, submodular function f, parameter $\epsilon \in (0, 1]$.
Initialize: $S_0 \leftarrow \emptyset, R_0 \leftarrow \emptyset, A \leftarrow \emptyset, A^* \leftarrow \emptyset$, constants $\alpha \geq 1, \beta \geq 1$ which depend on the constant ϵ.
Divide indices $\{1, \ldots, n\}$ into (α, β) windows as prescribed by Definition 5.
for window $w = 1, \ldots, k/\alpha$ **do**
  for every slot s_j in window w, $j = 1, \ldots, \alpha\beta$ **do**
    Concurrently for all subsequences of previous slots $\tau \subseteq \{s_1, \ldots, s_{j-1}\}$ of length $|\tau| < \alpha$ in window w, call the online algorithm in Algorithm 1 with the following inputs:
    - number of items $N = |s_j| + 1$, $\delta = \frac{\epsilon}{2}$, and
    - item values $I = (a_0, a_1, \ldots, a_{N-1})$, with

$$a_0 := \max_{x \in R_{1,\ldots,w-1}} \Delta(x | S_{1,\ldots,w-1} \cup \gamma(\tau))$$

$$a_\ell := \Delta(s_j(\ell) | S_{1,\ldots,w-1} \cup \gamma(\tau)), \forall \ell = 1, \ldots, N-1$$

    where $s_j(\ell)$ denotes the ℓ^{th} item in the slot s_j.
    Let $A_j(\tau)$ be the shortlist returned by Algorithm 1 for slot j and subsequence τ.
    Add all items except the dummy item 0 to the shortlist A. Let's $A(j) = \bigcup_\tau A_j(\tau)$. That is,

$$A \leftarrow A \cup (A(j) \cap s_j)$$

  end for
  After seeing all items in window w, compute R_w, S_w as defined in (3) and (4) respectively.
  $A^* \leftarrow A^* \cup (S_w \cap A)$
end for
return A, A^*.

▶ **Definition 5** ((α, β) windows)**.** Let $X_1, \ldots, X_{k\beta}$ be a $(n, k\beta)$-ball-bin random set. Divide the indices $\{1, \ldots, n\}$ into $k\beta$ slots, where the j-th slot, s_j, consists of X_j consecutive indices in the natural way, that is, slot 1 contains the first X_1 indices, slot 2 contains the next X_2 indices, etc. Next, we define k/α windows, where window w consists of $\alpha\beta$ consecutive slots, in the same manner as we assigned slots.

Thus, the q^{th} slot is composed of indices $\{\ell, \ldots, r\}$, where $\ell = X_1 + ... + X_{q-1} + 1$ and $r = X_1 + ... + X_q$. Further, if the ordered the input is $\bar{a}_1, \ldots, \bar{a}_n$, then we say that the items inside the slot s_q are $\bar{a}_\ell, \bar{a}_{\ell+1}, \ldots, \bar{a}_r$. To reduce notation, when clear from context, we will use s_q and w to also indicate the *set of items* in the slot s_q and window w respectively.

When α and β are large enough constants, some useful properties can be obtained from the construction of these windows and slots. First, roughly α items from the optimal set S^* are likely to lie in each of these windows; and further, it is unlikely that two items from S^* will appear in the same slot. (These statements will be made more precise in the analysis where precise setting of α, β in terms of ϵ will be provided.) Consequently, our algorithm can focus on identifying a constant number (roughly α) of optimal items from each of these windows, with at most one item coming from each of the $\alpha\beta$ slots in a window. The core of our algorithm is a subroutine that accomplishes this task in an online manner using a shortlist of constant size in each window.

To implement this task, we use a greedy selection method that considers all possible α sized subsequences of the $\alpha\beta$ slots in a window, and aims to identify the subsequence that maximizes the increment over the 'best' items identified so far. More precisely, for any subsequence $\tau = (s_1, \ldots, s_\ell)$ of the $\alpha\beta$ slots in window w, we define a 'greedy' subsequence $\gamma(\tau)$ of items as:

$$\gamma(\tau) := \{i_1, \ldots, i_\ell\} \tag{1}$$

where

$$i_j := \arg\max_{i \in s_j \cup R_{1,\ldots,w-1}} f(S_{1,\ldots,w-1} \cup \{i_1, \ldots, i_{j-1}\} \cup \{i\}) - f(S_{1,\ldots,w-1} \cup \{i_1 \ldots, i_{j-1}\}). \tag{2}$$

In (2) and in the rest of the paper, we use shorthand $S_{1,\ldots,w}$ to denote $S_1 \cup \cdots \cup S_w$, and $R_{1,\ldots,w}$ to denote $R_1 \cup \cdots \cup R_w$, etc. We also will take unions of subsequences, which we interpret as the union of the elements in the subsequences. Here R_w is defined to be the union of all greedy subsequences of length α, and S_w to be the best subsequence among those. That is,

$$R_w = \cup_{\tau:|\tau|=\alpha} \gamma(\tau) \tag{3}$$

and

$$S_w = \gamma(\tau^*), \tag{4}$$

where

$$\tau^* := \arg\max_{\tau:|\tau|=\alpha} f(S_{1,\ldots,w-1} \cup \gamma(\tau)) - f(S_{1,\ldots,w-1}). \tag{5}$$

Note that i_j (refer to (2)) can be set as either an item in slot s_j or an item *from a previous greedy subsequence* in $R_1 \cup \cdots \cup R_{w-1}$. The significance of the latter relaxation will become clear in the analysis.

As such, identifying the sets R_w and S_w involves looking forward in a slot s_j to find the best item (according to the given criterion in (2)) among all the items in the slot. To obtain an online implementation of this procedure, we use an online subroutine that employs the algorithm (Algorithm 1) for the basic secretary problem with shortlists described earlier. This online procedure will result in selection of a set H_w potentially larger than R_w, while ensuring that each element from R_w is part of H_w with a high probability $1 - \delta$ at the cost of adding extra $\log(1/\delta)$ items to the shortlist. Note that R_w and S_w can be computed *exactly* at the end of window w.

Algorithm 2 summarizes the overall structure of our algorithm. In the algorithm, for any item i and set V, we define $\Delta_f(i|V) := f(V \cup \{i\}) - f(V)$.

The algorithm returns both the shortlist A which we show to be of size $O(k)$ in the following proposition, as well as a set $A^* = \cup_w (S_w \cap A)$ of size at most k to compete with S^*. In the next section, we will show that $\mathbb{E}[f(A^*)] \geq (1 - \frac{1}{e} - \epsilon - O(\frac{1}{k}))f(S^*)$ to provide a bound on the competitive ratio of this algorithm.

▶ **Proposition 6.** *Given k, n, and any constant α, β and ϵ, the size of shortlist A selected by Algorithm 2 is of size at most $4k\beta\binom{\alpha\beta}{\alpha}\log(2/\epsilon) = O(k)$.*

Proof. For each window $w = 1, \ldots, k/\alpha$, and for each of the $\alpha\beta$ slots in this window, lines 6 through 7 in Algorithm 2 runs Algorithm 1 for $\binom{\alpha\beta}{\alpha}$ times (for all α length subsequences). By construction of Algorithm 1, for each run it will add at most $L \leq 4\log(2/\epsilon)$ items to the shortlist. Therefore, over all windows, Algorithm 2 adds at most $\frac{k}{\alpha} \times \alpha\beta\binom{\alpha\beta}{\alpha}L = O(k)$ items to the shortlist. ◀

3 Bounding the competitive ratio (Proof of Theorem 1)

In this section we show that for any $\epsilon \in (0,1)$, Algorithm 2 with an appropriate choice of constants α, β, achieves the competitive ratio claimed in Theorem 1 for the submodular k-secretary problem with shortlists.

Recall the following notation defined in the previous section. For any collection of sets $V_1, \ldots, V_\ell$, we use $V_{1,\ldots,\ell}$ to denote $V_1 \cup \cdots \cup V_\ell$. Also, recall that for any item i and set V, we denote $\Delta_f(i|V) := f(V \cup \{i\}) - f(V)$.

Proof overview

The proof is divided into two parts. We first show a lower bound on the ratio $\mathbb{E}[f(\cup_w S_w)]/\mathrm{OPT}$ in Proposition 24, where S_w is the subset of items as defined in (4) for every window w. Later in Proposition 27, we use the said bound to derive a lower bound on the ratio $\mathbb{E}[f(A^*)]/\mathrm{OPT}$, where $A^* = A \cap (\cup_w S_w)$ is the subset of the shortlist returned by Algorithm 2.

Specifically, in Proposition 24, we provide settings of parameters α, β such that $\mathbb{E}[f(\cup_w S_w)] \geq \left(1 - \frac{1}{e} - \frac{\epsilon}{2} - O(\frac{1}{k})\right) \mathrm{OPT}$. A central idea in the proof of this result is to show that for every window w, given $R_{1,\ldots,w-1}$, the items tracked from the previous windows, any of the k items from the optimal set S^* has at least $\frac{\alpha}{k}$ probability to appear either in window w, or among the tracked items $R_{1,\ldots,w-1}$. Further, the items from S^* that appear in window w, appear independently, and in a uniformly random slot in this window. (See Lemma 15.) These observations allow us to show that, in each window w, there exists a subsequence $\tilde{\tau}_w$ of close to α slots, such that the greedy sequence of items $\gamma(\tilde{\tau}_w)$ will be almost "as good as" a randomly chosen sequence of α items from S^*. More precisely, denoting $\gamma(\tilde{\tau}_w) = (i_1, \ldots, i_t)$, in Lemma 19, for all $j = 1, \ldots, t$, we lower bound the increment in function value $f(\cdots)$ on adding i_j over the items in $S_{1,\ldots,w-1} \cup i_{1,\ldots,j-1}$ as:

$$\begin{aligned}
&\mathbb{E}[\Delta_f(i_j|S_{1,\ldots,w-1} \cup \{i_1, \ldots, i_{j-1}\})|T_{1,\ldots,w-1}, i_1, \ldots, i_{j-1}] \\
&\geq \frac{1}{k}\left((1 - \frac{\alpha}{k})f(S^*) - f(S_{1,\ldots,w-1} \cup \{i_1, \ldots, i_{j-1}\})\right) .
\end{aligned}$$

We then deduce (using standard techniques for the analysis of greedy algorithm for submodular functions) that

$$\begin{aligned}
&\mathbb{E}[\left(1 - \frac{\alpha}{k}\right) f(S^*) - f(S_{1,\ldots,w-1} \cup \gamma(\tilde{\tau}_w))|S_{1,\ldots,w-1}] \\
&\leq e^{-t/k}\left(\left(1 - \frac{\alpha}{k}\right) f(S^*) - f(S_{1,\ldots,w-1})\right) .
\end{aligned}$$

Now, since the length t of $\tilde{\tau}_w$ is close to α (as we show in Lemma 21) and since $S_w = \gamma(\tau^*)$ with τ^* defined as the "best" subsequence of length α (refer to definition of τ^* in (5)), we can show that a similar inequality holds for $S_w = \gamma(\tau^*)$, i.e.,

$$\begin{aligned}
&\left(1 - \frac{\alpha}{k}\right) f(S^*) - \mathbb{E}[f(S_{1,\ldots,w-1} \cup S_w)|S_{1,\ldots,w-1}] \\
&\leq e^{-\alpha/k}\left(1 - \delta'\right)\left(\left(1 - \frac{\alpha}{k}\right) f(S^*) - f(S_{1,\ldots,w-1})\right) ,
\end{aligned}$$

where $\delta' \in (0,1)$ depends on the setting of α, β. (See Lemma 23.) Then repeatedly applying this inequality for $w = 1, \ldots, k/\alpha$, and setting δ, α, β appropriately in terms of ϵ, we can obtain $\mathbb{E}[f(S_{1,\ldots,W})] \geq \left(1 - \frac{1}{e} - \frac{\epsilon}{2} - \frac{1}{k}\right) f(S^*)$, completing the proof of Proposition 24.

However, a remaining difficulty is that while the algorithm keeps a track of the set S_w for every window w, it may not have been able to add all the items in S_w to the shortlist A during the online processing of the inputs in that window. In the proof of Proposition 27, we show that in fact the algorithm will add most of the items in $\cup_w S_w$ to the shortlist. More precisely, we show that given that an item i is in S_w, it will be in shortlist A with probability $1-\delta$, where δ is the parameter used while calling Algorithm 1 in Algorithm 2. Therefore, using properties of submodular functions it follows that with $\delta = \epsilon/2$, $\mathbb{E}[f(A^*)] = \mathbb{E}[f(\cup_w S_w \cap A)] \geq (1-\frac{\epsilon}{2})\mathbb{E}[f(\cup_w S_w)]$ (see Proposition 27). Combining this with the lower bound $\frac{\mathbb{E}[f(\cup_w S_w)]}{\text{OPT}} \geq (1-\frac{1}{e}-\frac{\epsilon}{2}-O(\frac{1}{k}))$ proven in Proposition 24, we complete the proof of competitive ratio bound stated in Theorem 1.

3.1 Preliminaries

The following properties of submodular functions are well known (e.g., see [8, 12, 14]).

▶ **Lemma 7.** *Given a monotone submodular function f, and subsets A, B in the domain of f, we use $\Delta_f(A|B)$ to denote $f(A \cup B) - f(B)$. For any set A and B, $\Delta_f(A|B) \leq \sum_{a\in A\setminus B} \Delta_f(a|B)$.*

▶ **Lemma 8.** *Denote by $A(p)$ a random subset of A where each element has a probability at least p to appear in A (not necessarily independently). Then $E[f(A(p))] \geq (1-p)f(\emptyset) + (p)f(A)$.*

We will use the following well known deviation inequality for martingales (or supermartingales/submartingales).

▶ **Lemma 9** (Azuma-Hoeffding inequality). *Suppose $\{X_k : k = 0, 1, 2, 3, ...\}$ is a martingale (or super-martingale) and $|X_k - X_{k-1}| < c_k$, almost surely. Then for all positive integers N and all positive reals r,*

$$P(X_N - X_0 \geq r) \leq \exp\left(\frac{-r^2}{2\sum_{k=1}^{N} c_k^2}\right).$$

And symmetrically (when X_k is a sub-martingale):

$$P(X_N - X_0 \leq -r) \leq \exp\left(\frac{-r^2}{2\sum_{k=1}^{N} c_k^2}\right).$$

▶ **Lemma 10** (Chernoff bound for Bernoulli r.v.). *Let $X = \sum_{i=1}^{N} X_i$, where $X_i = 1$ with probability p_i and $X_i = 0$ with probability $1-p_i$, and all X_i are independent. Let $\mu = \mathbb{E}(X) = \sum_{i=1}^{N} p_i$. Then,*

$$P(X \geq (1+\delta)\mu) \leq e^{-\delta^2\mu/(2+\delta)}$$

for all $\delta > 0$, and

$$P(X \leq (1-\delta)\mu) \leq e^{-\delta^2\mu/2}$$

for all $\delta \in (0, 1)$.

3.2 Some useful properties of (α, β) windows

All the proofs in this section are omitted and are provided in the full version [1].

We first prove some useful properties of (α, β) windows defined in Definition 5 and used in Algorithm 2. The first observation is that every item will appear uniformly at random in one of the $k\beta$ slots in (α, β) windows.

▶ **Definition 11.** For each item $e \in \bar{I}$, define $Y_e \in [k\beta]$ as the random variable indicating the slot in which e appears. We call vector $Y \in [k\beta]^n$ a *configuration*.

▶ **Lemma 12.** *Random variables $\{Y_e\}_{e\in I}$ are i.i.d. with uniform distribution on all $k\beta$ slots.*

This follows from the uniform random order of arrivals, and the use of the balls in bins process to determine the number of items in a slot during the construction of (α, β) windows.

Next, we make some observations about the probability of assignment of items in S^* to the slots in a window w, given the sets $R_{1,...,w-1}, S_{1,...,w-1}$ (refer to (3), (4) for definition of these sets). To aid analysis, we define the following new random variable T_w that will track all the useful information from a window w.

▶ **Definition 13.** Define $T_w := \{(\tau, \gamma(\tau))\}_\tau$, for all α-length subsequences $\tau = (s_1, \ldots, s_\alpha)$ of the $\alpha\beta$ slots in window w. Here, $\gamma(\tau)$ is a sequence of items as defined in (1). Also define $Supp(T_{1,\cdots,w}) := \{e | e \in \gamma(\tau)$ for some $(\tau, \gamma(\tau)) \in T_{1,\cdots,w}\}$ (Note that $Supp(T_{1,\cdots,w}) = R_{1,\ldots,w}$).

▶ **Lemma 14.** *For any window $w \in [W]$, $T_{1,\ldots,w}$ and $S_{1,\ldots,w}$ are independent of the ordering of elements within any slot, and are determined by the configuration Y.*

Following the above lemma, given a configuration Y, we will some times use the notation $T_{1,\ldots,w}(Y)$ and $S_{1,\ldots,w}(Y)$ to make this mapping explicit.

▶ **Lemma 15.** *For any item $i \in S^*$, window $w \in \{1, \ldots, W\}$, and slot s in window w, define*

$$p_{is} := Pr(i \in s \cup Supp(T) | T_{1,\ldots,w-1} = T). \tag{6}$$

Then, for any pair of slots s', s'' in windows $w, w+1, \ldots, W$,

$$p_{is'} = p_{is''} \geq \frac{1}{k\beta} \ . \tag{7}$$

▶ **Lemma 16.** *For any window w, $i, j \in S^*, i \neq j$ and $s, s' \in w$, the random variables $\mathbf{1}(Y_i = s | T_{1,\cdots,w-1} = T)$ and $\mathbf{1}(Y_j = s' | T_{1,\cdots,w-1} = T)$ are independent. That is, given $T_{1,\cdots,w-1} = T$, items $i, j \in S^*, i \neq j$ appear in any slot s in w independently.*

3.3 Bounding $\mathbb{E}[f(\cup_w S_w)]/\mathrm{OPT}$

In this section, we use the observations from the previous sections to show the existence of a random subsequence of slots $\tilde{\tau}_w$ of window w such that we can lower bound $f(S_{1,\ldots,w-1} \cup \gamma(\tilde{\tau}_w)) - f(S_{1,\ldots,w-1})$ in terms of $\mathrm{OPT} - f(S_{1,\ldots,w-1})$. This will be used to lower bound increment $\Delta_f(S_w | S_{1,\ldots,w-1}) = f(S_{1,\ldots,w-1} \cup \gamma(\tau^*)) - f(S_{1,\ldots,w-1})$ in every window.

▶ **Definition 17** (Z_s **and** $\tilde{\gamma}_w$)**.** Create sets of items $Z_s, \forall s \in w$ as follows: for every slot s, add every item from $i \in S^* \cap s$ independently with probability $\frac{1}{k\beta p_{is}}$ to Z_s. Then, for every item $i \in S^* \cap Supp(T)$, with probability α/k, add i to Z_s for a randomly chosen slot s in w. Define subsequence $\tilde{\tau}_w$ as the sequence of slots with $Z_s \neq \emptyset$.

▶ **Lemma 18.** *Given any $T_{1,\ldots,w-1} = T$, for any slot s in window w, all $i, i' \in S^*, i \neq i'$ will appear in Z_s independently with probability $\frac{1}{k\beta}$. Also, given T, for every $i \in S^*$, the probability to appear in Z_s is equal for all slots s in window w. Further, each $i \in S^*$ occurs in Z_s for at most one slot s.*

Proof. First consider $i \in S^* \cap Supp(T)$. Then, $\Pr(i \in Z_s | T) = \frac{\alpha}{k} \times \frac{1}{\alpha\beta} = \frac{1}{k\beta}$ by construction. Also, the event $i \in Z_s | T$ is independent from $i' \in Z_s | T$ for any $i' \in S^*$ as i and i' are independently assigned to a Z_s in construction. Further, items in $S^* \cap Supp(T)$ are assigned with equal probability to slots in window w.

Now, consider $i \in S^*, i \notin Supp(T)$. Then, for all slots s in window w,

$$\Pr(i \in Z_s | T) = \Pr(Y_i = s | T) \frac{1}{p_{is} k\beta} = p_{is} \times \frac{1}{p_{is} k\beta} = \frac{1}{k\beta},$$

where p_{is} is defined in (6). We used that $p_{is} = \Pr(Y_i = s | T)$ for $i \notin Supp(T)$. Independence of events $i \in Z_s | T$ for items in $S^* \backslash Supp(T)$ follows from Lemma 16, which ensures $Y_i = s | T$ and $Y_j = s | T$ are independent for $i \neq j$; and from independent selection among items with $Y_i = s$ into Z_s.

The fact that every $i \in S^*$ occurs in at most one Z_s follows from construction: i is assigned to Z_s of only one slot if $i \in Supp(T)$; and for $i \notin Supp(T)$, it can only appear in Z_s if i appears in slot s. ◀

▶ **Lemma 19.** *Given the sequence $\tilde{\tau}_w = (s_1, \ldots, s_t)$ defined in Definition 17, let $\gamma(\tilde{\tau}_s) = (i_1, \ldots, i_t)$, with $\gamma(\cdot)$ as defined in* (1)*. Then, for all $j = 1, \ldots, t$,*

$$\begin{aligned}&\mathbb{E}[\Delta_f(i_j | S_{1,\ldots,w-1} \cup \{i_1, \ldots, i_{j-1}\}) | T_{1,\ldots,w-1}, i_1, \ldots, i_{j-1}] \\ &\geq \frac{1}{k}\left((1 - \frac{\alpha}{k}) f(S^*) - f(S_{1,\ldots,w-1} \cup \{i_1, \ldots, i_{j-1}\})\right) \ .\end{aligned}$$

Proof. For any slot s' in window w, let $\{s : s \succ_w s'\}$ denote all the slots that appear after s' in the sequence of slots in window w.

Now, using Lemma 18, for any slot s such that $s \succ_w s_{j-1}$, we have that the random variables $\mathbf{1}(i \in Z_s | Z_{s_1} \cup \ldots \cup Z_{s_{j-1}})$ are i.i.d. for all $i \in S^* \backslash \{Z_{s_1} \cup \ldots \cup Z_{s_{j-1}}\}$. Next, we show that the probabilities $\Pr(i \in Z_{s_j} | Z_{s_1} \cup \ldots \cup Z_{s_{j-1}})$ are identical for all $i \in S^* \backslash \{Z_{s_1} \cup \ldots \cup Z_{s_{j-1}}\}$:

$$\begin{aligned}
&\Pr(i \in Z_{s_j} | Z_{s_1} \cup \ldots \cup Z_{s_{j-1}}) \\
&= \sum_{s : s \succ_w s_{j-1}} \Pr(i \in Z_s, s = s_j | Z_{s_1} \cup \ldots \cup Z_{s_{j-1}}) \\
&= \sum_{s : s \succ_w s_{j-1}} \Pr(i \in Z_s | s = s_j, Z_{s_1} \cup \ldots \cup Z_{s_{j-1}}) \Pr(s = s_j | Z_{s_1} \cup \ldots \cup Z_{s_{j-1}}) \ .
\end{aligned}$$

Now, from Lemma 18, the probability $\Pr(i \in Z_s | s = s_j, Z_{s_1} \cup \ldots \cup Z_{s_{j-1}})$ must be identical for all $i \notin Z_{s_1} \cup \ldots \cup Z_{s_{j-1}}$. Therefore, from above we have that for all $i, i' \in S^* \backslash \{Z_{s_1} \cup \ldots \cup Z_{s_{j-1}}\}$,

$$\Pr(i \in Z_{s_j} | Z_{s_1} \cup \ldots \cup Z_{s_{j-1}}) = \Pr(i' \in Z_{s_j} | Z_{s_1} \cup \ldots \cup Z_{s_{j-1}}) \geq \frac{1}{k} \ . \tag{8}$$

The lower bound of $1/k$ followed from the fact that at least one of the items from $S^* \backslash \{Z_{s_1} \cup \ldots \cup Z_{s_{j-1}}\}$ must appear in Z_{s_j} for s_j to be included in $\tilde{\tau}_w$. Thus, each of these probabilities is at least $1/k$. In other words, if an item is randomly picked from Z_{s_j}, it will be i with probability at least $1/k$, for all $i \in S^* \backslash \{Z_{s_1} \cup \ldots \cup Z_{s_{j-1}}\}$.

Now, by definition of $\gamma(\cdot)$ (refer to (1)), i_j is chosen greedily to maximize the increment $\Delta_f(i | S_{1,\ldots,w-1} \cup i_{1,\ldots,s-1})$ over all $i \in s_j \cup \mathit{Supp}(T_{1,\ldots,w-1}) \supseteq Z_{s_j}$. Therefore, we can lower bound the increment provided by i_j by that provided by a randomly picked item from Z_{s_j}. By using monotonicity of f,

$$\begin{aligned}
& && \mathbb{E}[\Delta_f(i_j | S_{1,\ldots,w-1} \cup \{i_1, \ldots, i_{j-1}\} | T_{1,\ldots,w-1} = T, i_1, \ldots, i_{j-1}] \\
\text{(by (8))} &\geq && \frac{1}{k}\mathbb{E}[\sum_{i \in S^* \backslash \{Z_1, \ldots Z_{s_{j-1}}\}} \mathbb{E}[\Delta_f(i | S_{1,\ldots,w-1} \cup \{i_1, \ldots, i_{j-1}\} | T, i_1, \ldots, i_{j-1}]] \\
\text{(by Lemma 7)} &\geq && \frac{1}{k}\mathbb{E}[\left(f(S^* \backslash \{Z_1, \ldots Z_{s_{j-1}}\}) - f(S_{1,\ldots,w-1} \cup \{i_1, \ldots, i_{j-1}\})\right) | T] \\
&\geq && \frac{1}{k}\mathbb{E}[(f(S^* \backslash \cup_{s' \in w} Z_{s'}) - f(S_{1,\ldots,w-1} \cup \{i_1, \ldots, i_{j-1}\}) | T] \\
\text{(by Lemma 18 and 8)} &\geq && \frac{1}{k}\left(\left(1 - \frac{\alpha}{k}\right) f(S^*) - f(S_{1,\ldots,w-1} \cup \{i_1, \ldots, i_{j-1}\})\right)
\end{aligned}$$

The last inequality uses the observation from Lemma 18 that given T, every $i \in S^*$ appears in $\cup_{s' \in w} Z_{s'}$ independently with probability α/k, so that every $i \in S^*$ appears in $S^* \backslash \cup_{s' \in w} Z_{s'}$ independently with probability $1 - \frac{\alpha}{k}$; along with Lemma 8 for submodular function f. ◀

Using standard techniques for the analysis of greedy algorithm, the following corollary of the previous lemma can be derived: given any $T_{1,\ldots,w-1} = T$:

▶ **Lemma 20.**

$$\mathbb{E}\left[\left(1 - \frac{\alpha}{k}\right) f(S^*) - f(S_{1,\ldots,w-1} \cup \gamma(\tilde{\tau}_w)) | T\right] \leq \mathbb{E}\left[e^{-\frac{|\tilde{\tau}_w|}{k}} \mid T\right] \left(\left(1 - \frac{\alpha}{k}\right) f(S^*) - f(S_{1,\ldots,w-1})\right).$$

Proof. Let $\pi_0 = (1 - \frac{\alpha}{k}) f(S^*) - \mathbb{E}[f(S_{1,\ldots,w-1}) | T_{1,\ldots,w-1} = T]$, and for $j \geq 1$,

$$\pi_j := (1 - \frac{\alpha}{k}) f(S^*) - \mathbb{E}[f(S_{1,\ldots,w-1} \cup \{i_1, \ldots, i_j\}) | T_{1,\ldots,w-1} = T, i_1, \ldots, i_{j-1}],$$

Then, subtracting and adding $(1 - \frac{\alpha}{k}) f(S^*)$ from the left hand side of the previous lemma, and taking expectation conditional on $T_{1,\ldots,w-1} = T, i_1, \ldots, i_{j-2}$, we get

$$-\mathbb{E}[\pi_j | T, i_1, \ldots, i_{j-2}] + \pi_{j-1} \geq \frac{1}{k} \pi_{j-1}$$

which implies

$$\mathbb{E}[\pi_j|T, i_1, \ldots, i_{j-2}] \le \left(1 - \frac{1}{k}\right)\pi_{j-1} \le \left(1 - \frac{1}{k}\right)^j \pi_0 \ .$$

By the martingale stopping theorem, this implies:

$$\mathbb{E}[\pi_t|T] \le \mathbb{E}\left[\left(1 - \frac{1}{k}\right)^t |T\right]\pi_0 \le \mathbb{E}\left[e^{-t/k}|T\right]\pi_0$$

where stopping time $t = |\tilde{\tau}_w|$. ($t = |\tilde{\tau}_w| \le \alpha\beta$ is bounded, therefore, the martingale stopping theorem can be applied). ◀

Next, we compare $\gamma(\tilde{\tau}_w)$ to $S_w = \gamma(\tau^*)$. Here, τ^* was defined has the 'best' greedy subsequence of length α (refer to (4) and (5)). To compare it with $\tilde{\tau}_w$, we need a bound on size of $\tilde{\tau}_w$.

▶ **Lemma 21.** *For any real $\delta \in (0, 1)$, and if $k \ge \alpha\beta$, $\alpha \ge 8\log(\beta)$ and $\beta \ge 8$, then given any $T_{1,\ldots,w-1} = T$,*

$$(1 - \delta)\left(1 - \frac{4}{\beta}\right)\alpha \le |\tilde{\tau}_w| \le (1 + \delta)\alpha,$$

with probability at least $1 - \exp(-\frac{\delta^2\alpha}{8\beta})$.

Proof. Appears in the full version. ◀

▶ **Lemma 22** (Corollary of Lemma 21)**.** *For any real $\delta' \in (0, 1)$, if parameters k, α, β satisfy $k \ge \alpha\beta$, $\beta \ge \frac{8}{(\delta')^2}$, $\alpha \ge 8\beta^2 \log(1/\delta')$, then given any $T_{1,\ldots,w-1} = T$, with probability at least $1 - \delta' e^{-\alpha/k}$,*

$$|\tilde{\tau}_w| \ge (1 - \delta')\alpha \ .$$

▶ **Lemma 23.** *For any real $\delta' \in (0, 1)$, if parameters k, α, β satisfy $k \ge \alpha\beta$, $\beta \ge \frac{8}{(\delta')^2}$, $\alpha \ge 8\beta^2 \log(1/\delta')$, then*

$$\mathbb{E}\left[\frac{k-\alpha}{k} OPT - f(S_{1,\ldots,w})|T_{1,\ldots,w-1}\right] \le (1 - \delta')e^{-\alpha/k}\left(\frac{k-\alpha}{k} OPT - f(S_{1,\ldots,w-1})\right) \ .$$

Proof. The lemma follows from substituting Lemma 22 in Lemma 20. ◀

Now, we can deduce the following proposition.

▶ **Proposition 24.** *For any real $\delta' \in (0, 1)$, if parameters k, α, β satisfy $k \ge \alpha\beta$, $\beta \ge \frac{8}{(\delta')^2}$, $\alpha \ge 8\beta^2 \log(1/\delta')$, then the set $S_{1,\ldots,W}$ tracked by Algorithm 2 satisfies*

$$\mathbb{E}[f(S_{1,\ldots,W})] \ge (1 - \delta')^2(1 - 1/e)OPT.$$

Proof. By multiplying the inequality Lemma 23 from $w = 1, \ldots, W$, where $W = k/\alpha$, we get

$$\mathbb{E}[f(S_{1,\ldots,W})] \ge (1 - \delta')(1 - 1/e)(1 - \frac{\alpha}{k})\text{OPT}.$$

Then, using $1 - \frac{\alpha}{k} \ge 1 - \delta'$ because $k \ge \alpha\beta \ge \frac{\alpha}{\delta'}$, we obtain the desired statement. ◀

3.4 Bounding $\mathbb{E}[f(A^*)]/\text{OPT}$

Here, we compare $f(S_{1\ldots,W})$ to $f(A^*)$, where $A^* = S_{1\ldots,W} \cap A$, with A being the shortlist returned by Algorithm 2. The main difference between the two sets is that in construction of shortlist A, Algorithm 1 is being used to compute the argmax in the definition of $\gamma(\tau)$, in an online manner. This argmax may not be computed exactly, so that some items from $S_{1\ldots,W}$ may not be part of the shortlist A. We use the following guarantee for Algorithm 1 to bound the probability of this event.

▶ **Proposition 25.** *For any $\delta \in (0, 1)$, and input $I = (a_1, \ldots, a_N)$, Algorithm 1 returns $A^* = \max(a_1, \ldots, a_N)$ with probability $(1 - \delta)$.*

The proof of the above proposition appears in the full version. Intuitively, it follows from the observation that if we select every item that improves the maximum of items seen so far, we would have selected $\log(N)$ items in expectation. The exact proof involves showing that on waiting $n\delta/2$ steps and then selecting maximum of every item that improves the maximum of items seen so far, we miss the maximum item with at most δ probability, and select at most $O(\log(1/\delta))$ items with probability $1-\delta$.

▶ **Lemma 26.** *Let A be the shortlist returned by Algorithm 2, and δ is the parameter used to call Algorithm 1 in Algorithm 2. Then, for given configuration Y, for any item a and window w, we have*

$$Pr(a \in A|Y, a \in S_{1,\cdots,w}) \geq 1-\delta\ .$$

Proof. From Lemma 14 by conditioning on Y, the set $S_{1,\cdots,W}$ is determined. Now if $a \in S_{1,\ldots,w}$, then for some slot s_j in an α length subsequence τ of some window w, we must have

$$a = \arg \max_{i \in s_j \cup R_{1,\ldots,w-1}} f(S_{1,\ldots,w-1} \cup \gamma(\tau) \cup \{i\}) - f(S_{1,\ldots,w-1} \cup \gamma(\tau)).$$

Let w' be the first such window, $\tau', s_{j'}$ be the corresponding subsequence and slot. Then, it must be true that

$$a = \arg \max_{i \in s_{j'}} f(S_{1,\ldots,w'-1} \cup \gamma(\tau') \cup \{i\}) - f(S_{1,\ldots,w'-1} \cup \gamma(\tau')).$$

(Note that the argmax in above is not defined on $R_{1,\cdots,w'-1}$). The configuration Y only determines the set of items in the items in slot $s_{j'}$, the items in $s_{j'}$ are still randomly ordered (refer to Lemma 14). Therefore, from Proposition 25, with probability $1-\delta$, a will be added to the shortlist $A_{j'}(\tau')$ by Algorithm 1. Thus $a \in A \supseteq A_{j'}(\tau')$ with probability at least $1-\delta$. ◀

▶ **Proposition 27.**

$$\mathbb{E}[f(A^*)] := \mathbb{E}[f(S_{1,\cdots,W} \cap A)] \geq (1-\frac{\epsilon}{2})\mathbb{E}[f(S_{1,\cdots,W})]$$

where $A^* := S_{1,\cdots,W} \cap A$ is the size k subset of shortlist A returned by Algorithm 2.

Proof. From the previous lemma, given any configuration Y, we have that each item of $S_{1,\cdots,W}$ is in A with probability at least $1-\delta$, where $\delta = \epsilon/2$ in Algorithm 2. Therefore using Lemma 8, the expected value of $f(S_{1,\cdots,W} \cap A)$ is at least $(1-\delta)\mathbb{E}[F(S_{1,\cdots,W})]$. ◀

Proof of Theorem 1

Now, we can show that Algorithm 2 provides the results claimed in Theorem 1 for appropriate settings of α, β in terms of ϵ. Specifically for $\delta' = \epsilon/4$, set α, β as smallest integers satisfying $\beta \geq \frac{8}{(\delta')^2}$, $\alpha \geq 8\beta^2 \log(1/\delta')$. Then, using Proposition 24 and Proposition 27, for $k \geq \alpha\beta$ we obtain:

$$\mathbb{E}[f(A^*)] \geq (1-\frac{\epsilon}{2})(1-\delta')^2(1-1/e)\mathrm{OPT} \geq (1-\epsilon)(1-1/e)\mathrm{OPT}.$$

This implies a lower bound of $1-\epsilon-1/e-\alpha\beta/k = 1-\epsilon-1/e-O(1/k)$ on the competitive ratio.

The $O(k)$ bound on the size of the shortlist was demonstrated in Proposition 6.

4 Streaming (Proof of Theorem 2)

In this section, we show that Algorithm 2 can be implemented in a way that it uses a memory buffer of size at most $\eta(k) = O(k)$; and the number of objective function evaluations for each arriving item is $O(1+\frac{k^2}{n})$. This will allow us to obtain Theorem 2 as a corollary of Theorem 1.

In the current description of Algorithm 2, there are several steps in which the algorithm potentially needs to store $O(n)$ previously seen items in order to compute the relevant quantities. First, in Step 6, in order to be able to compute $\gamma(\tau)$ for all less than α length subsequences τ of slots $s_1, \ldots, s_{j-1}$, the algorithm should have stored all the items that arrived in the slots $s_1, \ldots, s_{j-1}$. However, this memory requirement can be reduced by a small modification of the algorithm, so that at the end of iteration $j-1$, the algorithm has already computed $\gamma(\tau)$ for all such τ, and stored them to be used in iteration j. In fact, this can be implemented in a memory efficient manner, in the following way. For every subsequence τ of slots $s_1,\ldots,s_{j-1}$ of length $<\alpha$, consider prefix $\tau' = \tau \backslash s_{j-1}$. Assume $\gamma(\tau')$ is available from iteration $j-2$. If $\tau' = \tau$, then $\gamma(\tau) = \gamma(\tau')$. Otherwise, in Step 6 of iteration $j-1$, the algorithm must have considered the subsequence τ' while going through all subsequences of length less than α of slots $s_1,\ldots,s_{j-2}$. Now, modify the implementation of Step 6 so that the algorithm also tracks the (true) maximum $M_{j-1}(\tau')$ of $a_0, a_1, \ldots, a_N$ for each τ'. Then, $\gamma(\tau)$ can be obtained by extending $\gamma(\tau')$ by $M_{j-1}(\tau')$, i.e., $\gamma(\tau) = \{\gamma(\tau'), M_{j-1}(\tau')\}$. Thus, at the end of iteration $j-1$, $\gamma(\tau)$ would have been computed for all subsequences τ relevant for iteration j, and so on. In order to store these $\gamma(\tau)$ for every subsequence τ (of at most α slots from $\alpha\beta$ slots), we require a memory buffer of size at most $\alpha^2\binom{\alpha\beta}{\alpha} = O(1)$.

Secondly, across windows and slots, the algorithm keeps track of $R_w, S_w, w = 1,\ldots,k/\alpha$ where $W = k/\alpha$. In the current description of Algorithm 2, these sets are computed after seeing all the items in window w in Step 9. Thus, all the items arriving in that window would be needed to be stored in order to compute them, requiring $O(n)$ memory buffer. However, the alternate implementation discussed in the previous paragraph reduces this memory requirement to $O(k)$ as well. Using the above implementation, at the end of iteration $\alpha\beta$ for the last slot $s_{\alpha\beta}$ in window w, we would have computed and stored $\gamma(\tau)$ for all the subsequences τ of length α of slots $s_1,\ldots,s_{\alpha\beta}$. R_w is simply defined as union of all items in $\gamma(\tau)$ over all such τ (refer to (3)). And, $S_w = \gamma(\tau^*)$ for the best subsequence τ^* among these subsequences (refer to (4)). Thus, computing R_w and S_w does not require any additional memory buffer. Storing R_w and S_w for all windows requires a buffer of size at most $\sum_w |R_w| + |S_w| = \frac{k}{\alpha} \times \alpha\binom{\alpha\beta}{\alpha} + k = O(k)$. Therefore, the total buffer required to implement Algorithm 2 is of size $O(k)$.

Finally, let's bound the number of objective function evaluations for each arriving item. Each arriving item is processed in Step 6, where objective function is evaluated twice for each subsequence to compute the corresponding a_i. Since there are atmost $\binom{\alpha\beta}{\alpha}$ subsequences τ for which this quantity is computed, the total number of times this computation is performed is bounded by $2\binom{\alpha\beta}{\alpha} = O(1)$. For each τ, we also compute a_0 in the beginning of the slot. Computing a_0 for each τ involves taking max over all items in $R_{1,\ldots,w-1}$, and requires $2|R_{1,\ldots,w-1}| \le 2k\binom{\alpha\beta}{\alpha}$ evaluations of the objective function. Due to this computation, in the worst-case, the update time for an item can be $2k\binom{\alpha\beta}{\alpha}^2 + 2\binom{\alpha\beta}{\alpha} = O(k)$. However, since a_0 is computed *once* in the beginning of the slot for each τ, the total update time over all items is bounded by $2k\binom{\alpha\beta}{\alpha}^2 \times k\beta + \binom{\alpha\beta}{\alpha} \times n = O(k^2+n)$. Therefore the amortized update time for each item is $O(1+\frac{k^2}{n})$. This concludes the proof of Theorem 2.

5 Impossibility Result (Proof of Theorem 3)

In this section we provide an upper bound showing the following:

▶ **Theorem 3.** *No online algorithm (even with unlimited computational power) can achieve a competitive ratio better than $7/8 + o(1)$ for the submodular k-secretary problem with shortlists, while using a shortlist of size $\eta(k) = o(n)$.*

In the following proof, for simplicity of notation, we prove the desired bound for submodular $(k+1)$-secretary problem. For any given n, k, we construct a set of instances of the submodular $(k+1)$-secretary problem with shortlists such that any online algorithm that uses a shortlist of size $\eta(k+1)$ will have competitive ratio of at most $\frac{7}{8} + \frac{\eta(k+1)}{2n}$ on a randomly selected instance from this set.

First, we define a monotone submodular function f as follows. The ground set consists of $\frac{n}{2k} + n - 1$ items. There are two types of items, C and D, with $L := n/2k$ items of type C and $n-1$ items of type D. We define $f(\phi) := 0$, $f(\{c\}) := k$ for $c \in C$, and $f(\{d\}) := 1$ for all $d \in D$. Also there is a collection of L *disjoint* sets $T_\ell = \{c^\ell, d_1^\ell, \cdots, d_k^\ell\}$, $\ell = 1, 2, \ldots L$, such that $c^\ell \in C$ and $d_j^\ell \in D$. We define $f(T_\ell) := 2k$ for all $\ell = 1, \ldots, L$. Now, let

$$g(t) := k + \frac{k}{2} + \cdots + \frac{k}{2^{i-1}} + \frac{(t-ik)}{2^i},$$

where $i = \lfloor t/k \rfloor$. It is easy to see that g is a monotone submodular function.

Now, define f on the remaining subsets of the ground set as follows. For all S with $|S| \geq 1$,

- $|S \cap C| \geq 2 \implies f(S) := 2k + 1$
- $|S \cap C| = 0 \implies f(S) := 1 + g(|S| - 1)$
- $|S \cap C| = 1 \implies S \cap C = \{c^\ell\}$ for some $\ell \in [L] \implies$

$$f(S) := \min\{2k+1, k + \frac{1}{2}g(|S|-1) + \frac{k'}{2^{i+1}}\},$$

where $k' = |S \cap \{d_1^\ell, \cdots, d_k^\ell\}|$, $i = \lfloor (|S|-1)/k \rfloor$.

Observe that since $g(k) = k$, for any such subset S of size at most $k+1$, we have $f(S) \leq k + \frac{k}{2} + \frac{k}{2} = 2k$.

▶ **Lemma 28.** *f is a monotone submodular function.*

Now, denote $D^\ell := T^\ell \cap D = \{d_1^\ell, \cdots, d_k^\ell\}$ for $\ell = 1, 2, \ldots, L$. Also, let $D' = D \setminus (\bigcup_{\ell=1}^{L} D^\ell)$. Now define L input instances $\{I_\ell\}_{\ell=1,\ldots,L}$, each of size n, as follows. For any arbitrary subset $\tilde{D} \subseteq D'$ of size $n - Lk - 1$, define $I_\ell = \bigcup_{i=1,\ldots,L} D^i \cup \tilde{D} \cup \{c^\ell\}$, for $\ell = 1, \ldots, L$. Thus, for instance I_ℓ, the the optimal $k+1$ subset is T^ℓ with value $f(T^\ell) = 2k$.

Now consider any algorithm for the submodular secretary problem with shortlists and cardinality constraint $k+1$. We denote by Alg the set of $\eta(k+1)$ items selected by the algorithm as part of the shortlist. Let $\bar{I}$ denote an instance chosen uniformly at random from $I_\ell, \ell = 1, \ldots, L$. Let π denote a random ordering of n items in $\bar{I}$. We denote by random variable $(\bar{I}, \pi)$ the randomly ordered input instance to the algorithm. Also we denote by $\bar{T}, \bar{D}$ and $\bar{c}$, the corresponding T^ℓ, D^ℓ and c^ℓ.

Now we claim

▶ **Lemma 29.** $\mathbb{E}_{(\bar{I},\pi)}[|Alg \cap \bar{D}|] \leq k/2 + \eta(k+1)/L$.

Proof. Appears in the full version [1]. ◀

Now on input $\bar{I}$, if the algorithm doesn't select $\bar{c}$ as part of shortlist Alg, then by definition of f for sets that do not contain any item of type C, we have

$$f(A^*) := \max_{S \subseteq Alg : |S| \leq k+1} f(S) \leq 1 + g(k) < k + 2.$$

Otherwise, if algorithm selects $\bar{c}$, then by definition of f,

$$f(A^*) := \max_{S \subseteq Alg : |S| \leq k+1} f(S) \leq \max_{S \subseteq Alg \setminus (\bar{D} \cup \{\bar{c}\}) : |S| \leq k - |Alg \cap \bar{D}|} f(S \cup \bar{D} \cup \{\bar{c}\}) = k + \frac{k}{2} + \frac{1}{2}|Alg \cap \bar{D}|,$$

and therefore by lemma 29

$$\mathbb{E}[f(A^*)] \leq k + \frac{k}{2} + \frac{k}{4} + \frac{\eta(k+1)}{2L} = \frac{7k}{4} + \frac{k\eta(k+1)}{n} .$$

Since the optimal is equal to $\mathbb{E}[f(\bar{T})] = 2k$, the competitive ratio is upper bounded by

$$\frac{7}{8} + \frac{\eta(k+1)}{2n}.$$

This proves a competitive ratio upper bound of $\frac{7}{8} + o(1)$ when $\eta(k+1) = o(n)$, to complete the proof of Theorem 3.

References

1 Shipra Agrawal, Mohammad Shadravan, and Cliff Stein. Submodular Secretary Problem with Shortlists, 2018. arXiv:1809.05082.

2 Shipra Agrawal, Zizhuo Wang, and Yinyu Ye. A Dynamic Near-Optimal Algorithm for Online Linear Programming. *Operations Research*, 62(4):876–890, 2014.

3 Miklos Ajtai, Nimrod Megiddo, and Orli Waarts. Improved Algorithms and Analysis for Secretary Problems and Generalizations. *SIAM J. Discret. Math.*, 14(1):1–27, January 2001.

4 Moshe Babaioff, Nicole Immorlica, David Kempe, and Robert Kleinberg. Online Auctions and Generalized Secretary Problems. *SIGecom Exch.*, 7(2):7:1–7:11, June 2008.

5 Ashwinkumar Badanidiyuru, Baharan Mirzasoleiman, Amin Karbasi, and Andreas Krause. Streaming Submodular Maximization: Massive Data Summarization on the Fly. In *Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '14, pages 671–680, New York, NY, USA, 2014. ACM.

6 Mohammadhossein Bateni, Mohammadtaghi Hajiaghayi, and Morteza Zadimoghaddam. Submodular Secretary Problem and Extensions. *ACM Trans. Algorithms*, 9(4):32:1–32:23, October 2013.

7 Niv Buchbinder, Moran Feldman, and Mohit Garg. Online Submodular Maximization: Beating 1/2 Made Simple. *arXiv preprint*, 2018. arXiv:1807.05529.

8 Niv Buchbinder, Moran Feldman, Joseph (Seffi) Naor, and Roy Schwartz. Submodular Maximization with Cardinality Constraints. In *Proceedings of the Twenty-fifth Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA '14, pages 1433–1452. Society for Industrial and Applied Mathematics, 2014.

9 Amit Chakrabarti and Sagar Kale. Submodular maximization meets streaming: matchings, matroids, and more. *Mathematical Programming*, 154(1):225–247, December 2015.

10 Nikhil Devanur and Thomas Hayes. The Adwords Problem: Online Keyword Matching with Budgeted Bidders under Random Permutations. In *ACM EC*, 2009.

11 E. B. Dynkin. The optimum choice of the instant for stopping a Markov process. *Soviet Math. Dokl*, 4, 1963.

12 Uriel Feige, Vahab S. Mirrokni, and Jan Vondrák. Maximizing Non-monotone Submodular Functions. *SIAM J. Comput.*, 40(4):1133–1153, July 2011.

13 J. Feldman, M. Henzinger, N. Korula, V. Mirrokni, and C. Stein. Online stochastic packing applied to display ad allocation. *Algorithms–ESA 2010*, pages 182–194, 2010.

14 Moran Feldman and Rico Zenklusen. The Submodular Secretary Problem Goes Linear. In *Proceedings of the 2015 IEEE 56th Annual Symposium on Foundations of Computer Science (FOCS)*, FOCS '15, pages 486–505. IEEE Computer Society, 2015.

15 Thomas S Ferguson et al. Who solved the secretary problem? *Statistical science*, 4(3):282–289, 1989.

16 Anupam Gupta, Aaron Roth, Grant Schoenebeck, and Kunal Talwar. Constrained Non-monotone Submodular Maximization: Offline and Secretary Algorithms. In *Proceedings of the 6th International Conference on Internet and Network Economics*, WINE'10, pages 246–257. Springer-Verlag, 2010.

17 Tom Hess and Sivan Sabato. The submodular secretary problem under a cardinality constraint and with limited resources. *CoRR*, abs/1702.03989, 2017. arXiv:1702.03989.

18 Bala Kalyanasundaram and Kirk Pruhs. Speed is as powerful as clairvoyance. *J. ACM*, 47(4):617–643, 2000.

19 Michael Kapralov, Ian Post, and Jan Vondrák. Online Submodular Welfare Maximization: Greedy is Optimal. In *Proceedings of the Twenty-fourth Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA '13, pages 1216–1225. Society for Industrial and Applied Mathematics, 2013.

20 Thomas Kesselheim and Andreas Tönnis. Submodular Secretary Problems: Cardinality, Matching, and Linear Constraints. In *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques (APPROX/RANDOM 2017)*, Leibniz International Proceedings in Informatics (LIPIcs), pages 16:1–16:22, 2017.

21 Robert Kleinberg. A Multiple-choice Secretary Algorithm with Applications to Online Auctions. In *Proceedings of the Sixteenth Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA '05, pages 630–631, Philadelphia, PA, USA, 2005. Society for Industrial and Applied Mathematics.

22 Nitish Korula, Vahab Mirrokni, and Morteza Zadimoghaddam. Online Submodular Welfare Maximization: Greedy Beats 1/2 in Random Order. In *Proceedings of the Forty-seventh Annual ACM Symposium on Theory of Computing*, STOC '15, pages 889–898. ACM, 2015.

23 D. V. Lindley. Dynamic Programming and Decision Theory. *Journal of the Royal Statistical Society. Series C (Applied Statistics)*, 10(1):39–51, 1961.

24 Andrew McGregor and Hoa T Vu. Better Streaming Algorithms for the Maximum Coverage Problem. In *20th International Conference on Database Theory*, 2017.

25 Baharan Mirzasoleiman, Ashwinkumar Badanidiyuru, Amin Karbasi, Jan Vondrák, and Andreas Krause. Lazier Than Lazy Greedy. In *Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence*, AAAI'15, pages 1812–1818. AAAI Press, 2015.

26 Martin Moser, Dusan P Jokanovic, and Norio Shiratori. An algorithm for the multidimensional multiple-choice knapsack problem. *IEICE transactions on fundamentals of electronics, communications and computer sciences*, 80(3):582–589, 1997.

27 George L Nemhauser and Laurence A Wolsey. Best algorithms for approximating the maximum of a submodular set function. *Mathematics of operations research*, 3(3):177–188, 1978.

28 Ashkan Norouzi-Fard, Jakub Tarnawski, Slobodan Mitrovic, Amir Zandieh, Aidasadat Mousavifar, and Ola Svensson. Beyond 1/2-Approximation for Submodular Maximization on Massive Data Streams. In *Proceedings of the 35th International Conference on Machine Learning*, volume 80, pages 3829–3838. PMLR, 10–15 jul 2018.

29 Cynthia A. Phillips, Cliff Stein, Eric Torng, and Joel Wein. Optimal time-critical scheduling via resource augmentation. *Algorithmica*, 32:163–200, 2001.

30 Robert J Vanderbei. The optimal choice of a subset of a population. *Mathematics of Operations Research*, 5(4):481–486, 1980.

31 Jan Vondrak. Optimal Approximation for the Submodular Welfare Problem in the Value Oracle Model. In *Proceedings of the Fortieth Annual ACM Symposium on Theory of Computing*, STOC '08, pages 67–74. ACM, 2008.

32 John G. Wilson. Optimal choice and assignment of the best m of n randomly arriving items. *Stochastic Processes and their Applications*, 39(2):325–343, 1991.

33 John G Wilson. Optimal choice and assignment of the best m of n randomly arriving items. *Stochastic processes and their applications*, 39(2):325–343, 1991.