Submodular Secretary Problem with Shortlists under General Constraints

Mohammad Shadravan

Submitted in partial fulfillment of the
requirements for the degree of
Doctor of Philosophy
in the Graduate School of Arts and Sciences

COLUMBIA UNIVERSITY

2020

Abstract

Submodular Secretary Problem with Shorlitsts under General Constraints

Mohammad Shadravan

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 $O(k)$ shortlist, can an online algorithm achieve a competitive ratio close to the best achievable online 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 $\eta_\epsilon(k) = O(k)$. Also, for the special case of m-submodular functions, we demonstrate an algorithm that achieves a $1 - \epsilon$ competitive ratio for any constant $\epsilon > 0$, using an $O(1)$ shortlist. Finally, 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 for submodular function maximization in the random order streaming model. This substantially improves upon the previously best known approximation factor of $1/2 + 8 \times 10^{-14}$ [Norouzi-Fard et al. 2018] that

used a memory buffer of size $O(k \log k)$.

We further generalize our results to the case of matroid constraints. We design an algorithm that achieves a $1/2(1 - 1/e^2 - \epsilon - O(1/k))$ competitive ratio for any constant $\epsilon > 0$, using a shortlist of size O(k). This is especially surprising considering that the best known competitive ratio for the matroid secretary problem is $O(\log \log k)$. An important application of our algorithm is for the random order streaming of submodular functions. We show that our algorithm can be implemented in the streaming setting using $O(k)$ memory. It achieves a $1/2(1 - 1/e^2 - \epsilon - O(1/k))$ approximation. The previously best known approximation ratio for streaming submodular maximization under matroid constraint is 0.25 (adversarial order) due to [Feldman et al.], [Chekuri et al.], and [Chakrabarti et al.]. Moreover, we generalize our results to the case of p-matchoid constraints and give a $\frac{1}{p+1}(1 - 1/e^{p+1} - \epsilon - O(1/k))$ approximation using $O(k)$ memory, which asymptotically approaches the best known offline guarantee $\frac{1}{p+1}$ [Nemhauser et al.]. Finally we empirically evaluate our results on real world data sets such as YouTube video and Twitter stream.

Table of Contents

List of Tables

List of Figures

Acknowledgements

First and foremost, I would like to thank my advisors, Clifford Stein and Shipra Agrawal. I learned a great deal from their insightful comments on my research and their great advice on algorithmic thinking and problem solving. They thought me how to conduct research and how to write an academic paper. I am grateful to them for helping me to clarify my ideas and explanation. They also gave me the freedom in pursuing different research areas. Moreover, I would like to thank them for carefully reading this thesis.

I would like to thank the other members of my thesis committee, Jay Sethuraman, Tim Roughgarden, and Amin Karbasi, for taking the time and effort to read my thesis. I would also thank Jay for always being available to give helpful advice. Before coming to Columbia, I was fortunate to go to university of Waterloo and Sharif university of technology, where I was lucky to work with many excellent advisors and students in particular Jochen and Zac. Finally, I would like to thank my parents and my sister, for their support throughout my life.

Dedication

This thesis is dedicated to my parents.

Preface

In chapter 1, we introduce different notions defined in this thesis such as submodularity, secretary problems, matroid constraints and our model *shortlist model*. We also briefly take a look at relevant problems. In the second chapter we study some simpler instances of sumobular functions under the secretary setting and design near optimal algorithm for these cases. One special case is the m-submodular functions that will be the building block of our main algorithm in Chapter 4. We also extend the algorithm for multiple choice secretary problem [35] to a subclass of submodular functions with near optimal guarantee. In chapter 3, we provide our first attempt for submodular k-secretary problem with shortlists. We achieve an algorithm with the same guarantee as it was known previously in the streaming setting, i.e., $1/2 - \epsilon$. But we improve the required memory buffer, in the sense that we remove the dependency on ϵ. In chapter 4, we present our main result which is a near optimal algorithm for submodular k-secretary problem with shortlists, both in terms of competitive ratio and size of shortlist (memory buffer). In the final chapter, we further generalize our result to the case of matroid constraints and p-mathcoid constraints.

Chapter 1: Introduction and Background

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 thesis, 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 $O(k)$ 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 $(1/e - O(k^{-1/2}))(1 - 1/e)$ [34]. Further, for the special case of m-submodular functions, we demonstrate an algorithm that achieves $1-\epsilon$ competitive ratio for any constant $\epsilon > 0$, using an $O(1)$ shortlist.

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 substantially improves upon [44], which achieved the previously best known approximation factor of $1/2 + 8 \times 10^{-14}$ using $O(k \log k)$ memory. Furthermore in the random order streaming setting our algorithm is asymptotically tight due to $1 - 1/e$ upper bound in [39].

Next, we consider more general constraints, namely, matroid constraints. In the *matroid secretary problem*, which is a generalization of the classic secretary problem, the elements of a matroid $\mathcal{M}$ arrive in random order. Once we observe an item we need to irrevocably decide whether or not to accept it. The set of selected elements should form an independent set of the matroid. The goal is to maximize the total sum of the values assigned to these elements. The existence of a constant competitive algorithm is a long-standing open problem.

In this thesis, we introduce a version of this problem, which we refer to as submodular matroid secretary problem with shortlists. In this setting, the algorithm is allowed to choose a subset of items as part of a shortlist, possibly more than $k = rk(\mathcal{M})$ items. Then, after seeing the entire input, the algorithm can choose an independent subset from the shortlist. Furthermore we generalize the objective function to any monotone submodular function. The main question is can an online algorithm achieve a constant competitive ratio using a shortlist of size $O(k)$?

We design an algorithm that achieves a $\frac{1}{2}(1 - 1/e^2 - \epsilon - O(1/k))$ competitive ratio for any constant $\epsilon > 0$, using a shortlist of size $O(k)$. This is especially surprising considering that the best known competitive ratio for the matroid secretary problem is $O(\log \log k)$. We are also able to get a constant competitive algorithm using shortlist of size at most k and also a constant competitive algorithm in the preemption model.

An important application of our algorithm is for the random order streaming of submodular functions. We show that our algorithm can be implemented in the streaming setting using $O(k)$ memory. It achieves a $\frac{1}{2}(1-1/e^2-\epsilon-O(1/k))$ approximation. The previously best known approximation ratio for streaming submodular maximization under matroid constraint is 0.25 (adversarial order) due to [19], [12] and [11]. Moreover, we generalize our results to the case of p-matchoid constraints and give a $\frac{1}{p+1}(1 - 1/e^{p+1} - \epsilon - O(1/k))$ approximation using $O(k)$ memory. which asymptotically (as p and k increase) approaches the best known offline guarantee $\frac{1}{p+1}$ [43]. In the next few subsections we introduce the basic notions and definitions that we will use in this thesis.

1.1 Submodular Functions

Submodularity is a property of set functions. Most natural functions that we see in real world satisfy this property [46, 8]. Thus the study of optimization problems for such functions lead to many applications. Submodularity has many similarities to concavity, in other ways it resembles convexity. It has many applications in different areas of computer science such as image segmentation, data summarising, etc.

Submodularity is a property of set functions, i.e., functions $f : 2^V \to R$ that assign each subset $S \subseteq V$ a value $f(S)$. The set V is called the ground set. Also we assume that $f(\emptyset) = 0$, i.e., the empty set has zero value.

The definition of submodulairty is based on a notion of *marginal gain*, which is defined below

Definition 1. *(Marginal Value) For a set function $f : 2^V \to R$, $S \subseteq V$, and $e \in V$, let $\Delta_f(e|S) := f(S \cup \{e\}) - f(S)$ be the discrete derivative of f at S with respect to e.*

Definition 2. *(Submodularity) A function $f : 2^V \to R$ is submodular if for every $A \subseteq B \subseteq V$ and $e \in V \setminus B$ it holds that*

$$\Delta(e|A) \geq \Delta(e|B).$$

Equivalently, a function $f : 2^V \to R$ is submodular if for every $A, B \subseteq V$,

$$f(A \cap B) + f(A \cup B) \leq f(A) + f(B).$$

An important subclass of submodular functions are those which are monotone, the value of function on a set would not decrease by adding new elements to it.

Definition 3. *(Monotonicity) A function $f : 2^V \to R$ is monotone if for every $A \subseteq B \subseteq V, f(A) \leq f(B)$.*

Note that a function f is monotone iff all its discrete derivatives are nonnegative, i.e., iff for every $A \subseteq V$ and $e \in V$ it holds that $\Delta(e|A) \geq 0$.

Throughout this thesis we only consider maximization of monotone submodular functions. There are several different algorithms for minimizing submodular functions. The algorithms for minimization problems are able to find the optimum in polynomial time. So, similar to convex functions, minimization is much easier for submodular functions.

As in the literature, we will assume that we have access to a value oracle for submodular function f. It is a black box that computes $f(S)$ on any input set S. Throughout this thesis, we assume each oracle access can be done in $O(1)$ time.

1.1.1 Examples of submodular function maximization

Influence maximization

We are given a set of target customers in a social network. The question is how can we choose a set S of early adopters and market them in order to generate a cascade of adoptions? Starting with S in a graph $G(V, E)$, we can define influence function $f(S)$ as the expected number of influenced nodes at the end of the process. Now we want to maximize $f(S)$ subject to upper bound for the cost of the initial set S.

$$\begin{aligned} \text{maximize} \quad & f(S) \\ \text{s.t.} \quad & c(S) \leq B, \end{aligned}$$

where $c(\cdot)$ is a cost function defined over subsets of nodes. B is and upper bound on the cost of initial set S.

Theorem 1 (Kempe et al. [33])**.** *In the cascade model defined above, influence maximization is NP-hard to approximate to a factor of $n^{1-\epsilon}$ for any $\epsilon > 0$.*

Feature selection:

More generally, let V be the finite set of discrete random variables. and $F : 2^V \rightarrow R$ be a set function, where $F(A)$ measures the residual uncertainty after we observe $A \subseteq V$. Given, a cost

function $c : 2^V \rightarrow N$ and a budget L, we are interested in computing

$$A^* = \underset{A \subseteq V : c(A) \leq L}{\arg\max} \; F(A).$$

A commonly used criterion for measuring uncertainty is the entropy of a distribution $P : \{x_1, \cdots, x_d\} \rightarrow [0, 1]$,

$$H(P) = -\sum_k x_k P(x_k) \log P(x_k).$$

A major limitation of this approach is that joint entropy is an indirect measure of information: It aims to maximize the uncertainty about the selected variables, but does not consider prediction quality for unobserved variables.

A more direct measure of value of information is the information gain $I(B; A)$, which is defined as

$$I(B; A) = H(B) - H(B|A).$$

The analogous subset selection problem for the information gain is to compute

$$\underset{A \subseteq V : c(A) \leq L}{\arg\max} \; I(V; A).$$

Suppose we are given random variables $Y, X_1, \cdots, X_n$, the goal is to predict Y from subset $X_A = \{X_{i_1}, \cdots, X_{i_k}\}$.

The question is how do we find k most informative features?

$$f(S) = I(Y; XS) = H(Y) - H(Y|XS)$$

f is monotone and submodular if X_i are conditionally independent given Y.

1.1.2 Maximizing Monotone Submodular Functions under Cardinality Constraint

Since the functions we're dealing with are monotone, it is obvious that, for unconstrained maximization problem, the set with maximum value is always the ground set V. However, we will impose different types of constraints. The simplest one is a cardinality constraint - that is, finding the set of size at most k that maximizes the utility. It is well known that maximizing submodular functions is NP-hard even for cardinality constraints. Thus our focus is on developing approximation algorithms for maximizing submodular functions under different constraints.

For monotone submodular maximization under cardinality constraints, in one hand Feige et al. [15] have shown using PCP techniques that unless $P = NP$, there is no algorithm that approximates maximum coverage better than $1 - 1/e$. On the other hand suppose that the submodular function is given to us as as a value oracle. Nemhauser and Wolsey [43] showed that exponentially many queries to the value oracle is required to achieve an approximation algorithm whose performance is better than $1 - 1/e$.

The greedy algorithm

There is a simple algorithm for maximizing a monotone submodular function under cardinality constraint which is called the greedy algorithm. It provides a good approximation algorithm for this problem. It starts with the empty set S_0, and in iteration i, adds the element maximizing the marginal value $\Delta(e|S_{i-1})$.

$$S_i := S_{i-1} \cup \{\arg\max_e \Delta(e|S_{i-1})\}.$$

Nemhauser et al. [43] prove that the greedy algorithm provides a $1 - 1/e$ approximation to the optimal solution of the NP-hard optimization problem.

Since we will refer to this proof later on in this thesis we give the proof here.

Theorem 2 ([43])**.** *For a nonnegative monotone submodular function $f : 2^V \rightarrow R^+$, the greedy algorithm defined above is a $1 - 1/e$ competitive algorithm for maximizing monotone submodular*

functions under cardinality constraint.

Proof. Let $S^* = \{v_1^*, \cdots, v_k^*\}$ be the optimal solution in an arbitrary order. Also let $OPT = f(S^*)$ be the value of optimal solution. Define $S_i := \{v_1^*, \cdots, v_i^*\}$. We show that $f(S_k) \geq (1 - 1/e)OPT$. For every $i = 1, \cdots, k$,

$$\begin{aligned} f(S^*) &\leq f(S^* \cup S_i) \\ &= f(S_i) + \sum_{j=1}^{k} \Delta(v_j^* | \{v_1^*, \cdots, v_{j-1}^*\}) \\ &\leq f(S_i) + \sum_{z \in S^*} \Delta(z | S_i) \\ &\leq f(S_i) + \sum_{z \in S^*} \Delta(v_{i+1} | S_i) \\ &= f(S_i) + k\Delta(v_{i+1} | S_i). \end{aligned}$$

By rearranging we have

$$\Delta(v_{i+1} | S_i) \geq \frac{1}{k}(OPT - f(S_i)).$$

Now define $\delta_i = OPT - f(S_i)$. Therefore,

$$\delta_i - \delta_{i+1} \geq \frac{1}{k}\delta_i$$

$$\delta_{i+1} \leq (1 - 1/k)\delta_i. \tag{1.1}$$

By applying equation 1.1 repeatedly we can show

$$\delta_k \leq (1 - 1/k)^k \delta_0 \leq \frac{1}{e} OPT.$$

Thus

$$OPT - f(S_k) \leq \frac{1}{e} OPT.$$

Hence,

$$f(S_k) \geq (1 - 1/e)OPT.$$

□

1.2 Matroid constraints

We can generalize cardinality constraints to a class of more general constraints, namely *matroid* constraints. A matroid is a combinatorial object satisfying nice properties that are general enough to capture most of natural constraints that we see in optimization problems, such as cardinality constraints (called uniform matroids) and they are also specialized enough that allow us to prove good approximation algorithms for the optimization problems involved these constraints.

Definition 4. *(**Matroids**). A matroid is a finite set system $\mathcal{M} = (\mathcal{N}, \mathcal{I})$, where $\mathcal{N}$ is a set and $\mathcal{I} \subseteq 2^{\mathcal{N}}$ is a family of subsets such that: (i) $\emptyset \in I$, (ii) If $A \subseteq B \subseteq N$, and $B \in I$, then $A \in I$, (iii) If $A, B \in I$ and $|A| < |B|$, then there is an element $b \in B \setminus A$ such that $A + b \in I$. In a matroid $\mathcal{M} = (\mathcal{N}, \mathcal{I})$, N is called the ground set and the members of $\mathcal{I}$ are called independent sets of the matroid. The bases of $\mathcal{M}$ share a common cardinality, called the rank of $\mathcal{M}$.*

In this thesis, we further design algorithms for monotone submodular function maximization subject to p-matchoid constraints. These constraints generalize many basic combinatorial constraints such as the cardinality constraint, the intersection of p matroids, and matchings in graphs. Throughout this section, k would refer to the size of the largest feasible set. A formal definition of a p-matchoid is as follows:

Definition 5. *(**Matchoids**). Let $\mathcal{M}_1 = (\mathcal{N}_1, \mathcal{I}_1), \cdots, \mathcal{M}_q = (\mathcal{N}_q, \mathcal{I}_q)$ be q matroids over overlapping groundsets. Let $\mathcal{N} = \mathcal{N}_1 \cup \cdots \cup \mathcal{N}_q$ and $\mathcal{I} = \{S \subseteq \mathcal{N} : S \cap \mathcal{N} \in \mathcal{I}_\ell, \forall \ell\}$. The finite set system $\mathcal{M}_p = (\mathcal{N}, \mathcal{I})$ is a p-matchoid if for every element $e \in \mathcal{N}$, e is a member of $\mathcal{N}$ for at most p indices $\ell \in [q]$.*

Suppose that the ground set V is partitioned into r parts $P_1, \cdots, P_r$ (e.g. Max-SAT, each part consists of the two possible assignments to each variable). A set is feasible if it contains exactly

one point out of each part. This type of constraint is called a partition matroid of rank r. If all parts contain the same elements then this is the same as optimization over a uniform matroid. In other words, partition matroids generalize uniform matroids.

The greedy algorithm is also guaranteed to provide good solutions for matroid constraints (but not optimal). Suppose (V, I) is a matroid, and we wish to solve the problem

$$\max_{S \in \mathcal{I}} f(S).$$

Suppose the greedy algorithm starts with the empty set S_G and sets $S_G \leftarrow S_G \cup \underset{e \in S_G : S_G \cup \{e\} \in I}{\arg\max} \Delta(e|S_G)\}$ until there is no more e such that $S_G \cup \{e\} \in I$ (i.e., there is no element which can be added to create a feasible solution). The greedy algorithm is guaranteed to return a solution S_G so that $f(S_G) \geq 1/2 \max_{S \in I} f(S)$ [43].

Even for more general constraints called p-matroid constraints, suppose $(V, I_1), \cdots, (V, I_p)$ are p matroids, and $I = \cap_i I_i$. That is, I consists of all subsets of V that are independent in all p matroids. The greedy algorithm generate a solution so that $f(S_G) \geq 1/(p+1) \max_{S \in I} f(S)$. In fact, this results holds even more generally whenever (V, I) is a p-extensible system (a combinatorial notion which generalizes the intersections of p matroids).

Finally [22] present a novel combinatorial algorithm that gives optimal, $1 - 1/e$ approximation algorithm for monotone submodular optimization over a matroid constraint.

1.3 Streaming Submodular Functions

In recent years, submodular optimization has found applications for numerous machine learning and data mining applications including news article recommendation, non-parametric learning, data summarization, network inference, determinantal point processes and influence maximization in social networks [17, 6, 32].

The streaming model is one way to model the problem of analyzing massive data. The model assumes that the data is presented as a stream $(x_1, x_2, \cdots, x_m)$. Real time data like server logs,

user clicks and search queries are modeled by streams. The available *memory buffer* is much less than the size of the stream, so a streaming algorithm must process a stream in a single pass using sublinear space.

In data summarization tasks, the collection of elements is generated continuously, and keeping a real time summary of the the data seen so far is important. Thus, a series of recent papers studied streaming algorithms for maximizing a submodular function. The first work to consider a one-pass streaming algorithm were the work of Badanidiyuru et al. [6], who described a $(1/2 - \epsilon)$-approximation streaming algorithm for maximizing a monotone submodular function subject to a k-cardinality constraint, with a memory of size $O(\frac{1}{\epsilon}k \log k)$. Recently, Kazermi et al. [32] proposed a new one-pass streaming algorithm with the same approximation ratio but with improved memory $O(k)$.

[44] give an upper bound of $1/2 + o(1)$ on the approximation ratio achievable by any algorithm for streaming submodular maximization that only queries the value of the submodular function on feasible sets (sets of size at most k) while using $o(n)$ memory [44]. Consequently, they consider the random order streaming model to go beyond this worst case analysis for the adversarial order inputs. They achieve a $1/2 + 8 \times 10^{-14}$ approximation for maximizing a monotone submodular function in the random order streaming model, using a memory buffer of size $O(k \log k)$. In chapter 2, we substantially improve their result to $1 - 1/e - \epsilon - O(1/k)$ approximation which is close to the best possible guarantee in the offline setting, i.e., $1 - 1/e$ (assuming $P \neq NP$). Furthermore, we improve the required memory buffer to only $O(k)$.

1.4 Secretary Problems

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 [14, 21, 38]. 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 [14]. This algorithm, given by [14], 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 a $1/e$-competitive ratio in expectation.

Many variants and generalizations of the secretary problem have been studied in the literature, see e.g., [4, 50, 48, 51, 35, 5]. [35, 5] 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 [35] gave an algorithm with an asymptotic competitive ratio of $1 - O(1/\sqrt{k})$. Thus as $k \to \infty$, the competitive ratio approaches 1. Recent literature studied several generalizations of this setting to multidimensional knapsacks [41], and proposed algorithms for which the expected online solution approaches the best offline solution as the knapsack sizes becomes large (e.g., [16, 13, 3]).

In another variant of multiple-choice secretary problem, [7] and [27] 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 a value oracle access to the function, i.e., for any given set T, an algorithm can query an oracle to find its value $f(T)$ [49]. 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 [34], 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 [42]. 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.

1.4.1 Our model: secretary problem with shortlists.

In this thesis, 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 se-

lected. 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 [30, 45].

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 $\theta(\log(n))$. 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 size shortlist, we can break the $1/e$ barrier for the secretary problem and achieve a competitive ratio that is arbitrarily close to 1. In this thesis, we will apply to this model to more general problems, namely, submodular k-secretary problemand its extension to matroid constraints. We will also compare this model to the streaming model.

1.5 Matroid Secretary Problem

In the *matroid secretary problem* the elements of a matroid arrive in random order. When an element arrives, the algorithm observes its value and make an irrevocable decision regarding whether or not to accept it. The set of accepted elements should form an independent set of the

matroid. The goal is to maximize the total sum of the value of these elements. [35] presents an $O(\log k)$-competitive algorithm for general matroids. They leave the problem of existence of an $O(1)$ competitive algorithm as an open problem. Currently, the best known algorithm is a $O(\log \log k)$ competitive algorithm [37].

Most of the interesting auctions can be modeled by making the subsets of the bidders to be the independent sets of a matroid. The matroid secretary problem has application in welfare maximizing online mechanism design. where a good is sold to agents arriving online. Here, the agents correspond to the secretaries and they reveal prices that they are willing to pay in an online manner.

For an online algorithm it is usually not possible to achieve exactly the optimum value of its objective function in the worst case. For example, the Vickrey-Clarke-Groves (VCG), which is the most important technique for designing truthful offline mechanisms, can not be applied in most online problems because it requires finding an optimal allocation, which is generally impossible in the online setting. VCG-based online mechanisms are (dominant-strategy) truthful in the rare cases where the underlying allocation problem admits an online algorithm with competitive ratio 1.

It has been shown the random order arrival model for these problems make them easier. For example, for the single-item auction the problem becomes the famous secretary problem. Also it is known algorithms for secretary problem can be transformed into truthful online mechanisms which are constant-competitive for agents with random arrival order [5]. The matroid secretary problem with submodular objective functions has also been considered. For this setting, O(1)-approximations have been found for knapsack constraints, uniform matroids, and more generally for partition matroids for monotone submodular functions [7, 27]. Also when the elements of the ground set are assigned to a set of weights uniformly at random hen a 5.7187-competitive algorithm is possible for any matroid [47]. Furthermore, a $16(1 - 1/e)$-competitive algorithm can be achieved as long as the weight assignment is done at random, even if we assume the adversarial arrival order.

1.6 Our Contribution

Our main result is an online algorithm for 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 approximation 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 $1/e - O(k^{-1/2})$ [34] 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 queary can be done in $O(1)$ time.

Theorem 3. *For any constant $\epsilon > 0$, there exists an online algorithm (Algorithm 8) 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, $\eta_\epsilon(k) = O(2^{poly(1/\epsilon)}k)$. The running time of this online algorithm is $O(n)$.*

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. The running time of our algorithm is linear in n, the size of the input, which 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[40]. 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.

Further, we show an 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 4. *For any constant $\epsilon \in (0,1)$, there exists an algorithm for the submodular random order streaming problemthat 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 [44], which are an approximation ratio of $\frac{1}{2}+8\times10^{-14}$ using a memory of size $O(k \log k)$.

It is natural to ask whether these k-lists 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. Note that the $1 - 1/e$ upperbound for streaming model [39] does not imply an upper bound for the shortlist model.

Theorem 5. *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 6. *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, $\eta_{\epsilon,m}(k) = (2m + 3)\ln(2/\epsilon)$.*

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 [35] asymptotically approaches optimal solution for such functions,

Furthermore for matroid and p-matchoid constraints in chapter 4 We design an algorithm that achieves a $\frac{1}{2}(1-1/e^2-\epsilon-O(1/k))$ competitive ratio for any constant $\epsilon > 0$, using a shortlist of size $O(k)$ for the matroid secretary problem with shortlists. This is especially surprising considering that the best known competitive ratio for the matroid secretary problem is $O(\log \log k)$. We are also able to get a constant competitive algorithm using shortlist of size at most k and also a constant competitive algorithm in the preemption model.

Theorem 26. *For any constant $\epsilon > 0$, there exists an online algorithm (Algorithm 8) for the submodular matroid secretary problem with shortlists that achieves a competitive ratio of $\frac{1}{2}(1 - \frac{1}{e^2} - \epsilon - O(\frac{1}{k}))$, with shortlist of size $\eta_\epsilon(k) = O(k)$. Here, $\eta_\epsilon(k) = O(2^{poly(1/\epsilon)}k)$. The running time of this online algorithm is $O(nk)$.*

Theorem 27. *For the matroid secretary problem in the preemption model, and matroid secretary problem that uses shortlist of size at most $\eta(k) = k$, there is an algorithm that achieves a constant competitive ratio.*

Furthermore, for a more general constraint, namely p-matchoid constraints (defined in section 5.5) we prove:

Theorem 28. *For any constant $\epsilon > 0$, there exists an online algorithm for the submodular secretary problem with p-matchoid constraints that achieves a competitive ratio of $\frac{1}{p+1}(1 - \frac{1}{e^{p+1}} - \epsilon - O(\frac{1}{k}))$, with shortlist of size $\eta_\epsilon(k) = O(k)$. Here, $\eta_\epsilon(k) = O(2^{poly(1/\epsilon)}k)$. The running time of this online algorithm is $O(n\kappa^p)$, where $\kappa = \max_{i\in[q]} rk(\mathcal{M}_i)$.*

The proposed algorithm also has implications for another important problem of submodular function maximization under random order streaming model and matchoid constraints.

Theorem 29. *For any constant $\epsilon \in (0, 1)$, there exists an algorithm for the submodular random order streaming problem with matroid constraints that achieves $\frac{1}{2}(1 - \frac{1}{e} - \epsilon - O(\frac{1}{k}))$ 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 $O(pk + \frac{k^2}{n})$.

Theorem 30. *For any constant $\epsilon > 0$, there exists an algorithm for the submodular random order streaming problem with p-matchoid constraints that achieves $\frac{1}{p+1}(1 - \frac{1}{e^{p+1}} - \epsilon - O(\frac{1}{k}))$ 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 $O(p\kappa + \kappa^p + \frac{k^2}{n})$, where $\kappa = \max_{i\in[q]} rk(\mathcal{M}_i)$.*

Chapter 2: Submodular Secretary Problem with Shortlists on Some Special Classes of Submodular Functions

2.1 Introduction

In this chapter we study the submodular secretary problem with shortlists, which shows that it is not possible to design an asymptotically optimal algorithm for submodular secretary problem with shortlists. In other words there is no online algorithm whose competitive ratio approaches 1 as k tends to infinity. On the other hand for the simplest case of the classic secretary problem (when the submodular function is max function) we show that there are $1 - \epsilon$ approximation algorithm for the secretary problem with shortlists of size $O(\log(1/\epsilon))$. Thus naturally we ask this question: are there any other special family of submodular functions that we can design an assymptotically optimal approximation algorithms for the submodular secretary problem with shortlists. In this chapter we provide two families of submodular functions that we able to approach the optimal competitive ratio 1. The first one is a family that we call it m-submodular functions. It is a generalization of the maximum function in the classic secretary problem and include some other important instances that we discuss. The other family is a family that we call them component-wise monotone submodular functions. It is defined over subsets of real numbers. The value of each subset is a function of real numbers in that subset and increasing each one will increase the value of the function defined on that subset. We briefly explain each function below. In the next to subsection we explain the algorithms that we design for these sub classes.

We study the submodular secretary problem with shortlist on some special family of submodular functions. For some special cases of monotone submodular functions, we show that there are algorithms that asymptotically approach the optimal solution. The first family is the family of functions we call m-submdular functions. A function f is m-submodular if it is submodular and

there exists a submodular function F such that for all S:

$$f(S) = \max_{T \subseteq S, |T| \leq m} F(T). \tag{2.1}$$

Theorem 7. *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, $\eta_{\epsilon,m}(k) = (2m + 3)\ln(2/\epsilon)$.*

Another special case of monontone submodular functions that we consider in this chapter is a 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 [35] asymptotically approaches the optimal solution for such functions. We call this subclass of submodular functions *Component-wise Monotone Submodular Functions*.

2.2 m-submodular functions

We defined m-submodular functions in eq 2.1. There are some examples of m-submodular functions. Note that maximum node weighted bipartite matching and maximum edge weighted bipartite matching defined on $G = (X \times Y)$ with $|Y| = m$ are m-submodular. In this problem, the size Y is fixed to be m. The nodes in X arrive in random order. We need to select k nodes from X in an online manner. At the end the maximum matching (node weighted or edge weighted) between these k nodes and nodes in Y would be the output of the function.

Now consider the following simple greedy algorithm:

Theorem 8. *Suppose S is the output of Algorithm 2 on the input $I = \{a_1, \cdots, a_n\}$ then $F(S) = F(I)$.*

Proof. Suppose S_i is the subset selected at iteration i. Since F is submodular, if $F(S_i \cup \{a_i\}) \leq F(S_i)$

Algorithm 1 Select-If-it-Improves(u)

Inputs: number of items n, items in $I = \{a_1, ..., a_n\}$ arriving sequentially, $u \in \{1, \cdots, n\}$.
$S \leftarrow \emptyset$
for $i = 0$ to n **do**
    if $F(S \cup \{a_i\}) > F(S)$ **then**
        $S \leftarrow S \cup \{a_i\}$
    end if
end for
return $S \setminus \{a_1, \cdots, a_u\}$

then $F(S \cup \{a_i\}) \leq F(S)$. Therefore every $e \in I \setminus S$ has marginal value 0 with respect to S, i.e., $F(S) = F(I)$. □

Theorem 9. $E[|S|] = m \log n$.

Proof. Suppose $F(S_i) = f(T)$, where $|T| = m$. If $a_i \notin T$ then it is not selected. Because if $a_i \notin T$ and is selected then it should have positive F marginal value, which means $F(S_i) = F(S_{i-1} \cup \{a_i\}) > F(S_{i-1}) = f(T)$, it is a contradiction. Thus only elements in T will be selected at position ℓ.

If you consider all permutations of S_i, an element will be selected at position ℓ if it is subset of T, the probability is $|T|/\ell = m/\ell$. Therefore the total expected number of selections will be at most $\sum_{\ell=1}^{n} \frac{m}{\ell} = m \ln n$ □

Theorem 10. *Algorithm 2, with parameter $u = n\epsilon$, selects a set S with $|S| < m \ln(1/\epsilon) + \ln(1/\delta) + \sqrt{\ln^2(1/\delta) + 2m \ln(1/\delta) \ln(1/\epsilon)}$ and $E[F(S)] = (1 - \epsilon - \delta)OPT$ (where OPT is the value of the optimal solution).*

Proof. We use Freedman's inequality. If $\{a_1, \cdots, a_i\}$ has a unique maximum subset of size m, define Y_i to be a random variable indicating whether the algorithm has selected a_i or not, where $Y_i = 1 - \frac{m}{i}$ if a_i is selected and $Y_i = -\frac{m}{i}$ otherwise. If it has no unique solution define $Y_i = 0$. (a_i will not be selected) Also define $\mathcal{F}_i = \{Y_n, Y_{n-1}, \cdots, Y_{n-i+1}\}$.

Let $X_i = \sum_{j=n-i+1}^{n} Y_j$, then $\{X_i\}$ is a martingle, because $E[X_{i+1}|\mathcal{F}_i] = X_i + E[Y_{n-i}|\mathcal{F}_i]$. If $\{a_1, \cdots, a_i\}$ has a unique maximum subset of size m, $E[Y_{n-i}|\mathcal{F}_i] = (m/i)(1 - m/i) + (1 - m/i)(-m/i) = 0$, otherwise $E[Y_{n-i}|\mathcal{F}_i] = 0$. So in both cases $E[X_{i+1}|\mathcal{F}_i] = X_i$. let $L =$

$\sum_{i=n\epsilon}^{n} Var(Y_i|F_{i-1})$.

$$L = \sum_{i=n\epsilon}^{n} \frac{m}{i}\left(1-\frac{m}{i}\right)^2 + \left(1-\frac{m}{i}\right)\left(\frac{m}{i}\right)^2 = \sum_{i=n\epsilon}^{n} \frac{m}{i}\left(1-\frac{m}{i}\right) < \sum_{i=n\epsilon}^{n} \frac{m}{i} = m\ln(1/\epsilon).$$

Therefore using Friedman's inequality

$$Pr(X_{n-n\epsilon} \geq \alpha \text{ and } L \leq m\ln(1/\epsilon)) \leq \exp\left(-\frac{\alpha^2}{2m\ln(1/\epsilon)+2\alpha}\right) < \delta.$$

Thus we get $\alpha > \ln(1/\delta) + \sqrt{\ln^2 1/\delta + 2m\ln(1/\delta)\ln(1/\epsilon)}$. Also $|S| = X_{n-n\epsilon} + m\ln(1/\epsilon)$. Therefore

$$Pr(|S| \geq m\ln(1/\epsilon) + \ln(1/\delta) + \sqrt{\ln^2 1/\delta + 2m\ln(1/\delta)\ln(1/\epsilon)}) \leq \delta.$$

So with probability $(1-\delta)$, $|S| \leq m\ln(1/\epsilon) + \ln(1/\delta) + \sqrt{\ln^2 1/\delta + 2m\ln(1/\delta)\ln(1/\epsilon)}$. Since f is submodular, $E[f(OPT \cap \{a_{n\epsilon}, \cdots, a_n\})] = (1-\epsilon)OPT$. Therefore $E[F(S)] \geq (1-\epsilon)OPT - \delta OPT$. □

Theorem 11. *Any online algorithm for m-submdoular secretary problem needs to select at least* $\frac{1}{2}\log(1/\epsilon) - \frac{1}{2}$ *elements, in expectation, to select the maximum element with probability at least* $(1-\epsilon)$ *in a random permutation. (we assume* $n > 1/\epsilon$*).*

Proof. Let $I_i = \{a_1, \cdots, a_{n/2^{i-1}}\}$, $T_i = \{a_{n/2^i+1}, \cdots, a_{n/2^{i-1}}\}$, and $R_i = I_1 \setminus I_i$, for $i = 1, \cdots, \log(1/\epsilon)$. Suppose M_i is the maximum element in I_i. Let S be the set of selected elements by algorithm at the end of execution. Suppose $\epsilon_i = E[M_i \notin S | M_i \in T_i]$, then $E[|S \cap T_i|] \geq \frac{1}{2}(1-\epsilon_i)$. Therefore $E[|S|] \geq \sum_{i=1}^{\log(1/\epsilon)} \frac{1}{2}(1-\epsilon_i)$. Also w.p. $\frac{1}{2^i}$, $M_1 \in T_i$, thus $\sum_{i=1}^{\log(1/\epsilon)} \frac{1}{2^i}\epsilon_i \leq \epsilon$. (Note that we use the fact $E[M_i \notin S | M_i \in T_i \text{ and } M_i = M_1] \leq \epsilon_i$, i.e, if algorithm selects one element it will select it even if we increase its value and keep the rest unchanged. In other words suppose in the input $a_1, \cdots, a_n$ the algorithm selects a_i. Now if we only increase the value of one item a_i, it wont change the decision of the algorithm regarding selection of a_i) Now $E[|S|]$ is minimized under above constraint if $\frac{1}{2^{\log(1/\epsilon)}}\epsilon_{\log(1/\epsilon)} = \epsilon$ and the rest are zero. Hence $E[|S|] \geq \frac{1}{2}\log(1/\epsilon) - \frac{1}{2}$.

□

Proposition 1. *For a m-submodular function f, any online algorithm needs to select at least $\frac{m}{2}\log(m/\epsilon) - \frac{m}{2}$ elements, in expectation, to select a set S, with $|S| \leq m$ such that $E[f(S)] \geq (1-\epsilon)OPT$, in a random permutation.*

Proof. We apply the previous theorem to m separate 1 to n matching. More precisely, suppose the first node in Y is connected to n nodes in x, the second node in Y is connected to n different nodes in X and so on. Now any matching in this graph will connect each of m nodes of Y to a node in X. Since neighbors of nodes in X are distinct then each would be a 1 to n matching from a node in Y to its n neighbors. So now we can apply previous theorem. □

Theorem 12. *For a general submodular function f, we need to select at least $\Omega(\frac{m}{\epsilon})$ elements, let's say set S, such that $E[\max_{T\subseteq S,|T|\leq m} f(T)] \geq (1-\epsilon)OPT$.*

Proof. Suppose S is the set of elements algorithm selects at the end of execution.

Our construction is based on a simple submodular function, namely, when f is a max coverage function. It is easy to see that f is submodular. There are n elements in the input. In the max coverage instance, each element of input corresponds to a subset of a ground set X. Consider the set cover example above. Suppose there is a set A of size $\frac{1}{\delta}$, also for every $a \in A$, we have a set $T_a = \{a\}$. In addition, we have a set $R = \{b\}$, s.t. $b \notin A$. The optimal solution is $A \cup R$, with size $1/\delta + 1$. Before we observe A in the input order, we can't distinguish R from the rest of $1/\delta$ sets T_a. Furthermore, w.p. 1/2, R shows up before A. Suppose the algorithm selects α fraction of the sets of size 1 that appear before A, in expectation. Then

$$E[R \cap S] = 1/2(1+\alpha).$$

In order to get $(1-\epsilon)OPT$, we should have

$$1 - 1/2(1+\alpha) \leq \epsilon(1+1/\delta).$$

Hence,

$$\alpha > 1 - 2\epsilon(1+\delta)/\delta.$$

The expected number of selections would be $\alpha\frac{1}{2\delta}$. Thus

$$E[|S|] \geq \frac{1}{2\delta} - \frac{\epsilon(1+\delta)}{\delta^2}.$$

Now by setting $\delta = 3\epsilon$

$$E[|S|] \geq \frac{1}{6\epsilon} - \frac{1+3\epsilon}{9\epsilon} \geq \frac{1}{\epsilon}\left(1/3 - \epsilon/3\right).$$

Therefore, when $\epsilon < 1/3$ is $E[|S|] \geq \frac{1}{4\epsilon} = \Omega(\frac{1}{\epsilon})$. □

2.3 Approximating submodular with m-submodular

In this section, we overview a result in [20] related to approximation of m-submdoular functions. Suppose we are given monotone submodular $f : \{0,1\}^n \to [0,1]$. We consider the following algorithm in [20], which gives a method to approximate m-submodular functions (by selecting more than m items).

Definition 6. *(Discrete derivatives). For $x \in \{0,1\}^n$, $b \in \{0,1\}$ and $i \in n$, let $x_i \leftarrow b$ denote the vector in $\{0,1\}^n$ that equals x with i-th coordinate set to b. For a function $f : \{0,1\} \to \mathbb{R}$ and index $i \in [n]$, define $\partial_i f(x) = f(x_{i\leftarrow 1}) - f(x_{i\leftarrow 0})$.*

Algorithm 2 Feldman-Vondrak

```
S = ∅
for i=1 to n do
    if Pr[∂_i f(1_{S(δ)}) > α] > 1/2 then
        S ← S ∪ {a_i}
    end if
end for
return S
```

The following is the summary of the argument in [20].

Lemma 1. *The number of variables chosen by the Algorithm 2 is $|S| \leq \frac{2}{\alpha\delta}$.*

Lemma 2. ***(boosting lemma)*** *If $\delta < 1/2$, then for any $i \in I \setminus S$*

$$Pr[\partial_i f(1_{S(1/2)}) > \alpha] \leq 2^{-1/(2\delta)}.$$

Theorem 13. *$\frac{m}{\epsilon}\log\frac{m}{\epsilon}$ selections, $f(S)$ is within ϵ additive error of OPT, with probability $1-\epsilon$.*

Proof. Call $x_S \in \{0,1\}^S$ bad, if there is $i \in I \setminus S$ such that $\partial_i f(x_S) > \alpha$. If x_S is not bad then $OPT - f(S) \leq m\alpha$. By boosting lemma the probability that x_S is bad is at most $2^{-1/(2\delta)}$. Now set $\delta = \frac{1}{2\log\frac{m}{\epsilon}}$, and $\alpha = \frac{1}{16}\epsilon^2\frac{\log\frac{m}{\epsilon}}{\log\frac{16n}{\epsilon}}$. Note that these parameters are different from [20]. By setting $m\alpha = \epsilon$, we get

$$\begin{aligned} & m\frac{\epsilon^2}{16}\frac{\log\frac{m}{\epsilon}}{\log\frac{16n}{\epsilon}} = \epsilon \\ \iff & 2(\alpha\delta)m\log\frac{m}{\epsilon} = \epsilon \\ \iff & \alpha\delta = \frac{\epsilon/2}{m\log\frac{m}{\epsilon}}. \end{aligned}$$

Therefore,

$$|S| \leq \frac{2}{\alpha\delta} = \frac{m}{\epsilon}\log\frac{m}{\epsilon}.$$

Also the probability that x_S is bad is at most $2^{-1/(2\delta)} = \frac{\epsilon}{m}$. Thus $OPT - f(S) \leq \epsilon$ with probability $\geq 1-\epsilon$. □

2.4 Component-Wise Monotone Subumodular Functions

In this section we provide another class of submodular functions for which we can design an online algorithm for submodular secretary problem that asymptotically approaches the optimal solution. More precisely, we find some assumption for sumbmodular functions under which Kleinberg's algorithm [35] is able to asymptotically approach the optimal solution. The algorithm is originally designed for the case of multiple-choice secretary problem and it asymptotically

achieves the optimal competitive ratio $1 - 1/\sqrt{k}$.

We consider a class of submodular functions that are defined on real-valued elements. In other words, submodular function F defined on a ground set $X \subseteq \mathbb{R}$. i.e., $f : 2^X \to \mathbb{R}$. We can represent F in a simpler way. Let's define $F(x_1, \cdots, x_j) = f(\{x_1, \cdots, x_j\})$. The value of k elements $a_1, \cdots, a_k$ selected from the ground set is $f(\{a_1, \cdots, a_k\})$. We can also represent it by the symmetric function $F(a_1, \cdots, a_k)$.

Now we focus on properties of $F : \mathbb{R}^k \to \mathbb{R}$ (Note the domain of $f : 2^X \to \mathbb{R}$). By making the following assumptions about f, we will show Kleinberg's algorithm asymptotically approaches the optimal solution.

1. f is a monotone submodular function and $F(a_1, \cdots, a_i + \alpha, \cdots, a_k) \geq F(a_1, \cdots, a_i, \cdots, a_k)$, for $\alpha > 0$ and $1 \leq i \leq k$.

It is a reasonable assumption specially in the online auctions, where increasing someone's bid should not decrease the overall valuation function. We claim that the Kleinberg's algorithm works under this assumption which means that the competetive ratio of the algortihm assymptotically approaches 1. Note that in this section we talk about online algorithms and we do not use any shortlist (selections are irrevocable).

Let's first review the Kleinberg's algorithm. Suppose the input sequence is $a_1, \cdots, a_n$, and we want to irrevocably select k elements in an online manner, so as to maximize the total sum of the selected elements. The algorithm recursively divides the input into two halves: It draws a random variable m from binomial distribution $m = B(n, 1/2)$. Recursively select $\ell = \lfloor k/2 \rfloor$ elements from $a_1, \cdots, a_m$. Suppose $y_1 > y_2 > \cdots > y_m$ are the elements in the first half. After observing a_m, select every element which exceeds y_ℓ, until we have selected k items or have seen all elements of S.

Theorem 14. *The Kleinberrg's algorithm [35], described above, achieves a competitive ratio of* $1 - \frac{1}{5\sqrt{k}}$ *for submodular secretary problem, where the submodular function is componenet-wise monotone submodular.*

2.5 Extending Kleinberg's Algorithm to Component-Wise Monotone Subumodular Functions

An immediate consequence of this assumption is that the optimal offline solution is the set of k largest elements in the input (not necessarily unique). Suppose $X = \{x_1 \geq x_2 \geq \cdots \geq x_n\}$. We assume the optimal offline solution is $\{x_1, \cdots, x_k\}$ (not necessarily unique). i.e., $f(\{x_1, \cdots, x_k\}) \geq f(\{x_{j_1}, \cdots, x_{j_k}\})$. The sketch of our approach is to show that the Kleinberg algorithm in fact selects $(1 - 5/\sqrt{k})k$ many elements from top k elements of input, i.e., $x_1, x_2, \cdots, x_k$. Suppose we have two subsets $S, U \subseteq X$. We say $S \geq U$ if $S = \{p_1 \geq p_2 \geq \cdots \geq p_r\}$ and $U = \{q_1 \geq \cdots \geq q_r\}$, and $p_1 \geq q_1, \cdots, p_r > q_r$. Note that because of property (1), $f(S) > f(U)$. We denote by P_S the probability that the algorithm selects all the elements of S from the input. For $S, U \subseteq X$ and $S \geq U$ we show that $P_S \geq P_U$. Therefore the expected value of items selected by the algorithm is at least as much as when we select $(1 - 5/\sqrt{k})k$ many items uniformly at random, in which case because of monotonicity and submodularity we can say that its expected value is at least $(1 - 5/\sqrt{k})f(x_1, \cdots, x_k)$.

Let $T \subseteq S$ denote the k largest elements of S.

Theorem 15. *Let S be any set of n non-negative real numbers. Let T be the k largest elements of S, and $OPT = f(T)$. The expected number of elements of T selected by the algorithm is at least $(1 - 5/\sqrt{k})k$.*

Proof. The proof is by induction on k. Let $y_1 > y_2 > \cdots > y_m$ be the first m samples, and let $z_1 > z_2 > \ldots > z_{n-m}$ be the remaining samples; denote these sets by Y and Z respectively. Conditional on the event $|Y \cap T| = r$, the expected number of the top $\ell = k/2$ elements of $Y \cap T$ is bounded below by

$$\sum_{r=1}^{k} Pr(|Y \cap T| = r) \cdot (\min(r, l)/k)k \geq (1 - \frac{1}{2\sqrt{k}})\frac{k}{2}.$$

Thus the expected number of elements selected from $Y \cap T$ is at least $(1 - 5/\sqrt{k/2})(1 -$

$1/2\sqrt{k})(k/2)$, by the induction hypothesis.

Similar to Kleinberg define the random variable q which counts the number of elements of Z exceeding y_ℓ. Let q_i be the number of elements of Z whose value lies between y_i and y_{i-1}. The q_i are stochastically dominated by i.i.d. geometrically distributed random variables each having mean 1 and variance 2. Thus their sum $q = \sum_{i=1}^{\ell} q_i$ satisfies $E[|q - \ell|] \leq \sqrt{k}$. Let $r = |q - \ell|$. The expected number of elements algorithm selects from $Z \cap T$ is at least $\ell - r$ (if $y_\ell \in T$ the argument is similar to Kleinberg if $y_\ell \notin T$ Then all the elements of $Z \cap T$ will be selected). Removing the conditioning on r and recalling that $E(r) \leq \sqrt{k}$, the algorithm selects a subset of Z with expected size $\ell - \sqrt{k}$. Combining this with the above paragraph will show that the expected number of elements selected from $Y \cup Z$ is $(1 - 5/\sqrt{k})k$.

□

If the algorithm could select these subsets uniformly at random then because of submodularity the expected value of function f that the algorithm selects from T will be $(1 - 5/\sqrt{k})OPT$. The next lemma will prove the expected value of selected elements by the algorithm is at least as much as uniform case.

Lemma 3. *If $T = \{a_1, a_2, \cdots, a_k\}$ and $a_1 > a_2 > \cdots > a_k$, the probability that the algorithm selects a_i is larger than the probability it selects a_j for $i < j$.*

Proof. Let's fix the sets Y and Z but not the ordering of elements in Z. The algorithm selects all the elements of Z greater than y_ℓ until it selects ℓ elements. So among all different possible permutations for Z, the algorithm will select the first ℓ elements of Z greater than y_ℓ. If the total number of these elements, t, is less than or equal to ℓ then regardless of permutation of elements in Z we select the same subset of Z (the t largest elements of Z).

We will see that which elements of $T \cap Z$ will be missed by the algorithm, and show that the smaller an element is the larger the probability of missing that item is. Suppose $a \in T \cap Z$, it will be missed by the algorithm if either $a \leq y_\ell$ or $a \geq y_\ell$ but there are ℓ elements larger than y_ℓ appearing before a (after selecting ℓ elements the rest are truncated).

If $a, b \in Z$ and $a > b$ the probability that we miss a for the first reason is less than the probability that we miss b for the first reason. Also if both a and b pass the condition of second case, i.e., $a > y_\ell$ and $b > y_\ell$ then they are both equally likely to be missed by algorithm (only depend on their position in the the input). So b is more likely to be missed by the algorithm.

If $a, b \in Y$ then by induction you can show the probability that a is selected is larger than the probability that b is selected.

Now consider the case that one of a or b is in Y and the other is in Z. The probability that $a \in Y, b \in Z$ is the same as the probability $b \in Y, a \in Z$. By switching the place of a and b and fixing the rest, the probability that a is selected in Y is larger than the probability b is selected because if b is larger than threshold, a is too. If a is missed for truncation, b will also be truncated if we replace it in the same place as a is in Y. For the Z part, if $a \in Y, b \in Z$, the threshold y_ℓ is larger than or equal the case $b \in Y, a \in Z$. Thus if b is not missed in Z with the larger threshold, a which is larger than b will not be missed with the smaller threshold.

Hence the probability that the algorithm misses a_i is less than or equal the probability that it misses a_{i+1}.

□

Lemma 4. *Assuming the expected number of elements that the algorithm selects is* $(1 - 5/\sqrt{k}).k$*, and* $p_1 > p_2 > \cdots > p_k$ *are respectively the probability that each element of* $T = \{a_1 > \cdots > a_k\}$ *is selected, then* $E[f(S)] > (1 - 5/\sqrt{k})OPT$*, where* S *is the subset of* T *selected by the algorithm.*

Proof. First by induction on k, we prove that the probability distribution Π over subsets of $T = \{a_1, \cdots, a_k\}$ that minimizes $E_{\Pi, S \in T} f(S)$, is the following distribution: $\Pi(\{a_1, \cdots, a_t\}) = (p_t - p_{t+1})$, $1 \leq t \leq k$. Suppose $p_{k+1} = 0$, and $P(\emptyset) = 1 - p_1$.

For $k = 1$, $\Pi(\{a_1\}) = p_1$ and $\Pi(\emptyset) = 1 - p_1$. Hence $E_\Pi[f(S)] = p_1 f(\{a_1\}) = p_1.OPT$, which is the only option.

Now we want to show that the above distribution is a minimizer for T. By projecting Π to subsets of $T' = \{a_1, \cdots, a_{k-1}\}$, say Π' , we have $\Pi'(S) = \Pi(S) + \Pi(S \cup \{a_k\})$, $\forall S \subseteq T'$. The marginal probabilities of elements in T' are $p_1 > \cdots > p_{k-1}$. By induction the Π' is the minimizer

of $E_{\Pi',S\in T'}[f(S)]$. Now we show Π is minimizer of for T. Consider a different distribution Θ. Suppose Θ' is its projection to T'.

$$E_{\Theta,S\in T} f(S) = \sum_{S\subseteq T} \Theta(S) f(S) = \sum_{S\subseteq T'} (\Theta(S) f(S) + \Theta(S\cup\{a_k\}) f(S\cup\{a_k\}))$$

$$= E_{\Theta',S\in T'} f(S) + \sum_{S\subseteq T'} \Theta(S\cup\{a_k\})(f(S\cup\{a_k\}) - f(S))$$

$$\geq E_{\Pi',S\in T'} f(S) + p_k(f(T) - f(T')) = E_{\Pi,S\in T} f(S).$$

Therefore Π is a minimizer for T. (the last inequality is because of submodularity and induction hypothesis)

Now we lowerbound $E_{\Pi,S\in T} f(S)$.

$$E_{\Pi,S\in T} f(S) = \sum_{S\subseteq T} \Pi(S) f(S) = \sum_{t=1}^{k} f(\{a_1,\cdots,a_t\})(p_t - p_{t+1})$$

$$\geq \sum_{t=1}^{k} \frac{t}{k} OPT(p_t - p_{t+1}) = \frac{OPT}{k} \sum_{t=1}^{k} p_i.$$

Thus the expected value of the output of Kleinberg's algorithm is at least $\frac{OPT}{k}(\sum_{i=1}^{k})p_i = (1 - 5/\sqrt{k})OPT$

□

2.6 Upperbounds for Online Setting

In this section we give an upper bound on the competitive ratio of any online algorithm for general submodular secretary problem with a cardinality constraint. Our bound is an information theoretic bound without any computational complexity assumption and it is only based on the online nature of the problem. Note that in Chapter 4 we generalize this bound for the shortlist model.

Our construction is based on a very simple submodular function, namely, when f is a max

coverage function. It is easy to see that f is submodular. There are n elements in the input and f is defined to be 0 on $n - 2k + 1$ elements and non-zero on $2k - 1$ elements, say $a_1, \cdots, a_{2k-1}$. In the max coverage instance, each element of input corresponds to a subset of a ground set X. For $1 \leq i \leq n$, a_i covers $\{b_i\} \subseteq X$, and a_{2k-1} covers a random subset of $X = \{b_1, \cdots, b_{2k-2}\}$ with size $k - 1$, say $\{b_1, \cdots, b_{k-1}\}$.

The optimal offline solution selects $\{a_k, \cdots, a_{2k-2}, a_{2k-1}\}$, with size $2(k - 1)$. We prove an upper bound on the expected value that any online algorithm can achieve. We even assume the algorithm is aware of the size of the input elements, i.e., $f(a_i)$. But it does not help the algorithm as opposed to the real valued functions in the last section.

Theorem 16. *No online algorithm can achieve competitive ratio better than* $7/8$ *in the above max coverage instance, assuming the input arrives uniformly at random.*

Proof. If algorithm does not select a_{2k-1}, it will cover at most k elements from X. Hence the competitive ratio will be at most $\frac{k}{2k-2} \sim 1/2$. So suppose the algorithm will select a_{2k-1}. Note that before we see a_{2k-1} in the input we don't know what elements of $\{b_1, \cdots, b_{2k-2}\}$ are going to be covered by a_{2k-1}. Therefore, whatever the algorithm selects from the input before seeing a_{2k-1}, will be later covered by a_{2k-1} w.p. $(k - 1)/(2k - 2) = 1/2$.

Conditional on the position of a_{2k-1} in the input, say ℓ, the expected number of elements in the set $X \setminus a_{2k-1}$ in the first ℓ sets is $\frac{\ell}{n}(k - 1)$. So the expected number of elements algorithm selects is $\frac{1}{2}\frac{\ell}{n}(k - 1) + (k - 1) + (1 - \frac{\ell}{n})(k - 1) = (2 - \ell/2n)(k - 1)$. Now uncondition on ℓ, the expected number of elements that the algorithm selects is $\frac{1}{n}\sum_{\ell=1}^{n}(2 - \ell/2n)(k - 1) = (k - 1)(7/4)$. So the competitive ratio is $7/4(k - 1)/2(k - 1) = 7/8$. □

2.7 Random Order Online Matching

Suppose we are given a bipartite graph $G = (X \times Y, E)$, and we receive elements of X in an online manner. Note that in contrast with the online matching problem, we do not select edges of the matching, instead we select a subset of vertices in X and find their corresponding maximal matching.

Theorem 17. *Consider the following simple algorithm: select an element iff it increases the size of current matching by one. Then this online algorithm will select all the elements of the optimal offline solution (and possibly more).*

Proof. If the algorithm selects k elements then it is optimal. Thus, suppose the algorithm selects $k' < k$ elements. Every rejected element u can not improve the maximum matching on selected elements before u. We want to show that u will not be in the optimal solution. For a subset S of elements, suppose $M(S)$ is the maximum matching of $S \times Y$.

Suppose T is the subset of elements selected by the algorithm before observing u, and $|M(T)| = |M(T \cup \{u\})|$ are equal (there is no augmenting path starting from u using $M(T)$). Now we show that for any $S \supseteq T$, also $|M'(S \cup \{u\})| = |M'(S)|$. It is clear that $N(U) \subseteq M'(T)$, and there is no augmenting path on $M(T)$ starting uv for $v \in N(U)$. So there is no augmenting path that includes uv for $M'(S)$ as well. (Otherwise you could have projected it to one for T, i.e., restrict $M'(S)$ to T, say $M'(T)$. $v \in M'(T)$ and cut the augmenting path when it meets the first vertex out of $M'(T)$, it will give a larger matching for $T \cup \{u\}$).

So every vertex u which has not been selected at some point will never create an augmenting path with respect to the maximum matching that the algorithm will create later in the algorithm. Therefore at the end of the algorithm we can not find any augmenting path starting from vertices that are not selected. □

2.8 Random Order Node Weighted Online Matching

We have a bipartite graph $G = (X \times Y, E)$. The elements of Y is given upfront and vertices of X are arriving in an online manner. Each vertex of $x \in X$ has a type $t_x \subseteq Y$, which is represented as a subset of Y, and a weight $w \in \mathbb{R}$. We have to select k vertices of X, say S with maximum node wighted matching, i.e., with maximum total weight of matched nodes in S. We first consider this regime that $n = |X|$ and k tends to infinity while $m = |Y|$ is fixed. We provide optimal asymptotic algorithm for this case. In the next section we prove impossibility result for the case that m is not fixed.

2.8.1 m is fixed

The algorithm is simple, for every $y \in Y$, we reserve k/m many selections for those vertices in X that are connected to y. Tentatively, for each $y \in Y$, by k/m selections reserved for y, we select the top m vertices among neighbors of y, $N(y)$. There is a asymptotic optimal algorithm for that with rate $1 - 1/\sqrt{k/m}$, similar to the Kleinberg algorithm [35] and the generalization described in the previous section. Let's call the algorithm $top(m, y, k/m)$. The algorithm needs to be aware of $N(y)$. Thus, in the beginning we sample $n^{1/3}$ of the input to estimate $N(y)$. After observing one element $x \in X$, for every $y \in t_x$, run the algorithm $top(m, y, k/m)$ for the input x and if any of them selects x, the algorithm will select x. We give pseudocode in Algorithm 3.

Algorithm 3 NodeWeightedOnlineMatching

```
sample the first n^{1/3} elements to estimate N(y) for each y ∈ Y
for each x ∈ X do
    for y ∈ t_x do
        if top(m, y, k/m) selects x then
            S ← S ∪ {x}
        end if
    end for
end for
Return S
```

Theorem 18. *Algorithm 3 achieves competitive ratio* $1 - o(1)$ *for the node weighted online matching problem defined above.*

Proof. Suppose the algorithm has selected the top m neighbors of each $y \in Y$. Then the maximum matching corresponding to the set S selected by the algorithm is optimal. In other words if the optimal offline matching matches y to $M(y)$ then $M(y)$ is among the top m neighbors of y. If the number of neighbors of y, $N(y)$ goes to infinity then the algorithm $top(m, y, k/m)$ will select $M(y)$ with probability at least $1 - 1/\sqrt{k/m}$. If $N(y)$ does not go to infinity then for large enough k, $k/m > N(y)$, thereofor we can select all neighbors of y. The only obstacle is to estimate $N(y)$ for each $y \in Y$, as it is required in $top(m, y, k/m)$. We use sampling to estimate $N(y)$. □

2.8.2 m is not fixed

In this section, we assume m is not fixed and provide upper bound which refutes an asymptotic optimal algorithm.

Theorem 19. *There is no online algorithm that achieves better than* $5/6$ *competitive ratio when m is not fixed.*

Proof. Consider the following example. Every type t is set of size one in Y, $m = k$ and every $y \in Y$ has two neighbors one with weight W and the other with weight $W + \omega$ w.p. $1/2$ or $W - \omega$ w.p. 1/2. Assume $\omega < W$. The optimal offline solution has weight $Wk + 1/2k\omega$. Now in the online algorithm, for each $y \in Y$, the expected weight matched to y is W. Therefore the expected weight of online algorithm is W. If the neighbor with weight W appears first in the input and the algorithm selects it, the weight that is matched to y is W. If it does not select the first one and select the other neighbor of y, the expected would still be $1/2(W + \omega) + 1/2(W - \omega) = W$. If the algorithm selects the first one and the second one is $W + \omega$ and the algorithm selects it, the associated weight to y is $W + \omega$. So the weight per element of these two selected elements is $(W + \omega)/2 = W/2 + \omega/2 < W$. So if the one with weight W appears first the weight per element of each of the selection is less than or equal to W so the total expected weight of selected elements is at most $Wk/2$. For those y that the neighbor with weight appears last the the expected weight per elements is $1/2(W + \omega + W) = W + \omega/2 < 3/2W$. So the total expected weight for these ys is $k/2(3/2W) = 3/4Wk$. Hence the competitive ratio is at most $(Wk/2 + 3/4Wk)/(Wk + 1/2k\omega) < 5/6$. □

Chapter 3: Cardinality Constraints: A Minmax $1/2 - \epsilon$ Approximation using Shortlist of Size $O(k \log^2 k)$ for the Submodular Secetary Problem

3.1 Introduction

In this section we study the submodular secretary problem with shortlists in the most general case of monotone submodular functions. Currently, the best result for submodular secretary problem is due to [34], 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 [42]. 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. 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.

Our main result in this chapter is an online algorithm for submodular k-secretary problem with shortlists that, we design a $1/2 - O(k^{-1/4})$ approximation algorithm for submodular secretary problem with shortlists. The size of the shortlist is $O(k \log^2 k)$. The algorithm is as described in Algorithm 4.

Note that for submodular k-secretary problem there is an upper bound of $1 - 1/e$ on the achievable approximation 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 $1/e - O(k^{-1/2})$ [34]. We assume the standard value oracle model: the only access to the submod-

ular function is through a black box returning $f(S)$ for a given set S, and each such queary can be done in $O(1)$ time.

The main theorem in of this chapter is as follows:

Theorem 20. *There exists an online algorithm (Algorithm 4) for the submodular k-secretary problem with shortlists that achieves a competitive ratio of $1/2 - O(k^{-1/4})$, using shortlist of size $\eta(k) = O(k \log^2 k)$.*

A line of papers studied streaming algorithms for maximizing a submodular function. The first one-pass streaming algorithm for maximizing a monotone submodular function subject to a k-cardinality constraint is due to [6], who propose a $(1/2-\epsilon)$-approximation streaming algorithm, with a memory of size $O(\frac{1}{\epsilon}k \log k)$.

The result in this section improves the required memory of the previously best known algorithm for the streaming setting [6] in the sense that it removes dependency on ϵ, but it achieves the same approximation ratio. We compare this algorithm with previous algorithms in the literature at the end of this chapter. Throughout this section, we also make this assumption that all the marginal values w.r.t. all the subsets are different.

We significantly improve the result presented in this chapter in the next chapter. We designed this approximation algorithm before the work of [44, 32] which improved the previously best known approximation of $1/2$.

3.2 The Algorithm

The algorithm divides the input into k equal intervals $I_1, \cdots, I_k$. The elements that the algorithm selects consists of two sets S and B. The set S has exactly k elements $\{a_1, \cdots, a_k\}$, $a_i \in I_i$. We denote the first i elements of S by S_i. We select a_i with the property $a_i = \arg\max_{e \in I_i} \Delta(e|S_{i-1})$. Also define $M_i = \max_{e \in I_i} \Delta(e|S_{i-1})$. Note that we can find maximum element using the online algorithm in the last section with error ϵ and $\log(1/\epsilon)$ many selections.

The set B consists of k subsets $B = B_1 \cup \cdots \cup B_k$. We call B_i the elements *blocked* by I_i. We *mark* B_i once its size exceeds $k \log k$ ($|B_i| > k \log k$). An element $a \in I_i$ is blocked if $\Delta(a|S_{i-1}) > th$,

Algorithm 4 Submodular Secretary

$I \leftarrow I_1 \cup I_2 \cup \cdots \cup I_k$ (divide input into k equal intervals)
$th \leftarrow \infty$
$S_i \leftarrow \emptyset, i = 0, \cdots, k$
for $j = 1$ to k **do**
    for $a \in I_j$ **do**
        if $F(S_{j-1} \cup \{a\}) - F(S_{j-1}) \geq th$ **then**
            $B_{thi} \leftarrow B_{thi} \cup \{a\}$
            if $|B_{thi}| > k \log k$ **then**
                mark B_{thi}
                $thi \leftarrow argmin\{M_i | B_i \text{ is not } marked\}_{i=1}^{j-1}$
                $th \leftarrow M_{thi}$
            end if
        end if
    end for
    $S_j = S_{j-1} \cup \text{argmax}_{a \in I_j} F(S_{j-1} \cup \{a\})$
(can be done with an online algorithm with error ϵ using $\log(1/\epsilon)$ selection)
    $M_j = F(S_j) - F(S_{j-1})$
end for
$B \leftarrow \cup_{i=1}^{k} B_i$
$OPT \leftarrow$ optimal offline solution
$B' \leftarrow B \cap OPT$
if $F(S_k) \geq OPT/2$ **then**
    return S_k
else
    return $S_{k-|B'|} \cup B'$
end if

where th is a threshold defined as $th = \min_{j<i}\{M_j | B_j \text{ is not marked}\}$. (we will later use the fact that for every element $a \in I_i$ that is not blocked, its marginal value w.r.t. S is less than every M_j, for $j < i$). If $a \in I_i$ is blocked we put a in B_{thi} where $thi = argmin_{j<i}\{M_j | B_j \text{ is not marked}\}$. If $|B_{thi}|$ becomes larger than $k \log k$, we *mark* B_{thi}.

At the end of the input suppose OPT is the optimal offline solution. If $F(S) > OPT/2$, we return S, otherwise we return $S_{k-|B'|} \cup B'$, where $B' = B \cap OPT$.

3.3 Analysis of the Algorithm 4

Lemma 5. *If $(a_0, \cdots, a_{n-1})$ is a random order of $\{1, \cdots, n\}$, and a_d is the first element with $a_d < a_0$ (or $a_d = n - 1$ if $a_0 = n$), and we have $S = \sum_{i=0}^{n-1} x_i$ is uniformly distributed on $\{x_i | a_i \geq a_0\}$, i.e.,*

$x_i = 0$ *if* $a_i < a_0$. *Then*

$$E[\sum_{i=0}^{d} x_i] = \frac{S}{n} O(\log n).$$

Proof. With probability $1/n$, $a_0 = i$, for $i = 1, \cdots, n$, and given $a_0 = i$, $E[d_i] = \frac{n-1}{a_0}$. Also the total value of S is uniformly distributed among $n - a_0$ variable x_i, so $E[\sum_{i=0}^{d} x_i | a_0 = i] = \frac{S}{n-a_0}$. Thus,

$$E[\sum_{i=0}^{d} x_i] \leq \frac{1}{n}\left(\sum_{a_0=1}^{n-1} \frac{n-1}{a_0}\frac{S}{n-a_0} + n\right) \leq \frac{S}{n}\left(\sum_{a_0=1}^{n-1}(\frac{1}{a_0} + \frac{1}{n-a_0}) + n\right) = \frac{S}{n} O(\log n).$$

□

Lemma 6. $E[|B|] = O(k \log^2 k)$.

Proof. Suppose th_j is the threshold th at iteration j, and $th_j = M_\ell$, for some $\ell < j$. Consider the marginal value of all elements in $I_\ell \cup \cdots \cup I_k$ w.r.t. $S_{\ell-1}$ and among them select the $(k - \ell) \log k$ elements with the largest marginal value $\Delta_F(e|S_{\ell-1})$. Denote these elements by T_ℓ. The probability that none of the elements in T_ℓ appears in I_ℓ is

$$(1 - 1/(k - \ell))^{(k-\ell)\log k} \leq (1/e)^{\log k} \leq 1/k.$$

Thus, the expected number of B_i's who are *marked* by the algorithm is 1. For a given interval i, suppose j is the first interval after i, satisfying

$$\max_{a \in I_j} \Delta_F(a|S_{i-1}) < M_i, \text{ and } M_i \text{ is not } \textit{marked}.$$

Define $d_i = j - i$. Suppose $S_k = \{a_1, \cdots, a_k\}$ such that $a_i = S_i \setminus S_{i-1}, \forall 1 \leq i \leq k$. Because $\max_{a \in I_j} \Delta_F(a|S_{i-1}) < M_i$, we have

$$\Delta_F(a_j|S_{i-1}) < M_i.$$

Now since F is submodular

$$\Delta_F(a_j|S_{j-1}) < M_i, i.e., M_j < M_i.$$

So

$$B_i \subset I_i \cup \cdots \cup I_j.$$

Moreover from lemma 5, and by setting $x_\ell = |T_i \cap I_{i+\ell}|$ in the lemma, we get

$$E[|(I_i \cup \cdots \cup I_j) \cap T_i|] = \frac{(k-i)\log k}{k-i} \log k = \log^2 k.$$

As a result $E[|B_i|] \leq \log^2 k$, since $B_i \subseteq (I_i \cup \cdots \cup I_j) \cap T_i$. In addition,

$$E[|B|] = \sum_{B_i \text{ is not marked}} E[|B_i|] + \sum_{B_i \text{ is marked}} E[|B_i|]$$

$$\leq k \log^2 k + k \log k = O(k \log^2 k).$$

□

Theorem 21. $|B| = O(k \log^2 k)$*, w.h.p.*

Proof. We use Freedman inequality. For $i = 1, \cdots, k$, define $\mathcal{F}_i = I_1 \cup \cdots \cup I_i$, for $1 \leq i \leq k$. Also define $Y_i = |B_i| - E[|B_i||\mathcal{F}_{i-1}]$, based on the algorithm we know $|Y_i| \leq k \log k$. Let $X_i = \sum_{j=1}^{i} Y_j$, for $1 \leq i \leq k$, then $\{X_i\}$ is a martingle, because

$$E[X_{i+1}|\mathcal{F}_i] = E[Y_{i+1} + X_i|\mathcal{F}_i] = E[Y_{i+1}|\mathcal{F}_i] + E[X_i|\mathcal{F}_i] = 0 + X_i.$$

As in Freedman's inequality, let $L = \sum_{i=1}^{k} Var(Y_i|\mathcal{F}_{i-1})$.

$$L \leq \sum_{i=1}^{k} \frac{1}{n-i}\left(1 - \frac{1}{n-i}\right)(n-i)\log k = \sum_{i=1}^{k}\left(1 - \frac{1}{n-i}\right)\log k < k \log k.$$

(Since each of $(n-i)\log k$ elements of T_i will be in I_i, with probability $\frac{1}{n-i}$ independently). Therefore,

$$Pr(X_k \geq \alpha \text{ and } L \leq k \log k) \leq \exp\left(-\frac{\alpha^2}{2k \log k + 2\alpha k \log k}\right) < \delta$$

Thus we get $\alpha > k \log k \log(1/\delta) + \sqrt{(k \log k \log(1/\delta))^2 + 4}$. Also $|B| \leq X_k + \sum_{i=1}^{k} E[|B_i||\mathcal{F}_{i-1}] = X_k + k \log^2 k$. Therefore,

$$Pr(|B| > k \log^2 k + 2k \log k \log(1/\delta)) < \delta.$$

□

Lemma 7. *The number of elements marked by the algorithm is at most $k^{3/4}$ w.h.p.*

Proof. Let X_i be the random variable indicating whether $T_i \cap I_i = \emptyset$. Also $X_1, \cdots, X_k$ are independent. Therefore by Bernstein inequality,

$$P\left(\frac{1}{k}|\sum_{i=1}^{k}(X_i - 1/k)| > \epsilon\right) < 2\exp\left(-\frac{k\epsilon^2}{2(1+\epsilon/3)}\right).$$

So

$$P\left(\sum_{i=1}^{k} X_i > 1 + k^{3/4}\right) < 2\exp\left(-\frac{\sqrt{k}}{2(1+1/(3k^{1/4}))}\right) < 2\exp(-k^{1/4}).$$

□

Lemma 8. *If $a \in OPT \setminus B$, and $a \in I_j$ then $\Delta_F(a|S_{j-1}) \leq M_i, \forall i < j$ that M_i is not marked.*

Proof. Since $a \notin B$ and $a \in I_j$, we have

$$F(S_{j-1} \cup \{a\}) - F(S_{j-1}) \leq \min_{1 \leq i < j} \{M_i | M_i \text{ is not marked}\}.$$

□

Lemma 9. *Let $O_j = |OPT \cap (I_1 \cup \cdots \cup I_j)|$, then w.h.p., $||O_j| - j| \leq k^{3/4}$, for $j > k^{3/4}$.*

Proof. Suppose $OPT = \{a_1, \cdots, a_k\}$. Consider one interval I of size ℓ. Let's X_i be a random variable indicating whether $a_i \in I$ or not, and define $S_k = \sum_{i=1}^{k} X_i$. Thus $E[S_k] = \frac{|I|}{n}k$. From Hoeffding's inequality,

$$Pr[|S_k - E[S_k]| > t] < 2\exp\left(-\frac{2t^2}{k}\right).$$

$$Pr[|S_k - \frac{|I|}{n}k| > k^{3/4}] < 2\exp\left(-\frac{2k^{3/2}}{k}\right) = 2e^{-\sqrt{k}}.$$

So for $k > \log^2(\epsilon/2)$, w.p. $1-\epsilon$, $||O_j| - j| \leq k^{3/4}$. □

Lemma 10. *For $A, B, C \subseteq dom(F)$,*

$$\Delta_F(A|B) \leq \Delta_F(A|C) + \Delta_F(C|B)$$

Proof.

$$\begin{aligned}
&\Delta_F(A|C) + \Delta_F(C|B) \\
=&F(A \cup C) - F(C) + F(C \cup B) - F(B) \\
\geq&F(A) - F(C \cup B) + F(C) - F(B) \\
\geq&F(A \cup C \cup B) - F(B) \\
\geq&F(A \cup B) - F(B) \\
\geq&F(A \cup B) - F(B) \\
=&\Delta_F(A|B).
\end{aligned} \tag{3.1}$$

□

There is another folklore lemma that we will use:

Lemma 11. *For $A, B \subseteq dom(F), \Delta_F(A|B) \leq \sum_{a \in A} \Delta_F(a|B)$.*

Theorem 22. *Algorithm 4 is a $(1/2 - O(k^{-1/4}) - \epsilon)$ competitive algorithm.*

Proof. Suppose R is the output of algorithm. If $R = S_k$ then $F(R) \geq OPT/2$. So suppose $R = S_{k-|B'|} \cup B'$ and $F(S_k) < OPT/2$. We remove the elements of OPT in the first $k^{3/4}/k$ fraction of the input. Suppose $OPT' = OPT \setminus (I_1 \cup \cdots \cup I_{k^{3/4}})$. Note that $\frac{k^{3/4}}{k} \to 0$ as $k \to \infty$. Also $E[OPT'] = (1 - k^{-1/4})OPT$. Applying lemma 9 on $I = \{I_1, \cdots, I_\ell\}$ for $\ell = k^{3/4}, k^{3/4} + 1, \cdots, k$ and then applying a union bound on all of them gives with probability $1 - 2ke^{-\sqrt{k}}$, $|OPT \cap \{I_1 \cup \cdots \cup I_j\}| - j < k^{3/4}$ for all $j > k^{3/4}$. i.e., $|OPT' \cap (I_1 \cup \cdots \cup I_j)| < j, \forall j > k^{3/4}$, w.h.p.

Now the idea is to match the elements of $OPT \setminus B$ to elements of $S_{k-|B'|}$. We can match $a \in OPT \setminus B$, $a \in I_j$, to $a_i \in S_k$ if

$$\Delta_F(a|S_{j-1}) \leq M_i,$$

then Lemma 8, implies that a can be matched to any a_i, $i < j$. Also lemma 9 implies that if we remove the first $k^{3/4}$ elements, w.h.p. all the elements of OPT' can be matched to $S_{k-|B'|}$. From lemma 12

$$\Delta_F(OPT \setminus B|S_k) \leq \sum_{a \in OPT \setminus B} \Delta_F(a|S_k) \leq \sum_{a \in OPT \setminus B, a \in I_j} (F(S_{j-1} \cup \{a\}) - F(S_{j-1})) \leq \sum_{a \in OPT \setminus B, a \in I_j} M_j.$$

Also because of lemma 8, we have

$$\sum_{a \in OPT' \setminus B, a \in I_j} M_j \leq \sum_{i=1}^{|OPT' \setminus B|} M_i \leq \sum_{i=1}^{|OPT \setminus B|} M_i.$$

Hence

$$\Delta_F(OPT \setminus B|S_k) \leq \sum_{i=1}^{|OPT \setminus B|} M_i.$$

Now

$$\Delta_F(OPT|R) = \Delta_F(OPT \setminus B|R) \leq \Delta_F(OPT \setminus B|S_k) + \Delta_F(S_k|R) \leq$$

$$\sum_{i=1}^{|OPT \setminus B|} M_i + \sum_{i=|OPT \setminus B|+1}^{k} M_i = F(S_k) < OPT/2.$$

The first equality is true because $OPT \cap B \subseteq R$. The penultimate inequality is because of lemma 10. Therefore

$$\Delta_F(OPT|R) = F(OPT \cup R) - F(R) < OPT/2.$$

So

$$F(R) > F(OPT \cup R) - OPT/2 \geq F(OPT) - OPT/2 = OPT/2.$$

Note that the total loss incurred above is $\left(k^{-1/4} + 2e^{-k^{1/4}} + k^{-1/4} + 2ke^{-\sqrt{k}} + \epsilon\right) OPT = O(k^{-1/4} + \epsilon)OPT$. □

Proposition 2. *Algorithm 4 achieves* $1/2 - O(k^{-1/4})$ *competitive ratio, by selecting* $O(k \log^2 k)$ *many elements in expectation.*

Proof. By setting $\epsilon = k^{-1/4}$. □

Proposition 3. *Algorithm 4 achieves* $1/2 - O(k^{-1/4})$ *approximation random order streaming algorithm, with memory of size* $O(k \log^2 k)$ *in expectation. The running time is polynomial in n and exponential in k.*

Proof. Among elements of $S_k \cup B$ with size $O(k \log^2 k)$, return $\max_{0 \le i \le k, C \subseteq B, |C| = k-i} F(S_i \cup C)$. □

3.4 Comparison

There are two main frameworks for submodular secretary problem with cardinality constraints: online algorithms and streaming algorithms. In the online framework, we assume the input has random order. All of the previous algorithms in this framework irrevocably select a subset of size k, whose value is close to OPT. The best known algorithm so far is [34], 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 k. Hence the best competitive ratio that they can get in polynomial time is $\frac{1}{e}(1 - \frac{1}{e})$(using an polynomial time $(1 - 1/e)$ approximation algorithm to find the offline solution).

On the other hand, in the streaming framework the decision to select an element is not irrevocable, and there is a buffer of limited size and you can add or remove elements from the buffer, i.e., you can deselect some elements that you have already selected. Also the random order assumption in this model is not necessary. The best known algorithm previous to this result was [6], a $1/2 - \epsilon$ approximation with a memory of size $\frac{1}{\epsilon} k \log k$. It does not need random order assumption. Also note that this algorithm can not be converted into an online algorithm. Streaming algorithms with a random order assumption have been considered for other problems. But until recently nothing better than [6] is known for this problem under random order setting.

Our algorithm is in the intersection of the above frameworks. We generalize the online model in the way that we are allowed to select more than k elements, and output a subset of size k whose value is close to OPT. Note that we are only interested in returning a subset of size at most k as our output. In algorithms like [20], although the output is close to OPT, it is not restricted to k elements, and we can't guarantee that it contains a subset of size k whose value is close to OPT. Our algorithm selects $k \log^2 k$ elements and it guarantees that there is a subset of size k whose value is $1/2 - O(k^{-1/4})$. Thus by relaxing the number of selections we improve upon the best competitive ratio $1/e$. Also in terms of computational complexity, our algorithm as in proposition 3 needs exponential time in k (polytime in n). while [34] needs to find optimal offline solution n times, each time on subsets of size n.An important aspect of our algorithm is that it is polynomial time in terms of n, but [34] is not. Both algorithms are not polynomial time in terms of k, to get a polynomial time in both n and k, one has to use a $1 - 1/e$ approximation to compute offline solutions.

Our algorithm can also be considered as a random order streaming algorithm with memory of size $k \log^2 k$. All the computation done by the algorithm to keep blocked set and marked element can be done in a memory efficient way. It is asymptotically a $1/2 - O(k^{-1/4})$ approximation algorithm. In terms of memory size, it improves upon the previous $\frac{1}{\epsilon} k \log k$, for $k > (1/\epsilon)^4$. Also in terms of running time it is polynomial in n, but it requires exponential time in k. Also note that algorithm [34] can not be converted into a streaming algorithm with nontrivial memory size. In table 3.1 and 3.2 we compare our algorithm to previously best known results in the online setting and streaming setting.

	#selections	Comp ratio	Running time	Comp ratio in poly(n)
[34]	k	$1/e - O(k^{-1/2})$	$exp(n)$	$(1 - 1/e)1/e$
this	$k \log^2 k$	$1/2 - O(k^{-1/4})$	$poly(n), exp(k)$	$1/2 - O(k^{-1/4})$

Table 3.1: Online algorithms framework

	Memory size	Approximation ratio	Running time	update time
[6]	$\frac{1}{\epsilon}k \log k$	$1/2 - \epsilon$	$poly(n, k, 1/\epsilon)$	$\frac{1}{\epsilon} \log k$
this	$k \log^2 k$	$1/2 - O(k^{-1/4})$	$poly(n), exp(k)$	$\log k$, amortize $O(1)$

Table 3.2: Random order streaming algorithms framework

Chapter 4: Cardinality Constraint: $1 - 1/e - \epsilon$ Approximation using Shortlist of Size $O(k)$

4.1 Introduction

In this chapter, we design a near-optimal algorithm for submodular k-secretary problem with shortlists. We first review the *Shortlist* model, and then precisely define the submodular k-secretary problem with shortlists. After that we describe the algorithm and the analysis of the algorithm. At the end, we prove an impossibility result.

We introduced the shortlist model in the introduction of this thesis. We explained the potential of this relaxation 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 $\log(n)$. 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 size 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 $\eta(k) = O(k)$?

In this thesis, 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. Furthermore since the upperbound given in [39] holds for random order streams, our result is asymptotically tight in this setting.

4.1.1 Problem Definition

We now give a more formal definition. Items from a set $\mathcal{U} = \{a_1, a_2, \ldots, a_n\}$ (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

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

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

The optimal offline utility is given by

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

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 $\frac{\text{ALG}}{\text{OPT}} \geq c$.

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 [44]. In this 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, this 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 problemwith shortlist of size $\eta(k)$ may potentially use a buffer of size larger than $\eta(k)$. Our algorithms, as described in the chapter, do use a large buffer, but we will show that the algorithm presented in this chapter can in fact be implemented to use only $\eta(k) = O(k)$ buffer, thus obtaining matching results for the streaming problem.

4.1.2 Our Results

Our main result is an online algorithm for 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 approximation 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 $1/e - O(k^{-1/2})$ [34] 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 queary can be done in $O(1)$ time.

Theorem 23. *For any constant $\epsilon > 0$, there exists an online algorithm (Algorithm 8) 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, $\eta_\epsilon(k) = O(2^{poly(1/\epsilon)}k)$. The running time of this online algorithm is $O(n)$.*

Specifically, we have $\eta_\epsilon(k) = c\frac{\log(1/\epsilon)}{\epsilon^2}\left(\frac{\frac{1}{\epsilon^6}\log(1/\epsilon)}{\frac{1}{\epsilon^4}\log(1/\epsilon)}\right) k$ for some constant c. The running time of our algorithm is linear in n, the size of the input, which 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[40]. 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.

Further, we show an 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 24. *For any constant $\epsilon \in (0,1)$, there exists an algorithm for the submodular random order streaming problemthat achieves an $\left(1 - \frac{1}{e} - \epsilon - O(\frac{1}{k})\right)$-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 [44], which are an approximation ratio of $\frac{1}{2}+8\times10^{-14}$ using a memory of size $O(k \log k)$.

It is natural to ask whether these k-lists 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. Note that the $1 - 1/e$ upperbound for streaming model [39] does not imply an upper bound for the shortlist model.

Theorem 25. *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 $\eta(k) = o(n)$.*

4.1.3 Comparison to related work

We compare our results (Theorem 23 and Theorem 29) 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 [34], 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 (polynomial time) algorithm achieves a competitive ratio of $1 - \frac{1}{e} - \epsilon - O(k^{-1})$. In addition to substantially improves the above-mentioned results 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 4.1 summarizes this comparison. Here, $O_\epsilon(\cdot)$ hides the dependence on the constant ϵ. The hidden constant in $O_\epsilon(.)$ is $c\frac{\log(1/\epsilon)}{\epsilon^2}\binom{\frac{1}{\epsilon^6}\log(1/\epsilon)}{\frac{1}{\epsilon^4}\log(1/\epsilon)}$ for some absolute constant c.

	#selections	Comp ratio	Running time	Comp ratio in poly(n)
[34]	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 4.1: submodular k-secretary problem settings

In the streaming setting, [11] 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. Further, 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 [6], who achieve a $1/2 - \epsilon$ approximation with a memory of size $O(\frac{1}{\epsilon}k\log k)$. There is an upper bound of $1/2 + o(1)$ on the competitive ratio achievable by any algorithm for streaming submodular maximization that only queries the value of the submodular function on feasible sets while using $o(n)$ memory [44].

[29] 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 [44] who achieve a $1/2 + 8 \times 10^{-14}$ approximation, while using a memory buffer of size $O(k \log k)$.

Table 4.2 provides a detailed comparison of our result in Theorem 29 to the above-mentioned results for submodular random order streaming problem, showing that our algorithm substantially improves the existing results on most aspects of the problem.

There is also a line of work studying the online variant of the submodular welfare maximization problem (e.g., [36, 9, 31]). In this problem, the items arrive online, and each arriving item should

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

Table 4.2: submodular random order streaming problem

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.

4.1.4 Organization

The rest of the chapter is organized as follows. Section 5.2 describes our main algorithm (Algorithm 8) for the submodular k-secretary problem with shortlists, and demonstrates that its shortlist size is bounded by $\eta_\epsilon(k) = O(k)$. In Section 5.4, we analyze the competitive ratio of this algorithm to prove Theorem 23. In Section 5.6, we provide an alternate implementation of Algorithm 8 that uses a memory buffer of size at most $\eta_\epsilon(k)$, in order to prove Theorem 29. Finally, in Section 4.5, we provide a proof of our impossibility result stated in Theorem 25. The proof of Theorem 7 along with the relevant algorithm appears in the appendix.

4.2 Algorithm description

Before giving our algorithm for submodular k-secretary problem with shortlists, we describe a simple technique for secretary problem with shortlists that achieves a $1 - \delta$ competitive ratio for with 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 the size of shortlist is less than or equal to $L = 4\ln(2/\delta)$. This algorithm is summarized as Algorithm 5. Clearly, for contant 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 6 in Section 5.4.)

Algorithm 5 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 $A^* := \max_{i \in A} a_i$

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.

Definition 7 ((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*

Algorithm 6 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 11.
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 5 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

$$\begin{aligned} a_0 &:= \max_{x \in R_{1,\ldots,w-1}} \Delta(x | S_{1,\ldots,w-1} \cup \gamma(\tau)) \\ a_\ell &:= \Delta(s_j(\ell) | S_{1,\ldots,w-1} \cup \gamma(\tau)), \forall \ell = 1, \ldots, N-1 \end{aligned}$$

where $s_j(\ell)$ denotes the ℓ^{th} item in the slot s_j.

        Let $A_j(\tau)$ be the shortlist returned by Algorithm 5 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 (5.5) and (5.6) respectively.
    $A^* \leftarrow A^* \cup (S_w \cap A)$
end for
return A, A^*.

way, that is, slot 1 *contains the first* X_1 *indices, slot* 2 *contains the next* X_2*, etc. Next, we define* k/α *windows, where window* i *consists of* $\alpha\beta$ *consecutive slots, in the same manner as we assigned slots.*

Thus, q^{th} slot is composed of indices $\{\ell, \ldots, r\}$, where $\ell = X_1 + \ldots + X_{q-1} + 1$ and $r = X_1 + \ldots + X_q$. 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 idea, 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:

$$\gamma(\tau) := \{i_1, \ldots, i_\ell\} \tag{4.1}$$

where

$$i_j := \arg \max_{i \in s_j \cup R_{1,\ldots,w-1}} f(S_{1,\ldots,w-1} \cup \{i_1, \ldots, i_{j-1}\} \cup \{i\}) - f(S_{1,\ldots,w-1} \cup \{i_1 \ldots, i_{j-1}\}). \tag{4.2}$$

In (5.3) 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. We also define R_w to be the union of all greedy subsequences of length α, and $S_w$to be the best subsequence among those. That is,

$$R_w = \cup_{\tau:|\tau|=\alpha} \gamma(\tau) \tag{4.3}$$

and

$$S_w = \gamma(\tau^*), \tag{4.4}$$

where

$$\tau^* := \arg\max_{\tau:|\tau|=\alpha} f(S_{1,\ldots,w-1} \cup \gamma(\tau)) - f(S_{1,\ldots,w-1}). \tag{4.5}$$

Note that i_j (refer to (5.3)) 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 (5.3)) 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 5) for the basic secretary problem 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 8 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 4. *Given k, n, and any constant α, β and ϵ, the size of shortlist A selected by Algorithm 8 is 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 9 in Algorithm 8 runs Algorithm 5 for $\binom{\alpha\beta}{\alpha}$ times (for all α length subsequences). By construction of Algorithm 5, for each run it will add at most $L \leq 4\log(2/\epsilon)$ items each time to the shortlist. Therefore, over all windows, Algorithm 8 adds at most $\frac{k}{\alpha} \times \alpha\beta\binom{\alpha\beta}{\alpha}L = O(k)$ items to the shortlist. □

4.3 Bounding the competitive ratio

In this section we show that for any $\epsilon \in (0,1)$, Algorithm 8 with an appropriate choice of constants α, β, achieves the competitive ratio claimed in Theorem 23 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 5, where S_w is the subset of items as defined in (5.6) for every window w. Later in Proposition 5.4, 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 shortlist returned by Algorithm 8.

Specifically, in Proposition 5, we provide settings of parameters α, β such that of $\mathbb{E}[f(\cup_w S_w)] \geq \left(1 - \frac{1}{e} - \frac{\epsilon}{2} - O(\frac{1}{k})\right)\mathrm{OPT}$. 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 at random slot in this window. (See Lemma 35.) This observation allows us to show that, in each window, 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 $\gamma(\tilde{\tau}_s) = (i_1, \ldots, i_t)$, in Lemma 40, 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:

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

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

functions) that

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

Now, since the length t of $\tilde{\tau}_w$ is close to α (as we show in Lemma 25) and since $S_w=\gamma(\tau^*)$ with τ^* defined as the "best" subsequence of length α (refer to definition of τ^* in (5.8)), we can show that a similar inequality holds for $S_w=\gamma(\tau^*)$, i.e.,

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

where $\delta'\in(0,1)$ depends on the setting of α,β. (See Lemma 45.) 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 5

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 5.4, we show that in fact the algorithm will add most of the items in $\cup_w S_w$ to the short list. 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 5 in Algorithm 8. 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 5.4). Combining this with the lower bound $\frac{\mathbb{E}[f(\cup_w S_w)]}{\mathrm{OPT}}\geq(1-\frac{1}{e}-\frac{\epsilon}{2}-O(\frac{1}{k}))$ mentioned earlier, we complete the proof of competitive ratio bound stated in Theorem 23.

4.3.1 Preliminaries

The following properties of submodular functions are well known (e.g., see [10, 15, 19]).

Lemma 12. *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 13. *Denote by $A(p)$ a random subset of A where each element has probability atleast 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 14 (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 15 (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 $\delta \in (0, 1)$.

4.3.2 Some useful properties of (α, β) windows

We first prove some useful properties of (α, β) windows, defined in Definition 11 and used in Algorithm 8. The first observation is that every item will appear uniformly at random in one of the $k\beta$ slots in (α, β) windows.

Definition 8. *For each item $e \in I$, 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 16. *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. It folllows from the following lemma.

Lemma 17. *For each $y \in [k\beta]^n$, $Pr\{Y = y\} = (\frac{1}{k\beta})^n$.*

Proof. Consider pair (π, ψ), where $\pi : I \to [n]$ defines the random order on I. Throw n balls uniformly into $k\beta$ bins. Let ψ_j be the bin that j-th ball goes into. Note that ψ and π are independent. Now consider

$$(\frac{1}{k\beta}s_1 + \cdots + \frac{1}{k\beta}s_{k\beta})^n = \left(\frac{1}{k\beta}\right)^n \sum_{t_1,\cdots,t_{k\beta}} Q_{t_1,\cdots,t_{k\beta}} {s_1}^{t_1} \cdots {s_{k\beta}}^{t_{k\beta}} .$$

For a given $y \in [k\beta]^I$, suppose t_i is the number of elements in slot s_i. Then from above expansion the probability that ψ divides input into slots of size $t_1, \cdots, t_{k\beta}$ is

$$\left(\frac{1}{k\beta}\right)^n Q_{t_1,\cdots,t_{k\beta}} = \left(\frac{1}{k\beta}\right)^n \binom{n}{t_1, t_2, \cdots, t_{k\beta}} .$$

Now for such a ψ, the probability that permutations π satisfy $Y = y$ is

$$\frac{t_1! \cdots t_{k\beta}!}{n!} .$$

Thus the probability that $Y = y$ is

$$\left(\frac{1}{k\beta}\right)^n \binom{n}{t_1, t_2, \cdots, t_{k\beta}} \frac{t_1! \cdots t_{k\beta}!}{n!} = \left(\frac{1}{k\beta}\right)^n .$$

□

Next, we make important observations about the probability of assignment of items in S^* in the slots in a window w, given the sets $R_{1,\ldots,w-1}, S_{1,\ldots,w-1}$ (refer to (5.5), (5.6) 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 9. *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* (5.2)*. 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 18. *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.*

Proof. Given the assignment of items to each slot, it follows from the definition of $\gamma(\tau)$ and S_w (refer to (5.2) and (5.6)) that $T_{1,\ldots,w}$ and $S_{1,\ldots,w}$ are independent of the ordering of items within a slot. Now, since the assignment of items to slot are determined by the configuration Y, we obtain the desired lemma statement. □

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

$$p_{is} := \mathbb{P}(i \in s \cup Supp(T) | T_{1,\ldots,w-1} = T). \tag{4.6}$$

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

$$p_{is'} = p_{is''} \geq \frac{1}{k\beta}\,. \tag{4.7}$$

Proof. If $i \in Supp(T)$ then the statement of the lemma is trivial, so consider $i \notin Supp(T)$. For such i, $p_{is} = \mathbb{P}(Y_i = s | T_{1,\ldots,w-1} = T)$.

We show that for any pair of slots s, s', where s is a slot in first $w-1$ windows and s' is a slot in window w,

$$\mathbb{P}(T_{1,\ldots,w-1} = T | Y_i = s) \leq \mathbb{P}(T_{1,\ldots,w-1} = T | Y_i = s') \,. \tag{4.8}$$

And, for any pair of slots s', s'' in windows $\{w, w+1, \cdots, W\}$,

$$\mathbb{P}(T_{1,\ldots,w-1} = T | Y_i = s') = \mathbb{P}(T_{1,\ldots,w-1} = T | Y_i = s''). \tag{4.9}$$

To see (4.8), suppose for a configuration Y we have $Y_i = s$ and $T_{1,\cdots,w-1}(Y) = T$. Since $i \notin Supp(T)$, then by definition of $T_{1,\ldots,w-1}$ we have that $i \notin \gamma(\tau)$ for any α length subsequence τ of slots in any of the windows $1, \ldots, w-1$. Therefore, if we remove i from windows $1, \cdots, w-1$ (i.e., consider another configuration where Y_i is in windows $\{w, \ldots, W\}$) then $T_{1,\cdots,w-1}$ would not change. This is because i is not the output of argmax in definition of $\gamma(\tau)$ (refer to (5.2)) for any τ, so that its removal will not change the output of argmax. Also by adding i to slot s', $T_{1,\cdots,w-1}$ will not change since s' is not in window $1, \cdots, w-1$. Suppose configuration Y' is a new configuration obtained from Y by changing Y_i from s to s'. Therefore $T_{1,\cdots,w-1}(Y') = T$. Also remember that from lemma 17, $\mathbb{P}(Y) = \mathbb{P}(Y')$. This mapping shows that $\mathbb{P}(T_{1,\ldots,w-1} = T | Y_i = s) \leq \mathbb{P}(T_{1,\ldots,w-1} = T | Y_i = s')$.

The proof for (4.9) is similar.

By applying Bayes' rule to (4.8) we have

$$\mathbb{P}(Y_i = s | T_{1,\ldots,w-1} = T) \frac{\mathbb{P}(T_{1,\ldots,w-1} = T)}{\mathbb{P}(Y_i = s)} \leq \mathbb{P}(Y_i = s' | T_{1,\ldots,w-1} = T) \frac{\mathbb{P}(T_{1,\ldots,w-1} = T)}{\mathbb{P}(Y_i = s')} \,.$$

Also from Lemma 33, $\mathbb{P}(Y_i = s) = \mathbb{P}(Y_i = s')$ thus

$$\mathbb{P}(Y_i = s | T_{1,\ldots,w-1} = T) \leq \mathbb{P}(Y_i = s' | T_{1,\ldots,w-1} = T) \,.$$

Now, for any pair of slots s', s'' in windows $w, w+1, \cdots, W$, by applying Bayes' rule to the equation (4.9), we have $p_{is'} = \mathbb{P}(Y_i = s' | T_{1,\ldots,w-1} = T) = \mathbb{P}(Y_i = s'' | T_{1,\ldots,w-1} = T) = p_{is''}$. That is, i

has as much probability to appear in s' or s'' as any of the other (at most $k\beta$) slots in windows $w, w+1, \ldots, W$. As a result $p_{is''} = p_{is'} \geq \frac{1}{k\beta}$.

□

Lemma 20. *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.*

Proof. To prove this, we show that $\mathbb{P}(Y_i = s | T_{1,\cdots,w-1} = T) = \mathbb{P}(Y_i = s | T_{1,\cdots,w-1} = T \text{ and } Y_j = s')$. Suppose Y' is a configuration such that $Y'_i = s$ and $Y'_j = s'$, and $T_{1,\cdots,w-1}(Y') = T$. Assume there exists another feasible slot assignment of j, i.e., there is another configuration Y'' such that $T_{1,\cdots,w-1}(Y'') = T$ and $Y''_j = s''$ where $s'' \neq s'$. (If no such configuration Y'' exists, then $\mathbf{1}(Y_j = s')|T$ is always 1, and the desired lemma statement is trivially true.) Then, we prove the desired independence by showing that there exists a feasible configuration where slot assignment of i is s, and j is s''. This is obtained by changing Y_j from s' to s'' in Y', to obtain another configuration $\bar{Y}$. In Lemma 37, we show that this change will not effect $T_{1,\cdots,w-1}$, i.e., $T_{1,\cdots,w-1}(\bar{Y}) = T$. Thus configuration $\bar{Y}$ satisfies the desired statement. □

Lemma 21. *Fix a slot s', T, and $j \notin Supp(T)$. Suppose that there exists some configuration Y' such that $T_{1,\cdots,w-1}(Y') = T$ and $Y'_j = s'$. Then, given any configuration Y'' with $T_{1,\ldots,w-1}(Y'') = T$, we can replace Y''_j with s' to obtain a new configuration $\bar{Y}$ that also satisfies $T_{1,\ldots,w-1}(\bar{Y}) = T$.*

Proof. Suppose the slot s' lies in window w'. If $w' \geq w$ then the statement is trivial. So suppose $w' < w$. Create an intermediate configuration by *removing* the item j from Y'', call it Y^-. Since $j \notin Supp(T_{1,\cdots,w-1}(Y'')) = Supp(T)$ we have $T_{1,\cdots,w-1}(Y^-) = T$. In fact, for every subsequence τ, the greedy subsequence for Y'', will be same as that for Y^-, i.e., $\gamma_{Y''}(\tau) = \gamma_{Y^-}(\tau)$. Now add item j to slot s' in Y^-, to obtain configuration $\bar{Y}$. We claim $T_{1,\cdots,w-1}(\bar{Y}) = T$.

By construction of $T_{1,\ldots,w}$, we only need to show that j will not be part of the greedy subsequence $\gamma_{\bar{Y}}(\tau)$ for any subsequence $\tau, |\tau| = \alpha$ containing the slot s' when the input is in configuration $\bar{Y}$. To prove by contradiction, suppose that j is part of greedy subsequence for some

τ ending in the slot s'. For this τ, let $\gamma_{Y^-}(\tau) := \{i_1, \cdots, i_{\alpha-1}, i_\alpha\} = \gamma_{Y''}(\tau)$. Note that since the items in the slots before s' are identical for $\bar{Y}$ and Y^-, we must have that $\gamma_{\bar{Y}}(\tau) = \{i_1, \cdots, i_{\alpha-1}, j\}$, i.e., $\Delta_f(j|S_{1,\ldots,w'-1} \cup \{i_1, \ldots, i_{\alpha-1}\}) \geq \Delta_f(i_\alpha|S_{1,\ldots,w'-1} \cup \{i_1, \ldots, i_{\alpha-1}\})$. On the other hand, since $T_{1,\cdots,w'-1}(Y') = T_{1,\cdots,w'-1}(Y'') = T$(restricted to $w'-1$ windows), we have that $\gamma_{Y'}(\tau) = \{i_1, \cdots, i_\alpha\}$. However, $Y'_j = s'$. Therefore j was not part of the greedy subsequence $\gamma_{Y'}(\tau)$ even though it was in the last slot in τ, implying $\Delta_f(j|S_{1,\ldots,w'-1} \cup \{i_1, \ldots, i_{t-1}\}) < \Delta_f(i_t|S_{1,\ldots,w'-1} \cup \{i_1, \ldots, i_{t-1}\})$. This contradicts the earlier observation.

□

4.3.3 Bounding $\mathbb{E}[f(\cup_w S_w)]$/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 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 10 (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 $i \in S^* \cap T$, 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 22. *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 of 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 is independently assigned to a Z_s in construction. Further, every $\in S^* \cap T$ is assigned with equal probability to every slot in s.

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 (5.10). 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 36, 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 23. *Given the sequence $\tilde{\tau}_w = (s_1, \ldots, s_t)$ defined in Definition 15, let $\gamma(\tilde{\tau}_s) = (i_1, \ldots, i_t)$, with $\gamma(\cdot)$ as defined in* (5.2)*. Then, for all $j = 1, \ldots, t$,*

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

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

Now, using Lemma 38, 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 38, 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 $i, i' \in S^* \backslash \{Z_{s_1} \cup \ldots \cup Z_{s_{j-1}}\}$,

$$\Pr(i \in Z_{s_j}|Z_{s_1} \cup \ldots \cup Z_{s_{j-1}}) = \Pr(i' \in Z_{s_j}|Z_{s_1} \cup \ldots \cup Z_{s_{j-1}}) \geq \frac{1}{k} . \tag{4.10}$$

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 (5.2)), 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 $i \in s_j \cup Supp(T_{1,\ldots,w-1}) \supseteq Z_{s_j}$. Therefore, we can lower bound the increment provided by i_j by that provided by a randomly picked item from Z_{s_j}.

$$
\begin{aligned}
& \mathbb{E}[\Delta_f(i_j|S_{1,\ldots,w-1} \cup \{i_1,\ldots,i_{j-1}\}|T_{1,\ldots,w-1} = T, i_1,\ldots,i_{j-1}] \\
\text{(using (5.12))} \quad \geq \quad & \frac{1}{k}\mathbb{E}[\sum_{i \in S^* \backslash \{Z_1,\ldots Z_{s_{j-1}}\}} \mathbb{E}[\Delta_f(i|S_{1,\ldots,w-1} \cup \{i_1,\ldots,i_{j-1}\}|T, i_1,\ldots,i_{j-1}]] \\
\text{(using Lemma 12, monotonicity of } f) \quad \geq \quad & \frac{1}{k}\mathbb{E}[\left(f(S^* \backslash \{Z_1,\ldots Z_{s_{j-1}}\}) - f(S_{1,\ldots,w-1} \cup i_{1,\ldots,s-1})\right)|T] \\
\text{(using monotonicity of } f) \quad \geq \quad & \frac{1}{k}\mathbb{E}[\left(f(S^* \backslash \cup_{s' \in w} Z_{s'}) - f(S_{1,\ldots,w-1} \cup i_{1,\ldots,s-1})\right)|T] \\
\text{(using Lemma 38 and Lemma 13)} \quad \geq \quad & \frac{1}{k}\left(\left(1 - \frac{\alpha}{k}\right) f(S^*) - f(S_{1,\ldots,w-1} \cup i_{1,\ldots,s-1})\right)
\end{aligned}
$$

The last inequality uses the observation from Lemma 38 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 13 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 24.

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

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

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

By martingale stopping theorem, this implies:

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

where stopping time $t = |\tilde{\tau}_w|$. ($t = |\tilde{\tau}_w| \leq \alpha\beta$ is bounded, therefore, 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 (5.6) and (5.8)). To compare it with $\tilde{\tau}_w$, we need a bound on size of $\tilde{\tau}_w$.

Lemma 25. *For any real $\delta \in (0, 1)$, and if $k \geq \alpha\beta$, $\alpha \geq 8\log(\beta)$ and $\beta \geq 8$, then given any $T_{1,\ldots,w-1} = T$,*

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

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

Proof. By definition,

$$|\tilde{\tau}_w| = |s \in w : Z_s \neq \phi| \ .$$

Again, we use $s' \prec_w s$ to denote all slots before s in window w. Then, from Lemma 38, given $T_{1,\ldots,w-1} = T$, for all $i \cap S^*$ and slot s in window w, $\Pr[i \in Z_s | Z_{s'}, s' \prec_w s, T]$ is either 0 or $1/(k\beta)$.

Therefore,

$$\Pr[Z_s \neq \phi | T, Z_{s'}, s' \prec_w s] \leq \sum_{i \in S^*} \frac{1}{k\beta} = \frac{1}{\beta} .$$

Therefore $X_s = |s' \preceq_w s : Z_{s'} \neq \phi| - \frac{s}{\beta}$ is a super-martingale, with $X_s - X_{s-1} \leq 1$. Since there are $\alpha\beta$ slots in window w, $X_{\alpha\beta} = |s \in w : Z_s \neq \phi| - \alpha$. Applying Azuma-Hoeffding inequality to $X_{\alpha\beta}$ (refer to Lemma 14) we get that

$$\Pr\left(|s \in w : Z_s \neq \phi| \geq (1+\delta)\alpha | T\right) \leq \exp\left(-\frac{\delta^2\alpha}{2\beta}\right) \tag{4.11}$$

which proves the desired upper bound.

For lower bound, first observe that every $i \in S^*$ appears in $\cup_{s \in w} Z_s$ independently with probability $\frac{\alpha}{k}$. Using Chernoff bound for Bernoulli random variables (Lemma 15), for any $\delta \in (0,1)$

$$\Pr(|| \cup_{s \in w} Z_s| - \alpha| > \delta\alpha) \leq \exp(-\delta^2\alpha/3) . \tag{4.12}$$

Also, from independence of $i \in Z_s|T$ and $i' \in Z_s|T$ for any $i, i' \in S^*, i \neq i'$ (refer to Lemma 38),

$$\Pr(i, i' \in Z_s | T, i, i' \notin Z_{s'} \text{ for any } s' \prec_w s) \leq \frac{1}{k^2\beta^2}$$

for any $s \in w$; so that

$$\Pr\left(|Z_s| = 1 | T, Z_{s'}, s' \prec_w s\right) \geq \frac{k - |Z_{s'} : s' \prec_w s|}{k\beta} - \frac{1}{\beta^2} \geq \left(1 - \frac{2\alpha}{k}\right)\frac{1}{\beta} - \frac{1}{\beta^2} - e^{-\frac{\alpha}{4}} =: p . \tag{4.13}$$

where in the last inequality we substituted the upper bound on $|Z_{s'} : s' \prec_w s|$ from (4.12). Specifically, using (4.12) with $\delta = 3/4$, we obtained that $|Z_{s'} : s' \prec_w s| \leq (1 + \frac{3}{4})\alpha \leq 2\alpha$ with probability $\exp(-\alpha/4)$. Also if $\alpha \geq 8\log(\beta)$, and $k \geq \alpha\beta$, we have $p := \left(1 - \frac{2\alpha}{k} - \frac{1}{\beta}\right)\frac{1}{\beta} - e^{-\frac{\alpha}{4}} \geq (1 - \frac{4}{\beta})\frac{1}{\beta}$.

Now, applying Azuma-Hoeffding inequality (Lemma 14), the total number of slots (out of $\alpha\beta$

slots) for which $|Z_s| = 1$ can be lower bounded by:

$$\Pr\left(|\{s \in w : |Z_s| = 1\}| \leq (1-\delta)p\alpha\beta|T\right) \leq \exp\left(-\frac{\delta^2 p^2 \alpha\beta}{2}\right) . \tag{4.14}$$

Substituting $p \geq (1 - \frac{4}{\beta})\frac{1}{\beta}$,

$$\Pr\left(|\{s \in w : |Z_s| = 1\}| \leq (1-\delta)(1-\frac{4}{\beta})\alpha|T\right) \leq \exp\left(-\frac{\delta^2(1-4/\beta)^2\alpha}{2\beta}\right) .$$

We further substitute $\beta \geq 8$ in the right hand side of the above inequality, to bound the probability by $\exp(-\delta^2\alpha/8\beta)$.

□

Lemma 26 (Corollary of Lemma 25)**.** *For any real $\delta' \in (0,1)$, if parameters k, α, β satisfy $k \geq \alpha\beta$, $\beta \geq \frac{8}{(\delta')^2}$, $\alpha \geq 8\beta^2 \log(1/\delta')$, then given any $T_{1,\ldots,w-1} = T$, with probability at least $1 - \delta' e^{-\alpha/k}$,*

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

Proof. We use the previous lemma with $\delta = \delta'/2$ to get lower bound of $(1-\delta')\alpha$ with probability $1 - \exp(-(\delta')^2\alpha/32\beta)$. Then, substituting $k \geq \alpha\beta \geq \frac{64\beta}{(\delta')^2}\log(1/\delta')$ so that using $\beta \leq \frac{k(\delta')^2}{64\log(1/\delta')}$ we can bound the violation probability by

$$\exp(-(\delta')^2\alpha/32\beta) \leq \exp(-(\delta')^2\alpha/64\beta)\exp(-\alpha/k) \leq \delta' e^{-\alpha/k}.$$

where the last inequality uses $\alpha \geq 8\beta^2\log(1/\delta')$ and $\beta \geq 8/(\delta')^2$. □

Lemma 27. *For any real $\delta' \in (0,1)$, if parameters k, α, β satisfy $k \geq \alpha\beta$, $\beta \geq \frac{8}{(\delta')^2}$, $\alpha \geq 8\beta^2\log(1/\delta')$, then*

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

Proof. The lemma follows from substituting Lemma 44 in Lemma 43. □

Now, we can deduce the following proposition.

Proposition 5. *For any real $\delta' \in (0,1)$, if parameters k, α, β satisfy $k \geq \alpha\beta$, $\beta \geq \frac{8}{(\delta')^2}$, $\alpha \geq 8\beta^2 \log(1/\delta')$, then the set $S_{1,\ldots,W}$ tracked by Algorithm 8 satisfies*

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

Proof. By multiplying the inequality Lemma 45 from $w = 1, \ldots, W$, where $W = k/\alpha$, we get

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

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

4.3.4 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 8. The main difference between the two sets is that in construction of shortlist A, Algorithm 5 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 5 to bound the probability of this event.

Proposition 6. *For any $\delta \in (0,1)$, and input $I = (a_1, \ldots, a_N)$, Algorithm 5 returns $A^* = \max(a_1, \ldots, a_N)$ with probability $(1-\delta)$.*

The proof of the above proposition appeared in the m-submodular chapter. 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 28. *Let A be the shortlist returned by Algorithm 8, and δ is the parameter used to call Algorithm 5 in Algorithm 8. Then, for given configuration Y, for any item a, we have*

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

Proof. From Lemma 34 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 34). Therefore, from Proposition 6, with probability $1 - \delta$, a will be added to the shortlist $A_{j'}(\tau')$ by Algorithm 5. Thus $a \in A \supseteq A_{j'}(\tau')$ with probability at least $1 - \delta$.

□

Proposition 7.

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

where $A^* := S_{1,\cdots,W} \cap A$ is the size k subset of shortlist A returned by Algorithm 8.

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 8. Therefore using Lemma 13, 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 23. Now, we can show that Algorithm 8 provides the results claimed in Theorem 23 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 5 and Proposition 5.4, for $k \geq \alpha\beta$ we obtain:

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

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 8.

4.4 Streaming

In this section, we show that Algorithm 8 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 29 (restated below) as a corollary of Theorem 23.

Theorem 24. *For any constant $\epsilon \in (0, 1)$, there exists an algorithm for the submodular random order streaming problemthat achieves an $\left(1 - \frac{1}{e} - \epsilon - O(\frac{1}{k})\right)$-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 $O(1 + \frac{k^2}{n})$.*

In the current description of Algorithm 8, 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 8, 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 (5.5)). And, $S_w = \gamma(\tau^*)$ for the best subsequence τ^* among these subsequences (refer to (5.6)). 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 8 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 τ 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 $2\binom{\alpha\beta}{\alpha} = O(1)$. However, 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 $2|R_{1,\ldots,w-1}| \leq 2k\binom{\alpha\beta}{\alpha}$ 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 29.

4.5 Impossibility Result (Proof of Theorem 25)

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

Theorem 25. *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 subset S of size at most $k+1$, we have $f(S) \leq k + \frac{k}{2} + \frac{k}{2} = 2k$.

Lemma 29. *f is a monotone submodular function.*

Proof. We have to show that for any item x and subsets $S \subseteq T$, $\Delta_f(x|S) \geq \Delta_f(x|T)$. We consider the following cases:

- if $|T \cap C| \geq 2 \implies \Delta_f(x|T) = 0$, so it is trivial.
- if $|T \cap C| = 0 \implies |S \cap C| = 0 \implies \Delta_f(x|S) \geq \Delta_f(x|T)$ because of submodularity of g.
- if $|T \cap C| = 1 \implies |S \cap C| \leq 1$
  - if $|S \cap C| = 1$ then $S \cap C = T \cap C = \{c^\ell\}$ for some ℓ:
    - $x \in \{d_1^\ell, \cdots, d_k^\ell\} \implies \Delta_f(x|S) = 1/2^{i+1} + 1/2^{i+1}$ for $i = \lfloor (|S|-1)/k \rfloor$, and $\Delta_f(x|T) = 1/2^j + 1/2^j$ for some $j = \lfloor (|T|-1)/k \rfloor$ and $j \geq i+1$.
    - $x \notin \{d_1^\ell, \cdots, d_k^\ell\} \implies \Delta_f(x|S) = 1/2^{i+1}$ for $i = \lfloor (|S|-1)/k \rfloor$ and $\Delta_f(x|T) = 1/2^j$ for some $j \geq i+1$.
  - if $|S \cap C| = 0 \implies \Delta_f(x|T) \leq 1/2^{j+1} + 1/2^{j+1}$ for $j = \lfloor (|T|-1)/k \rfloor$ and $\Delta_f(x|S) = 1/2^i$ for some $i \leq j$.

Thus $\Delta_f(x|S) \geq \Delta_f(x|T)$.

Monotonicity follows trivially from the definition of f. □

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

Proof. Suppose $(e_1, \cdots, e_n)$ indicates the ordered input according to random ordering π on $\bar{I}$. Now let t be the random variable indicating the index of c^ℓ in $(e_1, \cdots, e_n)$, i.e., $e_t = c^\ell$. Then, due to random ordering, and random choice of $\bar{I}$ from $I_1, \ldots, I_\ell$, we have

$$\mathbb{E}_{(\bar{I},\pi)}[|Alg \cap \{e_1, \cdots, e_{t-1}\} \cap D^1|] = \cdots = \mathbb{E}[|Alg \cap \{e_1, \cdots, e_{t-1}\} \cap D^L|] \,.$$

Also, since $D^\ell, \ell = 1, \ldots, L$ are disjoint,

$$\sum_{\ell=1}^{L} \mathbb{E}[|Alg \cap \{e_1, \cdots, e_{t-1}\} \cap D^\ell|] \leq \eta(k+1) \,.$$

Since $\bar{D} = D^\ell$ with probability $1/L$, we have

$$H := \mathbb{E}[|Alg \cap \{e_1, \cdots, e_{t-1}\} \cap \bar{D}|] = \frac{1}{L} \sum_{\ell=1}^{L} \mathbb{E}[|Alg \cap \{e_1, \cdots, e_{t-1}\} \cap D^\ell|] \leq \frac{1}{L}\eta(k+1) \,.$$

Now define $G := \mathbb{E}[|Alg \cap \{e_t, \cdots, e_n\} \cap \bar{D}|]$. We have

$$G \leq \mathbb{E}[|\bar{D} \cap \{e_t, \cdots, e_n\}|] \leq k/2 \ .$$

Thus

$$\mathbb{E}[|Alg \cap \bar{D}|] \leq G + H \leq k/2 + \eta(k+1)/L \ .$$

□

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

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

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

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

therefore

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

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 competitive ratio upper bound of $\frac{7}{8} + o(1)$ when $\eta(k+1) = o(n)$, to complete the proof of Theorem 25.

Chapter 5: Matroid constraints

5.1 Introduction

In recent years, submodular optimization has found applications for different machine learning and data mining applications including data summarization, sparsity, active learning, recommendation, high-order graphical model inference, determinantal point processes [17, 6, 32], network inference, network design [25, 23], and influence maximization in social networks [32].

In data summarization tasks, the collection of elements is generated continuously, and keeping a real time summary of the the data seen so far is important. Thus, a series of recent papers studied streaming algorithms for maximizing a submodular function. The first work to consider a one-pass streaming algorithm were the work of Badanidiyuru et al. [6], who described a $(1/2 - \epsilon)$-approximation streaming algorithm for maximizing a monotone submodular function subject to a k-cardinality constraint, with a memory of size $O(\frac{1}{\epsilon}k \log k)$. Recently, Kazermi et al. [32] proposed a new algorithm with the same approximation ratio but with improved memory $O(k)$.

[44] give an upper bound of $1/2 + o(1)$ on the approximation ratio achievable by any algorithm for streaming submodular maximization that only queries the value of the submodular function on feasible sets (sets of size at most k) while using $o(n)$ memory [44]. Consequently, they consider the random order streaming model to go beyond this worst case analysis for the adversarial order inputs. They achieve a $1/2 + 8 \times 10^{-14}$ approximation for maximizing a monotone submodular function in the random order model, using a memory buffer of size $O(k \log k)$. Subsequently, [2] substantially improve their result to $1 - 1/e - \epsilon - O(1/k)$ approximation which is close to the best possible guarantee in the offline setting, i.e., $1 - 1/e$ (assuming $P \neq NP$). Furthermore, they improve the required memory buffer to only $O(k)$.

In addition to the simple cardinality constraint, more general constraints have been studied

in the literature. Chakrabarti and Kale [11] give a $1/4p$ approximation algorithm for streaming monotone submodular functions maximization subject to the intersection of p matroid constraints. Chekuri et al. [12] extend it to p-matchoids constraints. p-matchoids generalize many basic combinatorial constraints such as the cardinality constraint, the intersection of p matroids, and matchings in graphs and hyper-graphs. A formal definition of a p-matchoid is given in Section 5.5. Recently, [17] designed a more efficient algorithm with lower number of function evaluations achieving the same approximation $1/4p$.

The algorithms of [17], [11] and [12] for monotone submodular objective functions require only $O(k)$ memory (k is the size of the largest feasible set) and using only $O(kq)$ value and independence oracle queries for processing a each element of the stream (q is a the number of matroids used to define the p-matchoid constraint).

Furthermore, the greedy algorithm yields a ratio of $1/(p+1)$ for p-independent systems [43]. These ratios for greedy are tight for all p even when the p-independent system is obtained as an intersection of p matroids. For large but fixed p, the p-dimensional matching problem is NP-hard to approximate to within an $\Omega(\log p/p)$ factor [28].

The submodular matroid secretary problem with shortlists. Motivated by the improvements of the competitive ratio for the submodular k-secretary problem in the *shortlist* model, we ask if a similar improvement can be achieved by relaxing the submodular matroid secretary problem to allow a shortlist. That is, instead of choosing an independent set of a matroid $\mathcal{M}$ with $rk(\mathcal{M}) = k$, 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 an independent subset of items from the $\eta(k)$ selected items. Then what is the best competetive ratio that we can achieve in this model for example when $\eta(k) = O(k)$? Is it possible to improve the best known competetive ratio for matroid secretary problem in this model?

Problem definition. We now give a more formal definition. We are given matroid $\mathcal{M} = (\mathcal{N}, \mathcal{I})$. Items from a set $\mathcal{U} = \{a_1, a_2, \ldots, a_n\}$ (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

$$\text{ALG} = \mathbb{E}[\max_{S \subseteq A, S \in \mathcal{I}} f(S)],$$

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

$$\text{OPT} := f(S^*), \text{ where } S^* = \arg\max_{S \subseteq [n], S \in \mathcal{I}} f(S).$$

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 $\frac{\text{ALG}}{\text{OPT}} \geq c$.

Given the shortlist A, since the problem of computing the solution $\arg\max_{S \subseteq A, S \in \mathcal{I}} f(S)$ can itself be computationally intensive, our algorithm will also track and output a subset $A^* \subseteq A, |A^*| \leq k$.

Our results. We design an algorithm that achieves a $\frac{1}{2}(1 - 1/e^2 - \epsilon - O(1/k))$ competitive ratio for any constant $\epsilon > 0$, using a shortlist of size $O(k)$ for the matroid secretary problem with shortlists. This is especially surprising considering that the best known competitive ratio for the matroid secretary problem is $O(\log \log k)$. We are also able to get a constant competitive algorithm using shortlist of size at most k and also a constant competitive algorithm in the preemption model.

Theorem 26. *For any constant $\epsilon > 0$, there exists an online algorithm (Algorithm 8) for the submodular matroid secretary problem with shortlists that achieves a competitive ratio of $\frac{1}{2}(1 - \frac{1}{e^2} - \epsilon - O(\frac{1}{k}))$, with shortlist of size $\eta_\epsilon(k) = O(k)$. Here, $\eta_\epsilon(k) = O(2^{poly(1/\epsilon)}k)$. The running time of this online algorithm is $O(nk)$.*

Theorem 27. *For the matroid secretary problem in preemption model, and matroid secretary problem that uses shortlist of size at most $\eta(k) = k$, there is an algorithm that achieves competitive ratio*

$\frac{1}{2}(1-1/e)(1-1/e^2-\epsilon)$.

Furthermore, for a more general constraint, namely p-matchoid constraints (defined in section 5.5) we prove:

Theorem 28. *For any constant $\epsilon > 0$, there exists an online algorithm for the submodular secretary problem with p-matchoid constraints that achieves a competitive ratio of $\frac{1}{p+1}(1-\frac{1}{e^{p+1}}-\epsilon-O(\frac{1}{k}))$, with shortlist of size $\eta_\epsilon(k) = O(k)$. Here, $\eta_\epsilon(k) = O(2^{poly(1/\epsilon)}k)$. The running time of this online algorithm is $O(n\kappa^p)$, where $\kappa = \max_{i\in[q]} rk(\mathcal{M}_i)$.*

The proposed algorithm also has implications for another important problem of submodular function maximization under random order streaming model and matchoid constraints. We show that our algorithm can be implemented in the streaming setting using $O(k)$ memory. It achieves $\frac{1}{p+1}(1-1/e^{p+1}-\epsilon-O(1/k))$ approximation.

Theorem 29. *For any constant $\epsilon \in (0,1)$, there exists an algorithm for the submodular random order streaming problem with matroid constraints that achieves $\frac{1}{2}(1-\frac{1}{e}-\epsilon-O(\frac{1}{k}))$ 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 $O(pk+\frac{k^2}{n})$.*

Theorem 30. *For any constant $\epsilon > 0$, there exists an algorithm for the submodular random order streaming problem with p-matchoid constraints that achieves $\frac{1}{p+1}(1-\frac{1}{e^{p+1}}-\epsilon-O(\frac{1}{k}))$ 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 $O(p\kappa+\kappa^p+\frac{k^2}{n})$, where $\kappa = \max_{i\in[q]} rk(\mathcal{M}_i)$.*

5.1.1 Related Work

In this section, we overview some of the related online problems. In the *submodular k-secretary problem* introduced by [7] and [27], the algorithm selects k items, but the value of the selected items is given by a monotone submodular function The algorithm can select at most

k items $S = \{a_1 \cdots, a_k\}$, from a randomly ordered sequence of n items. The goal is to maximize $f(S)$. Currently, the best result for this setting is due to [34], who achieve a $1/e$-competitive ratio in exponential time in k, or $\frac{1}{e}(1 - \frac{1}{e})$ in polynomial time in n and k. Submodular functions also has been used in the network design problems [25, 24]. There are also some related online coloring problems in the literature [26, 1].

In the *matroid secretary problem*, the elements of a matroid $\mathcal{M}$ arrive in random order. Once we observe an item we need to irrevocably decide whether or not to accept it. The set of selected elements should form an independent set of the matroid. The goal is to maximize the total sum of the values assigned to these elements. It has applications in online mechanism design, in particular when the set of acceptable agents form a matroid [5].

The existence of a constant competitive algorithm is a long-standing open problem. [37] provides the first $O(\log\log(k))$- competitive algorithm (the hidden constant is $2^{2^{34}}$). [18] give a simpler order-oblivious $2560(\log\log 4k + 5)$-competitive algorithm for the matroid secretary problem, by knowing only the cardinality of the matroid in advance. There are some $O(1)$-competitive algorithms for special variants of the matroid secretary problem. For example, the elements of the ground set are assigned to a set of weights uniformly at random hen a 5.7187-competitive algorithm is possible for any matroid [47]. Furthermore, a $16(1 - 1/e)$-competitive algorithm can be achieved as long as the weight assignment is done at random, even if we assume the adversarial arrival order.

Recently, [10] considered a different relaxation which is called preemptions model. In this model, elements added to S can be discarded later. The main result of [10], is a randomized 0.0893-competitive algorithm for cardinality constraints using $O(k)$ memory.

5.1.2 Related Work

In this section, we overview some of the related online problems. In the *submodular k-secretary problem* introduced by [7] and [27], the algorithm selects k items, but the value of the selected items is given by a monotone submodular function The algorithm can select at most

k items $S = \{a_1 \cdots, a_k\}$, from a randomly ordered sequence of n items. The goal is to maximize $f(S)$. Currently, the best result for this setting is due to [34], who achieve a $1/e$-competitive ratio in exponential time in k, or $\frac{1}{e}(1 - \frac{1}{e})$ in polynomial time in n and k.

In the *matroid secretary problem*, the elements of a matroid $\mathcal{M}$ arrive in random order. Once we observe an item we need to irrevocably decide whether or not to accept it. The set of selected elements should form an independent set of the matroid. The goal is to maximize the total sum of the values assigned to these elements. It has applications in online mechanism design, in particular when the set of acceptable agents form a matroid [5].

The existence of a constant competitive algorithm is a long-standing open problem. Lachish [37] provides the first $O(\log\log(k))$- competitive algorithm (the hidden constant is $2^{2^{34}}$). Feldman et al. [18] give a simpler order-oblivious $2560(\log\log 4k + 5)$-competitive algorithm for the matroid secretary problem, by knowing only the cardinality of the matroid in advance. There are some $O(1)$-competitive algorithms for special variants of the matroid secretary problem. For example, the elements of the ground set are assigned to a set of weights uniformly at random then a 5.7187-competitive algorithm is possible for any matroid [47]. Furthermore, a $16(1 - 1/e)$-competitive algorithm can be achieved as long as the weight assignment is done at random, even if we assume the adversarial arrival order.

Recently, [10] considered a different relaxation which is called preemptions model. In this model, elements added to S can be discarded later. The main result of [10], is a randomized 0.0893-competitive algorithm for cardinality constraints using $O(k)$ memory.

5.2 Algorithm description

Before describing our main algorithm we design a subroutine for a problem that we call it *secretary problem with replacement*: we are given a matroid $\mathcal{M} = (\mathcal{E}, \mathcal{I})$ and an independent set $S \in \mathcal{I}$. A pool of items $I = (a_1, \cdots, a_N)$ arriving sequentially in a uniformly random order, find an element e from I that can be added to S after removing possibly one element e' from S such that the set remains independent, i.e., $S + e - e' \in \mathcal{I}$. The goal is to choose element e and e' in an online

manner with maximum marginal increment $g(e, S) = f(S + e - e') - f(S)$. More precisely define function g as:

$$g(e, S) := f(S + e - \theta(e, S)) - f(S), \tag{5.1}$$

where θ is defined as:

$$\theta(e, S) := \arg\max_{e' \in S} \{f(S + e - e') | S + e - e' \in \mathcal{I}\}$$

We will consider the variant in which we are allowed to have a shortlist, where the algorithm can add items to a shortlist and choose one item from the shortlist at the end.

For the *secretary problem with replacement*, we give Algorithm 7 which is a simple modification of the *online max algorithm* in [2].

Lemma 31. *Algorithm 7 returns element e with maximum $g(e, S)$ with probability $1 - \delta$, thus it achieves a $1 - \delta$ competitive ratio for the secretary problem with replacement. The size of the shortlist that it uses is logarithmic in $1/\delta$.*

Algorithm 7 Secretary Problem with Replacement

Inputs: number of items N, an independent set S, 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 $g(a_i, S) > M$ **then**
        $M \leftarrow g(a_i, S)$
        if $i \geq u$ and $|A| < L$ **then**
            $A \leftarrow A \cup \{a_i\}$
        end if
    end if
end for
return A, and $A^* := \max_{i \in A} g(a_i, S)$

Similar to [2], we divide the input into sequential blocks that we refer to as (α, β) windows. Here $k = rk(\mathcal{M})$.

Algorithm 8 Algorithm for submodular matroid secretary with shortlist

Inputs: number of items n, 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.
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 7 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

$$\begin{aligned} a_0 &:= \max_{x \in R_{1,\ldots,w-1}} \Delta(x | S_{1,\ldots,w-1} \cup \gamma(\tau) \setminus \zeta(\tau)) \\ a_\ell &:= \Delta(s_j(\ell) | S_{1,\ldots,w-1} \cup \gamma(\tau) \setminus \zeta(\tau)), \forall 0 < \ell \leq N - 1 \end{aligned}$$

where $s_j(\ell)$ denotes the ℓ^{th} item in the slot s_j.

        Let $A_j(\tau)$ be the shortlist returned by Algorithm 7 for slot j and subsequence τ. Add all items except the dummy item 0 to the shortlist A. 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 and $\bar{S}_w$ as before
    $S_{1,\cdots,w} \leftarrow S_{1,\cdots,w-1} \cup S_w \setminus \bar{S}_w$
    $A^* \leftarrow A^* \cup (S_w \cap A) \setminus \hat{S}_w$
end for
return A, A^*.

Definition 11 ((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, etc. Next, we define k/α windows, where window i consists of $\alpha\beta$ consecutive slots, in the same manner as we assigned slots.*

Intuitively, for large enough α and β, 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.

The 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. Similar to [2], 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. But the difference is that adding items from a new window to the current solution S could make it a non-independent set of $\mathcal{M}$. In order to make the new set independent we have to remove some items from S. The removed item corresponding to e will be $\theta(e, S)$. We need to take care of all the removals for newly selected items in a window. Therefore we have to slightly change the definitions in [2]. We introduce $\zeta(\tau)$ which is counterpart of $\gamma(\tau)$ for the removed elements. More precisely, for any subsequence $\tau = (s_1, \ldots, s_\ell)$ of the $\alpha\beta$ slots in window w, recall the greedy subsequence $\gamma(\tau)$ of items as:

$$\gamma(\tau) := \{i_1, \ldots, i_\ell\} \tag{5.2}$$

where

$$i_j := \arg \max_{i \in s_j \cup R_{1,\ldots,w-1}} g(i, S_{1,\ldots,w-1} \cup \{i_1, \ldots, i_{j-1}\}) \tag{5.3}$$

now define $\zeta(\tau) := \{c_1, \ldots, c_\ell\}$ where

$$c_j := \theta(i_j, S_{1,\ldots,w-1} \cup \{i_1, \ldots, i_{j-1}\}) \tag{5.4}$$

Recall the definition of R_w in [2], which is the union of all greedy subsequences of length α, and S_w to be the best subsequence among those. That is,

$$R_w = \cup_{\tau:|\tau|=\alpha} \gamma(\tau) \tag{5.5}$$

and

$$S_w = \gamma(\tau^*), \tag{5.6}$$

now define

$$\bar{S}_w = \zeta(\tau^*), \tag{5.7}$$

where

$$\tau^* := \arg\max_{\tau:|\tau|=\alpha} f((S_{1,\dots,w-1} \cup \gamma(\tau)) \setminus \zeta(\tau)) - f(S_{1,\dots,w-1}). \tag{5.8}$$

also define

$$\hat{S}_w = \{c_{j_1}, \cdots, c_{j_t}\}, \text{where } (S_w \cap A) = \{i_{j_1}, \cdots, i_{j_t}\} \tag{5.9}$$

In other words, $\hat{S}_w$ is counterpart of elements of $S_w \cap A$ that are removed by g. Also note that in the main Algorithm 8, we remove $\zeta(\tau^*)$ from $S_{1,\cdots,w-1} \cup S_w$ at the end of window w and make $S_{1,\cdots,w}$ an independent set of $\mathcal{M}$.

In order to find the item with the maximum g value (5.3), among all the items in the slot. We use an online subroutine that employs the algorithm (Algorithm 7) for the *secretary problem with replacement* described earlier. Note that R_w, S_w and $\bar{S}_w$ can be computed *exactly at the end* of window w.

The algorithm returns both the shortlist A which similar to [2] is of size $O(k)$ as stated in the following proposition, as well the set A^*. Note that we remove $\hat{S}_w$ from A^* at the end of window w. In the next section, we will show that $\mathbb{E}[f(A^*)] \geq (1 - \frac{1}{e^2} - \epsilon - O(\frac{1}{k}))f(S^*)$ to provide a bound on the competitive ratio of this algorithm. As it is proved in [2],

Proposition 8. *Given k, n, and any constant α, β and ϵ, the size of shortlist A selected by Algorithm 8 is at most $4k\beta\binom{\alpha\beta}{\alpha}\log(2/\epsilon) = O(k)$.*

5.3 Preliminaries

Definition 12. *(**Matroids**). A matroid is a finite set system $\mathcal{M} = (\mathcal{N}, \mathcal{I})$, where $\mathcal{N}$ is a set and $\mathcal{I} \subseteq 2^{\mathcal{N}}$ is a family of subsets such that: (i) $\emptyset \in I$, (ii) If $A \subseteq B \subseteq N$, and $B \in I$, then $A \in I$, (iii) If $A, B \in I$ and $|A| < |B|$, then there is an element $b \in B \setminus A$ such that $A + b \in I$. In a matroid $\mathcal{M} = (\mathcal{N}, \mathcal{I})$, N is called the ground set and the members of $\mathcal{I}$ are called independent sets of the*

matroid. The bases of $\mathcal{M}$ share a common cardinality, called the rank of $\mathcal{M}$.

Lemma 32. *(**Brualdi**) If A, B are any two bases of matroid M then there exists a bijection π from A to B, fixing $A \cap B$, such that $A - x + \pi(x) \in M$ for all $x \in A$.*

In [2], we proved some useful properties of (α, β) windows, defined in Definition 11 and used in Algorithm 8. which we summarize it in this section.

The first observation is that every item will appear uniformly at random in one of the $k\beta$ slots in (α, β) windows.

Definition 13. *For each item $e \in I$, 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 33. *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 important observations about the probability of assignment of items in S^* in the slots in a window w, given the sets $R_{1,\ldots,w-1}, S_{1,\ldots,w-1}$ (refer to (5.5), (5.6) 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 14. *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* (5.2)*. 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 34. *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 35. *For any item $i \in S^*$, window $w \in \{1, \ldots, W\}$, and slot s in window w, define*

$$p_{is} := \mathbb{P}(i \in s \cup Supp(T) | T_{1,\ldots,w-1} = T). \tag{5.10}$$

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

$$p_{is'} = p_{is''} \geq \frac{1}{k\beta}\,. \tag{5.11}$$

Lemma 36. *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.*

Lemma 37. *Fix a slot s', T, and $j \notin Supp(T)$. Suppose that there exists some configuration Y' such that $T_{1,\cdots,w-1}(Y') = T$ and $Y'_j = s'$. Then, given any configuration Y'' with $T_{1,\ldots,w-1}(Y'') = T$, we can replace Y''_j with s' to obtain a new configuration $\bar{Y}$ that also satisfies $T_{1,\ldots,w-1}(\bar{Y}) = T$.*

5.4 Analysis of the algorithms

In this section we show that for any $\epsilon \in (0, 1)$, Algorithm 8 with an appropriate choice of constants α, β, achieves the competitive ratio claimed in Theorem 23 for the submodular matroid secretary problem with shortlists.

First, 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 $f((S_{1,\ldots,w-1} \cup \gamma(\tilde{\tau}_w)) \setminus \zeta(\tilde{\tau}_w)) - f(S_{1,\ldots,w-1})$ in terms of $\mathrm{OPT} - f(S_{1,\ldots,w-1})$. This will be used to lower bound increment $f(S_{1,\ldots,w-1} \cup \gamma(\tau^*) \setminus \zeta(\tau^*)) - f(S_{1,\ldots,w-1})$ in every window.

Definition 15 (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 $i \in S^* \cap T$, 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 $Z_s \neq \emptyset$.*

Similar to [2], we have the following property for Z_s:

Lemma 38. *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 of at most one slot s.*

Lemma 39. *We can show that for all $i, i' \in S^* \backslash \{Z_{s_1} \cup \ldots \cup Z_{s_{j-1}}\}$,*

$$\Pr(i \in Z_{s_j} | Z_{s_1} \cup \ldots \cup Z_{s_{j-1}}) = \Pr(i' \in Z_{s_j} | Z_{s_1} \cup \ldots \cup Z_{s_{j-1}}) \geq \frac{1}{k} . \tag{5.12}$$

Proof. The proof is similar to Lemma 12 in [2], and it is based on Lemma 38, □

In the following lemma we lower bound the marginal gain of a randomly picked element of optimal solution in slot s_j with respect to previously selected items.

Lemma 40. *Suppose the sequence $\tilde{\tau}_w = (s_1, \ldots, s_t)$ defined as in Definition 15, let $\gamma(\tilde{\tau}_s) = (i_1, \ldots, i_t)$, with $\gamma(\cdot)$ as defined in* (5.2)*. Then, for all $j = 1, \ldots, t$,*

$$\begin{aligned}
&\mathbb{E}[\Delta_f(a | S_{1,\ldots,w-1} \cup \{i_1, \ldots, i_{j-1}\} \setminus \{c_1, \cdots, c_{j-1}\}) | T_{1,\ldots,w-1}, i_{1,\ldots,j-1}, a \in S^* \cap Z_{s_j}] \\
&\qquad \geq \frac{1}{k} \left((1 - \frac{\alpha}{k}) f(S^*) - f(S_{1,\ldots,w-1} \cup \{i_1, \ldots, i_{j-1}\} \setminus \{c_1, \cdots, c_{j-1}\}) \right)
\end{aligned}$$

Proof. We can lower bound the increment assuming a is randomly picked item from $Z_{s_j} \cap S^*$:

$$\begin{aligned}
& \mathbb{E}[\Delta_f(a, S_{1,\ldots,w-1} \cup \{i_1, \ldots, i_{j-1}\} \setminus \{c_1, \cdots, c_{j-1}\}) | T_{1,\ldots,w-1} = T, i_1, \ldots, i_{j-1}, a \in S^* \cap Z_{s_j}] \\
\geq \; & \frac{1}{k} \mathbb{E}[\sum_{a \in S^* \backslash \{Z_1, \ldots Z_{s_{j-1}}\}} \mathbb{E}[\Delta_f(a, S_{1,\ldots,w-1} \cup \{i_1, \ldots, i_{j-1}\} \setminus \{c_1, \cdots, c_{j-1}\}) | T, i_1, \ldots, i_{j-1}]] \\
\geq \; & \frac{1}{k} \mathbb{E}[\left(f(S^* \backslash \{Z_1, \ldots Z_{s_{j-1}}\}) - f(S_{1,\ldots,w-1} \cup i_{1,\ldots,s-1} \setminus \{c_1, \cdots, c_{j-1}\}) \right) | T] \\
\geq \; & \frac{1}{k} \mathbb{E}[\left(f(S^* \setminus \cup_{s' \in w} Z_{s'}) - f(S_{1,\ldots,w-1} \cup i_{1,\ldots,s-1} \setminus \{c_1, \cdots, c_{j-1}\}) \right) | T] \\
\geq \; & \frac{1}{k} \left(\left(1 - \frac{\alpha}{k} \right) f(S^*) - f(S_{1,\ldots,w-1} \cup i_{1,\ldots,s-1} \setminus \{c_1, \cdots, c_{j-1}\}) \right)
\end{aligned}$$

The last inequality uses the observation from Lemma 38 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 $S^* \setminus \cup_{s' \in w} Z_{s'}$ independently with probability $1 - \frac{\alpha}{k}$; along with Lemma 13 for submodular function f. □

Lemma 41. *Suppose the sequence $\tilde{\tau}_w = (s_1, \ldots, s_t)$ defined as in Definition 15, let $\gamma(\tilde{\tau}_s) = (i_1, \ldots, i_t)$, with $\gamma(\cdot)$ as defined in* (5.2)*. Moreover, let S' be the extension of $S_{1,\cdots,w-1} \cup \{i_1, \cdots, i_{j-1}\} \setminus \{c_1, \cdots, c_{j-1}\}$ to an independent set in $\mathcal{M}$, and π be the bijection from Brualdi lemma (refer to Lemma 32) from S^* to S'. Then, for all $j = 1, \ldots, t$,*

$$\mathbb{E}[f(S_{1,\ldots,w-1} \cup \{i_1, \ldots, i_{j-1}\} \setminus \{c_1, \cdots, c_{j-1}, \pi(a)\}) | T_{1,\ldots,w-1}, i_{1,\ldots,j-1}, a \in S^* \cap Z_{s_j}] \geq (1 - \frac{1}{k - \alpha}) f(S_{1,\ldots,w-1} \cup \{i_1, \ldots, i_{j-1}\} \setminus \{c_1, \cdots, c_{j-1}\})$$

Proof. Since π is a bijection from S^* to S', from Brualdi's lemma (lemma 32), there is an onto mapping π' from S^* to $S_{1,\cdots,w-1} \cup \{i_1, \cdots, i_{j-1}\} \setminus \{c_1, \cdots, c_{j-1}\} \cup \{\emptyset\}$ such that $S_{1,\cdots,w-1} \cup \{i_1, \cdots, i_{j-1}\} \setminus \{c_1, \cdots, c_{j-1}\} - \pi'(a) + a \in M$, for all $a \in S^*$. Further, $\pi'(a) = \pi(a)$ if $\pi(a) \in S_{1,\cdots,w-1} \cup \{i_1, \cdots, i_{j-1}\} \setminus \{c_1, \cdots, c_{j-1}\}$ and $\pi'(a) = \emptyset$ otherwise.

Recall the definition of Z_{s_j}. Suppose a is a randomly picked item from $S^* \cap Z_{s_j}$. Note that from Lemma 38, conditioned on $T_{1,\cdots,w-1}$, the element a can be equally any element of $S^* \setminus \{Z_1, \ldots Z_{s_{j-1}}\}$ with probability at least $1/(k - \alpha)$. Therefore, $\pi'(a)$ would be any of $S_{1,\cdots,w-1} \cup \{i_1, \cdots, i_{j-1}\} \setminus \{c_1, \cdots, c_{j-1}\}$ with probability at most $1/(k - \alpha)$ (since π' might map some elements of S^* to the empty set). Now based on the definition of π and lemma 13 we have:

$$\mathbb{E}_a[f(S_{1,\ldots,w-1} \cup \{i_1, \ldots, i_{j-1}\} \setminus \{c_1, \cdots, c_{j-1}, \pi(a)\}) | T_{1,\ldots,w-1}, i_{1,\ldots,j-1}, a \in S^* \cap Z_{s_j}] \geq (1 - \frac{1}{k - \alpha}) f(S_{1,\ldots,w-1} \cup \{i_1, \ldots, i_{j-1}\} \setminus \{c_1, \cdots, c_{j-1}\})$$

□

Lemma 42. *Suppose function g is as defined in equation 5.1. For the sequence $\tilde{\tau}_w = (s_1, \ldots, s_t)$, and $\gamma(\tilde{\tau}_s) = (i_1, \ldots, i_t)$. Then, for all $j = 1, \ldots, t$,*

$$\begin{aligned}
&\mathbb{E}[g(i_j, S_{1,\ldots,w-1} \cup \{i_1, \ldots, i_{j-1}\} \setminus \{c_1, \cdots, c_{j-1}\}) | T_{1,\ldots,w-1}, i_{1,\ldots,j-1}] \\
\geq\ & \frac{1}{k}\left((1 - \frac{\alpha}{k-\alpha}) f(S^*) - 2f(S_{1,\ldots,w-1} \cup \{i_1, \ldots, i_{j-1}\} \setminus \{c_1, \cdots, c_{j-1}\})\right)
\end{aligned}$$

Proof. In the algorithm 8, at the end of window w, we set $S_{1,\cdots,w} = S_{1,\cdots,w-1} \cup S_w \setminus \bar{S}_w$. Suppose $a \in s_j \cap S^*$. Moreover, let S' be the extension of $S_{1,\cdots,w-1} \cup \{i_1, \cdots, i_{j-1}\} \setminus \{c_1, \cdots, c_{j-1}\}$ to an independent set in $\mathcal{M}$, and π be the bijection from Brualdi lemma (refer to Lemma 32) from S^* to S'. Thus the expected value of the function g on the element selected by the algorithm in slot s_j (the element with maximum g in the slot s_j) would be

$$\begin{aligned}
& \mathbb{E}[f(S_{1,\ldots,w-1} \cup \{i_1, \ldots, i_j\} \setminus \{c_1, \cdots, c_j\} | T_{1,\ldots,w-1}, i_{1,\ldots,j-1}] \\
\geq\ & \mathbb{E}[f(S_{1,\ldots,w-1} \cup \{i_1, \ldots, i_{j-1}, a\} \setminus \{c_1, \cdots, c_{j-1}, \pi(a)\} | T_{1,\ldots,w-1}, i_{1,\ldots,j-1}, a \in S^* \cap Z_{s_j}] \\
\geq\ & \mathbb{E}[f(S_{1,\ldots,w-1} \cup \{i_1, \ldots, i_{j-1}\} \setminus \{c_1, \cdots, c_{j-1}, \pi(a)\}) | T_{1,\ldots,w-1}, i_{1,\ldots,j-1}, a \in S^* \cap Z_{s_j}] \\
& + \mathbb{E}[\Delta_f(a | S_{1,\cdots,w-1} \cup \{i_1, \cdots, i_{j-1}\} \setminus \{c_1, \cdots, c_{j-1}, \pi(a)\}) | T_{1,\ldots,w-1}, i_{1,\ldots,j-1}, a \in S^* \cap Z_{s_j}] \\
\geq\ & \mathbb{E}[f(S_{1,\ldots,w-1} \cup \{i_1, \ldots, i_{j-1}\} \setminus \{c_1, \cdots, c_{j-1}, \pi(a)\}) | T_{1,\ldots,w-1}, i_{1,\ldots,j-1}, a \in S^* \cap Z_{s_j}] \\
& + \mathbb{E}[\Delta_f(a | S_{1,\cdots,w-1} \cup \{i_1, \cdots, i_{j-1}\} \setminus \{c_1, \cdots, c_{j-1}\}) | T_{1,\ldots,w-1}, i_{1,\ldots,j-1}, a \in S^* \cap Z_{s_j}]
\end{aligned}$$

The first inequality is from the definition of function g as it is defined in equation 5.1. The last

inequality from submoularity of f. Now from the last inequality and lemma 41 we have

$$\begin{aligned}
&\mathbb{E}[f(S_{1,\ldots,w-1} \cup \{i_1,\ldots,i_j\} \setminus \{c_1,\cdots,c_j\}|T_{1,\ldots,w-1}, i_{1,\ldots,j-1})] \\
&\geq (1-\frac{1}{k-\alpha})f(S_{1,\cdots,w-1} \cup \{i_1,\ldots,i_{j-1}\} \setminus \{c_1,\cdots,c_{j-1}\}) \\
&+ \mathbb{E}[\Delta_f(a|S_{1,\ldots,w-1} \cup \{i_1,\ldots,i_{j-1}\} \setminus \{c_1,\cdots,c_{j-1}\})||T_{1,\ldots,w-1}, i_{1,\ldots,j-1}, a \in S^* \cap Z_{s_j}].
\end{aligned}$$

Now from lemma 40 and the above inequality we can show

$$\begin{aligned}
&\mathbb{E}[f(S_{1,\ldots,w-1} \cup \{i_1,\ldots,i_j\} \setminus \{c_1,\cdots,c_j\}|T_{1,\ldots,w-1}, i_{1,\ldots,j-1})] \\
&\geq (1-\frac{1}{k-\alpha})f(S_{1,\cdots,w-1} \cup \{i_1,\ldots,i_{j-1}\} \setminus \{c_1,\cdots,c_{j-1}\}) \\
&+ \frac{1}{k}\left((1-\frac{\alpha}{k})f(S^*) - f(S_{1,\ldots,w-1} \cup \{i_1,\ldots,i_{j-1}\} \setminus \{c_1,\cdots,c_{j-1}\})\right) .
\end{aligned}$$

Thus,

$$\begin{aligned}
f(S_{1,\ldots,w-1} \cup \{i_1,\ldots,i_j\} \setminus \{c_1,\cdots,c_j\}) - f(S_{1,\cdots,w-1} \cup \{i_1,\ldots,i_{j-1}\} \setminus \{c_1,\cdots,c_{j-1}\}) \quad (5.13)\\
\geq \frac{1}{k}\left((1-\frac{\alpha}{k})f(S^*) - 2f(S_{1,\ldots,w-1} \cup \{i_1,\cdots,i_{j-1}\} \setminus \{c_1,\cdots,c_{j-1}\})\right) . \quad (5.14)
\end{aligned}$$

□

Using standard techniques for the analysis of greedy algorithm, the following corollary of the previous lemma can be derived,

Lemma 43.

$$\mathbb{E}\left[\left(1-\frac{\alpha}{k}\right)f(S^*) - 2f(S_{1,\ldots,w-1} \cup \gamma(\tilde{\tau}_w) \setminus \zeta(\tilde{\tau}_w))|T\right] \leq \mathbb{E}\left[e^{-\frac{2|\tilde{\tau}_w|}{k}} \mid T\right]\left(\left(1-\frac{\alpha}{k}\right)f(S^*) - 2f(S_{1,\ldots,w-1})\right)$$

Proof. Let $\pi_0 = (1-\frac{\alpha}{k})f(S^*) - 2\mathbb{E}[f(S_{1,\ldots,w-1})|T_{1,\ldots,w-1} = T]$, and for $j \geq 1$,

$$\pi_j := (1-\frac{\alpha}{k})f(S^*) - 2\mathbb{E}[f(S_{1,\ldots,w-1} \cup \{i_1,\ldots,i_j\} \setminus \{c_1,\cdots,c_j\})|T_{1,\ldots,w-1} = T, i_1,\ldots,i_{j-1}],$$

Then, subtracting and adding $\frac{1}{2}(1-\frac{\alpha}{k})f(S^*)$ from the left hand side of lemma 42, and taking expectation conditional on $T_{1,\ldots,w-1} = T, i_1, \ldots, i_{j-2}$, we get

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

which implies

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

By martingale stopping theorem, this implies:

$$\mathbb{E}[\pi_t|T] \leq \mathbb{E}\left[\left(1-\frac{2}{k}\right)^t |T\right]\pi_0 \leq \mathbb{E}\left[e^{-2t/k}|T\right]\pi_0 \ .$$

where stopping time $t = |\tilde{\tau}_w|$. ($t = |\tilde{\tau}_w| \leq \alpha\beta$ is bounded, therefore, 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 (5.6) and (5.8)). To compare it with $\tilde{\tau}_w$, we need a bound on size of $\tilde{\tau}_w$. We use concentration inequalities proved in [2]:

Lemma 44 (proved in [2])**.** *For any real $\delta' \in (0,1)$, if parameters k, α, β satisfy $k \geq \alpha\beta$, $\beta \geq \frac{8}{(\delta')^2}$, $\alpha \geq 8\beta^2\log(1/\delta')$, then given any $T_{1,\ldots,w-1} = T$, with probability at least $1-\delta' e^{-\alpha/k}$,*

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

Lemma 45. *For any real $\delta' \in (0,1)$, if parameters k, α, β satisfy $k \geq \alpha\beta$, $\beta \geq \frac{8}{(\delta')^2}$, $\alpha \geq 8\beta^2\log(1/\delta')$, then*

$$\mathbb{E}\left[\frac{k-\alpha}{k}OPT - 2f(S_{1,\ldots,w})|T_{1,\ldots,w-1}\right] \leq (1-\delta')e^{-2\alpha/k}\left(\frac{k-\alpha}{k}OPT - 2f(S_{1,\ldots,w-1})\right) \ .$$

Proof. The lemma follows from substituting Lemma 44 in Lemma 43. □

Theorem 26. *For any constant $\epsilon > 0$, there exists an online algorithm (Algorithm 8) for the submodular matroid secretary problem with shortlists that achieves a competitive ratio of $\frac{1}{2}(1 - \frac{1}{e^2} - \epsilon - O(\frac{1}{k}))$, with shortlist of size $\eta_\epsilon(k) = O(k)$. Here, $\eta_\epsilon(k) = O(2^{poly(1/\epsilon)}k)$. The running time of this online algorithm is $O(nk)$.*

Proof. Now from Lemma 45, we have, for any real $\delta' \in (0, 1)$, if parameters k, α, β satisfy $k \geq \alpha\beta$, $\beta \geq \frac{8}{(\delta')^2}$, $\alpha \geq 8\beta^2 \log(1/\delta')$, then the set $S_{1,\ldots,W}$ tracked by Algorithm 8 satisfies

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

Now, 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 8. The main difference between the two sets is that in construction of shortlist A, Algorithm 7 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.

Similar to Lemma 16 in [2], we can show that each element in A gets selected by the algorithm with probability at least $1 - \delta$. More precisely, let A be the shortlist returned by Algorithm 8, and δ is the parameter used to call Algorithm 7 in Algorithm 8. Then, for given configuration Y, for any item a, we have

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

Therefore using Lemma 13,

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

where $A^* := S_{1,\cdots,W} \cap A$ is the subset of shortlist A returned by Algorithm 8. The proof is similar to the proof in [2].

□

5.4.1 Preemption model and Shorlitst of size at most k

Finally we focus on the special case where the size of shortlist is at most k. We can get a constant competitive algorithm even with the slight relaxation of the *matroid secretary problem* to the case that we allow the algorithm to select a shortlist of size at most $k = rk(\mathcal{M})$. The algorithm finally outputs an independent subset of this shortlist of size k. There was no constant competttetive algorithm even for this natural relaxation of *matroid secretary problem*. Also we are not aware of any direct way to prove a constant factor guarantee for this simple relaxation without using the techniques that we develop using (α, β)-windows.

Theorem 27. *For the matroid secretary problem in preemption model, and matroid secretary problem that uses shortlist of size at most $\eta(k) = k$, there is an algorithm that achieves competitive ratio $\frac{1}{2}(1 - 1/e)(1 - 1/e^2 - \epsilon)$.*

Proof. We show that algorithm 8 with parameter $\alpha = \beta = 1$ satisfies the above mentioned properties. Firstly, algorithm 8 (with $\alpha = 1$, and $\beta = 1$) uses shortlist of size $\eta(k) \leq k$. The reason is that the algorithm divides the input into exactly k slots. Also each window contains exactly one slot. The function γ tries all α-subsequences of a window which is exactly one slot. Thus γ returns one element in that slot with hight value of $g(e, S)$ as defined in 5.1, which might cause removal of at most one element $\theta(S, e)$ from the current solution S. Therefore the algorithm has shortlist size at most k and also satisfies the preemption model. Now by setting $\alpha = 1, \beta = 1$ we can get a constant competttetive ratio that the error rate comes from lemma 45.

□

5.5 p-matchoid constraints

In this section, we present algorithms for monotone submodular function maximization subject to p-matchoid constraints. These constraints generalize many basic combinatorial constraints such as the cardinality constraint, the intersection of p matroids, and matchings in graphs. Throughout

this section, k would refer to the size of the largest feasible set. A formal definition of a p-matchoid is as follows:

Definition 16. *(**Matchoids**). Let $\mathcal{M}_1 = (\mathcal{N}_1, \mathcal{I}_1), \cdots, \mathcal{M}_q = (\mathcal{N}_q, \mathcal{I}_q)$ be q matroids over overlapping groundsets. Let $\mathcal{N} = \mathcal{N}_1 \cup \cdots \cup \mathcal{N}_q$ and $\mathcal{I} = \{S \subseteq \mathcal{N} : S \cap \mathcal{N} \in \mathcal{I}_\ell, \forall \ell\}$. The finite set system $\mathcal{M}_p = (\mathcal{N}, \mathcal{I})$ is a p-matchoid if for every element $e \in \mathcal{N}$, e is a member of $\mathcal{N}$ for at most p indices $\ell \in [q]$.*

There are some subtle differences in the algorithm as well as in the analysis. The main difference in the algorithm is that instead of removing one item from the current independent set S, we might remove up to p items form S. Each removed item corresponds to different ground sett N_i, in which the new item lies (based on the definition of p-matchoid constraints, Definition 16, there are at most p such elements).

For each index $\ell \in [q]$ define:

$$\Omega_\ell(e, S) := \{e' \in S | S + e - e' \in \mathcal{I}_\ell\} \tag{5.15}$$

For an element e in the input, suppose $e \in N_{\ell_i}$, for $i = 1, \cdots, p$. Define

$$\lambda(e, S) := \prod_{i=1}^{p} \Omega_{\ell_i}(e, S). \tag{5.16}$$

For a combination vector $r = (r_1, \cdots, r_p) \in \lambda(e, S)$, where $r_i \in \Omega_{\ell_i}(e, S)$, define:

$$\mu(r) := \{r_1, \cdots, r_p\}, \tag{5.17}$$

and

$$g_r(e, S) := f(S + e - \mu(r)) - f(S). \tag{5.18}$$

Also define:

$$\theta(e, S) := \mu(\arg \max_{r \in \lambda(e,S)} g_r(e, S)). \tag{5.19}$$

Furthermore define,

$$g(e, S) := \max_{r \in \lambda(e,S)} g_r(e, S). \tag{5.20}$$

As in the online subroutine for the main algorithm, we run Algorithm 7 with the new function g defined in equation 5.20. It returns element e with maximum $g(e, S)$, and it achieves a $1 - \delta$ competitive ratio with shortlists of size logarithmic in $1/\delta$.

Additionally, we make some changes in the main algorithm 8. In particular, we define γ similar to equation 5.2 but using the new definition of g in equation 5.20. Moreover, for a subsequence $\tau = (s_1, \ldots, s_\ell)$ define

$$\zeta(\tau) := \bigcup_{j=1}^{\ell} C_j, \tag{5.21}$$

where each C_j is a set defined as

$$C_j := \theta(i_j, S_{1,\ldots,w-1} \cup \{i_1, \ldots, i_{j-1}\}). \tag{5.22}$$

Note that in contrast with the definition of $\zeta(\tau)$ for the matroid constraints equation 5.22, in which c_j is only one item, now each C_j is a subset of the current independent set S. Further, the definition of $\bar{S}_w$, in equation 5.7, will be updated accordingly using the new definition of $\zeta(\tau)$.

Now we can generalize Lemma 41 to p-matchoid constraints.

Lemma 46. *Suppose the sequence $\tilde{\tau}_w = (s_1, \ldots, s_t)$ defined as in Definition 15, let $\gamma(\tilde{\tau}_s) = (i_1, \ldots, i_t)$, with $\gamma(\cdot)$ as defined in* (5.2)*. For any $j \in \{1, \ldots, t\}$, and element $b \in \mathcal{N}_\ell$, let S'_ℓ be the extension of $S_{1,\cdots,w-1} \cup \{i_1, \cdots, i_{j-1}\} \setminus \bigcup_{r \le j-1} C_r$ to an independent set in $\mathcal{M}_\ell$, and π_ℓ be the bijection from Brualdi lemma (refer to Lemma 32) from S^* to S'_ℓ. Further, let's denote by $\pi(b) := \{\pi_\ell(b) | b \in \mathcal{N}_\ell\}$, then*

$$\begin{aligned}\mathbb{E}[f(S_{1,\ldots,w-1} \cup \{i_1, \ldots, i_{j-1}\} \setminus (\bigcup_{r \le j-1} C_r \cup \pi(a) | T_{1,\ldots,w-1}, i_{1,\ldots,j-1}, a \in S^* \cap Z_{s_j}] \\ \ge (1 - \frac{p}{k}) f(S_{1,\ldots,w-1} \cup \{i_1, \ldots, i_{j-1}\} \setminus \{c_1, \cdots, c_{j-1}\}).\end{aligned}$$

Proof. The proof is similar to the proof of Lemma 41. For $\ell \in [q]$, since π_ℓ is a bijection from

$S^* \cap \mathcal{N}_\ell$ to S'_ℓ, from Brualdi's lemma (lemma 32), there is an onto mapping π'_ℓ from $S^* \cap \mathcal{N}_\ell$ to $S_{1,\cdots,w-1} \cup \{i_1, \cdots, i_{j-1}\} \setminus (\bigcup_{r \le j-1} C_r \cup \pi(a)) \cup \{\emptyset\}$ such that $S_{1,\cdots,w-1} \cup \{i_1, \cdots, i_{j-1}\} \setminus (\bigcup_{r \le j-1} C_r \cup \pi(a)) - \pi'_\ell(a) + a \in M_\ell$, for all $a \in S^*$. Further, $\pi'_\ell(a) = \pi_\ell(a)$ if $\pi_\ell(a) \in S_{1,\cdots,w-1} \cup \{i_1, \cdots, i_{j-1}\} \setminus \bigcup_{r \le j-1} C_r$ and $\pi'_\ell(a) = \emptyset$ otherwise.

Recall the definition of Z_{s_j} (refer to definition 15). Suppose a is a randomly picked item from $S^* \cap Z_{s_j}$. Note that from Lemma 38, conditioned on $T_{1,\cdots,w-1}$, the element a can be equally any element of $S^* \setminus \{Z_1, \ldots Z_{s_{j-1}}\}$ with probability at least $1/k$. Therefore, $\pi'_\ell(a)$ would be any of $S_{1,\cdots,w-1} \cup \{i_1, \cdots, i_{j-1}\} \setminus \bigcup_{r \le j-1} C_r$ with probability at most $1/k$ (since π'_ℓ might map some elements of S^* to the empty set).

For element $e \in S_{1,\cdots,w-1} \cup \{i_1, \cdots, i_{j-1}\} \setminus \bigcup_{r \le j-1} C_r$, let $\mathcal{N}(e)$ be the set of indices ℓ such that $e \in \mathcal{N}_\ell$. Because of the p-matchoid constraint, we have $|\mathcal{N}(e)| \le p$. Define

$$\pi^{-1}(e) := \{t | t \in \mathcal{N}_\ell, \text{ for some } \ell \in \mathcal{N}(e) \text{ and } \pi_\ell(t) = e\}.$$

We have also $|\pi^{-1}(e)| \le p$. Thus, each element $e \in S_{1,\cdots,w-1} \cup \{i_1, \cdots, i_{j-1}\} \setminus \bigcup_{r \le j-1} C_r$ belongs to $\pi(a)$ with probability at most p/k:

$$\Pr(e \in \pi(a) | a \in S^* \cap Z_{s_j}) = \Pr(a \in S^* \cap Z_{s_j} \cap \pi^{-1}(a)) \le \frac{p}{k}.$$

Now we apply Lemma 13. It is crucial to note that in Lemma 13 each element do not need to be selected necessarily independently. Definition of π and lemma 13 imply the lemma. □

Furthermore the main difference in the analysis is that instead of recursion 5.13, we get the following new recursion:

Lemma 47. *Suppose function g is as defined in equation 5.20. For the sequence $\tilde{\tau}_w = (s_1, \ldots, s_t)$, and $\gamma(\tilde{\tau}_s) = (i_1, \ldots, i_t)$. Then, for all $j = 1, \ldots, t$,*

$$\mathbb{E}\left[g(i_j, S_{1,\dots,w-1} \cup \{i_1, \dots, i_{j-1}\} \setminus \bigcup_{r \le j-1} C_r) | T_{1,\dots,w-1}, i_{1,\dots,j-1}\right]$$
$$\geq \frac{1}{k}\left((1 - \frac{\alpha}{k}) f(S^*) - (p+1) f(S_{1,\dots,w-1} \cup \{i_1, \dots, i_{j-1}\} \setminus \bigcup_{r \le j-1} C_r)\right)$$

Proof. The proof is similar to the proof of Lemma 42 with some changes regarding matchoid constraints. In the algorithm 8, at the end of window w, we set $S_{1,\cdots,w} = S_{1,\cdots,w-1} \cup S_w \setminus \bar{S}_w$. Suppose $a \in s_j \cap S^*$. Moreover, let S'_ℓ be the extension of $S_{1,\cdots,w-1} \cup \{i_1, \cdots, i_{j-1}\} \setminus \bigcup_{r \le j-1} C_r$ to an independent set in $\mathcal{M}_\ell$, and π_ℓ be the bijection in Brualdi lemma (refer to Lemma 32) from S^*_ℓ to S'_ℓ. Thus the expected value of the function g on the element selected by the algorithm in slot s_j (the element with maximum g in the slot s_j) would be

$$\mathbb{E}[f(S_{1,\dots,w-1} \cup \{i_1, \dots, i_j\} \setminus \bigcup_{r \le j} C_r) | T_{1,\dots,w-1}, i_{1,\dots,j-1}]$$
$$\geq \quad \mathbb{E}[f(S_{1,\dots,w-1} \cup \{i_1, \dots, i_{j-1}, a\} \setminus \bigcup_{r \le j-1} C_r \cup \pi(a) | T_{1,\dots,w-1}, i_{1,\dots,j-1}, a \in S^* \cap Z_{s_j}]$$
$$\geq \quad \mathbb{E}[f(S_{1,\dots,w-1} \cup \{i_1, \dots, i_{j-1}\} \setminus \bigcup_{r \le j-1} C_r \cup \pi(a)) | T_{1,\dots,w-1}, i_{1,\dots,j-1}, a \in S^* \cap Z_{s_j}]$$
$$+\mathbb{E}[\Delta_f(a | S_{1,\cdots,w-1} \cup \{i_1, \cdots, i_{j-1}\} \setminus \bigcup_{r \le j-1} C_r \cup \pi(a)) | T_{1,\dots,w-1}, i_{1,\dots,j-1}, a \in S^* \cap Z_{s_j}]$$
$$\geq \quad \mathbb{E}[f(S_{1,\dots,w-1} \cup \{i_1, \dots, i_{j-1}\} \setminus \bigcup_{r \le j-1} C_r \cup \pi(a)) | T_{1,\dots,w-1}, i_{1,\dots,j-1}, a \in S^* \cap Z_{s_j}]$$
$$+\mathbb{E}[\Delta_f(a | S_{1,\cdots,w-1} \cup \{i_1, \cdots, i_{j-1}\} \setminus \bigcup_{r \le j-1} C_r) | T_{1,\dots,w-1}, i_{1,\dots,j-1}, a \in S^* \cap Z_{s_j}]$$

The first inequality is from the definition of function g as it is defined in equation 5.20 and the fact that the algrotihm selects an element in slot s_j with maximum value of g. The second inequality is from submodularity and the last inequality is from monotonicity of f. Now from the last inequality

and Lemma 47, we can show,

$$\mathbb{E}[f(S_{1,\dots,w-1} \cup \{i_1,\dots,i_j\} \setminus \bigcup_{r\le j} C_r)|T_{1,\dots,w-1}, i_{1,\dots,j-1}]$$
$$\geq (1-\frac{p}{k}) f(S_{1,\cdots,w-1} \cup \{i_1,\dots,i_{j-1}\} \setminus \bigcup_{r\le j-1} C_r)$$
$$+ \mathbb{E}[\Delta_f(a|S_{1,\dots,w-1} \cup \{i_1,\dots,i_{j-1}\} \setminus \bigcup_{r\le j-1} C_r)||T_{1,\dots,w-1}, i_{1,\dots,j-1}, a \in S^* \cap Z_{s_j}]$$

Now from lemma 40 and the above inequality we can show

$$\mathbb{E}[f(S_{1,\dots,w-1} \cup \{i_1,\dots,i_j\} \setminus \bigcup_{r\le j} C_r)|T_{1,\dots,w-1}, i_{1,\dots,j-1}]$$
$$\geq (1-\frac{p}{k}) f(S_{1,\cdots,w-1} \cup \{i_1,\dots,i_{j-1}\} \setminus \bigcup_{r\le j-1} C_r)$$
$$+ \frac{1}{k}\left((1-\frac{\alpha}{k}) f(S^*) - f(S_{1,\dots,w-1} \cup \{i_1,\dots,i_{j-1}\} \setminus \bigcup_{r\le j-1} C_r)\right).$$

Thus,

$$f(S_{1,\dots,w-1} \cup \{i_1,\dots,i_j\} \setminus \bigcup_{r\le j} C_r) - f(S_{1,\cdots,w-1} \cup \{i_1,\dots,i_{j-1}\} \setminus \bigcup_{r\le j-1} C_r) \tag{5.23}$$
$$\geq \frac{1}{k}\left((1-\frac{\alpha}{k}) f(S^*) - (p+1) f(S_{1,\dots,w-1} \cup \{i_1,\cdots,i_{j-1}\} \setminus \bigcup_{r\le j-1} C_r)\right). \tag{5.24}$$

□

By solving the recursion and similar to the analysis for matroid constraints we can show the following theorem:

Theorem 28. *For any constant $\epsilon > 0$, there exists an online algorithm for the submodular secretary problem with p-matchoid constraints that achieves a competitive ratio of $\frac{1}{p+1}(1 - \frac{1}{e^{p+1}} - \epsilon - O(\frac{1}{k}))$, with shortlist of size $\eta_\epsilon(k) = O(k)$. Here, $\eta_\epsilon(k) = O(2^{poly(1/\epsilon)}k)$. The running time of this online algorithm is $O(n\kappa^p)$, where $\kappa = \max_{i\in[q]} rk(\mathcal{M}_i)$.*

5.6 Streaming

In this section, we show that Algorithm 8 can be implemented in a way that it uses a memory buffer of size at most $\eta(k) = O(k)$; also we compute the number of objective function evaluations for each arriving item as follows.

Theorem 29. *For any constant $\epsilon \in (0, 1)$, there exists an algorithm for the submodular random order streaming problem with matroid constraints that achieves $\frac{1}{2}(1 - \frac{1}{e} - \epsilon - O(\frac{1}{k}))$ 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 $O(pk + \frac{k^2}{n})$.*

Similarly for p-matchoid constraint we have the following result for the streaming setting:

Theorem 30. *For any constant $\epsilon > 0$, there exists an algorithm for the submodular random order streaming problem with p-matchoid constraints that achieves $\frac{1}{p+1}(1 - \frac{1}{e^{p+1}} - \epsilon - O(\frac{1}{k}))$ 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 $O(p\kappa + \kappa^p + \frac{k^2}{n})$, where $\kappa = \max_{i\in[q]} rk(\mathcal{M}_i)$.*

Proof. Th difference between Algorithm 8 in this paper and the main Algorithm in [2] is that, we remove elements of $\bar{S}_w$ from S at the end of each window w. Therefore, with the same argument in the proof of Theorem 2 in [2], we keep track of all parameters in Algorithm 8 including $\bar{S}_w, S_w, R_w, \hat{S}_w$ in a memory efficient way using memory $O(k)$. The other difference between the two algorithms is in the subroutine 7 that finds the element with maximum g in a slot. In [2], $g(e, S)$ can be computed using only one oracle access, whereas in the new definition of g in equation 5.20, we need access to independence oracle of p matroids that e belongs to, in order to check the independence of $S + e - e'$ for each $e' \in S$. At most $p\kappa$ elements of S are eligible (they are in the ground set of a matroid that e also member of). Hence, in order to create $\Omega_\ell(e, S)$, for each arriving element e in the input, we need $O(p\kappa)$ access to Independence oracle. Similarly the total access to the value oracle is $O(p\kappa)$. In order to compute $\lambda(e, S)$, we need to consider all κ^p combinations and

have access to value oracle. Therefore the number of access to the value oracle is $O(p\kappa + \kappa^p)$ per element. But, since the first element a_0 is computed in the beginning of each slot for each τ, we would have in average an additional $O(k^2/n)$ function evaluation per element. □

In the next section, we empirically compare our streaming algorithms with the state of the art algorithms in the streaming setting.

5.6.1 Experiments

In this section, we consider different types of constraints including uniform matroid, intersection of partition matroids and p-matchoid constraints. We compare our algorithm with state of the art algorithm for each type of constraint using YouTube dataset and Twitter dataset described in the next section.

DataSets

The experiments will be on a Twitter stream summarization task and a YouTube Video summarization task similar to the one in Kazemi et al. [32].

Twitter Stream Summarization In this application, we want to produce real-time summaries for Twitter feeds. It is valuable to create a succinct summary that contains all the important information. We use the dataset created by [32]. They gather recent tweets from 30 different popular news accounts, to collect a total of 42,104 unique tweets. They also define a monotone submodular function f that measure the redundancy of important stories in a set S. It is defined as follows on a set $S \subseteq V$ of tweets:

$$f(S) := \sum_{w \in W} \sqrt{\sum_{e \in S} score(w, e)},$$

where function f is defined over a ground set V of tweets. Each tweet $e \in V$ consists of a positive value vale denoting its number of retweets and a set of ℓ_e keywords $W_e = \{w_{e,1}, \cdots, w_{e,\ell_e}\}$ from a general set of keywords W. The score of a word $w \in W_e$ for a tweet e is defined by $score(w, e) = vale_e$. If $w \notin W_e$. Define $score(w, e) = 0$.

YouTube Video Summarization For the YouTube dataset, we want to select a subset of frames from video feeds which are representative of the entire video. We use the same dataset as in [32], which is YouTube videos of New Year's Eve celebrations from ten different cities around the world.

They compresses each frame into a 4-dimensional representative vector. Given a ground set V of such vectors, define a matrix M such that $M_{ij} = e - dist(v_i, v_j)$, where $dist(v_i, v_j)$ is the euclidean distance between vectors $v_i, v_j \in V$. Intuitively, M_{ij} encodes the similarity between the frames represented by v_i and v_j. They define a function that intuitively measure the diversity of the vectors in a set S as follows: $f(S) = \log det(I + \alpha M_S)$, where I is the identity matrix, $\alpha > 0$ and M_S is the principal sub-matrix of M indexed by S.

Uniform Matroid

The simplest constraint that we can impose is the uniform matroid or equivalently the cardinality constraint. In the simplest form our algorithm is similar to [2]. We compare our algorithm to the state of the art algorithm in the streaming setting [32]. As we established an upper bound on the constant factor $\eta_\epsilon(k)$ in theorem 29, the performance of our algorithm crucially depends on the choice of α and β. The running time also is a function of α and β, and it grows rapidly as we increase α and β. Surprisingly, our algorithm outperforms [32] substantially even with relatively small choices of $\alpha = 6$ and $\beta = 2$. We also observe that the utility of the output returned by our algorithm can be very close to what the optimal offline algorithm, namely the Greedy algorithm achieves. In Figure 5.1, we have plotted the performance of all three algorithms on the YouTube dataset. Note that in our experiment we use a simplistic version of our algorithm in which we subsample from the shortlist in beginning of each window and only use that subsample rather than the entire shortlist. Furthermore we observe that our algorithm is slower than [32], but the interesting fact about our algorithm as stated in Theorem 29 is that it is highly parallel thus it has the potential to become $\eta_\epsilon(k)$ times faster.

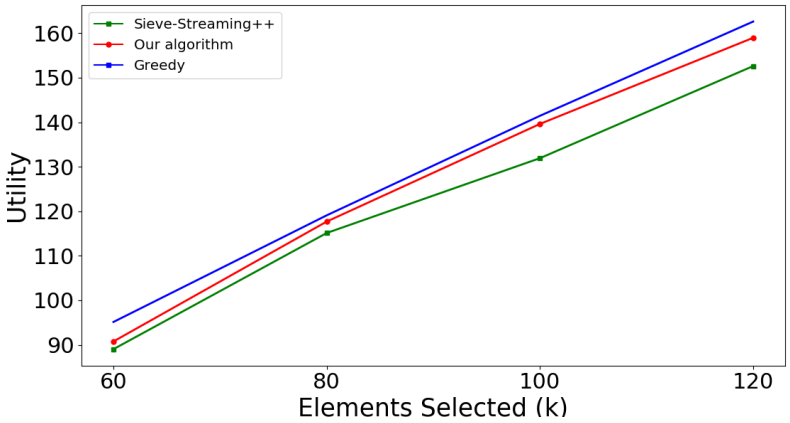

Figure 5.1: The plot is for unifrom matroid, $\alpha = 6$ and $\beta = 2$

p-matchoid constraints

For the case of p-matchoid constraints, state of the art algorithm for general streaming setting is due to Feldman et al. [17]. In our experiment, we divide the elements of input into q categories $\mathcal{N} = \mathcal{N}_1 \cup \cdots, \cup \mathcal{N}_q$. We assign p tags to each element e. Each tag belongs to one of the catergories $1, \cdots, q$ (generated randomly). Further, we impose a cardinality constraint 3 for each category (i.e, $\mathcal{I}_\ell$ is a cardinality constraint). The objective is to select at most 3 elements from each category. In other words, an independent set of p-matchoid is defined as

$$\mathcal{I} = \{S \subseteq \mathcal{N} : |S \cap \mathcal{N}_i| \leq 3, \forall i \in [q]\}.$$

In our algorithm, we set $\alpha = 3$ and $\beta = 2$. We have plotted the performance of our algorithms and [17] on the Twitter dataset below. The first plot, Figure 5.2, is for fixed $p = 3$ and different number of categories q. The second plot, Figure 5.3, is for fixed number of categories $q = 30$ and different values of p from $1, \cdots, 10$. As the competitive ratio of our algorithm suggests, by increasing p the ratio of our utility versus the utility of [17] increases. Furthermore, as the plots shows we outperform the state of the art algorithm [17] for different values of p even with relatively small α and β.

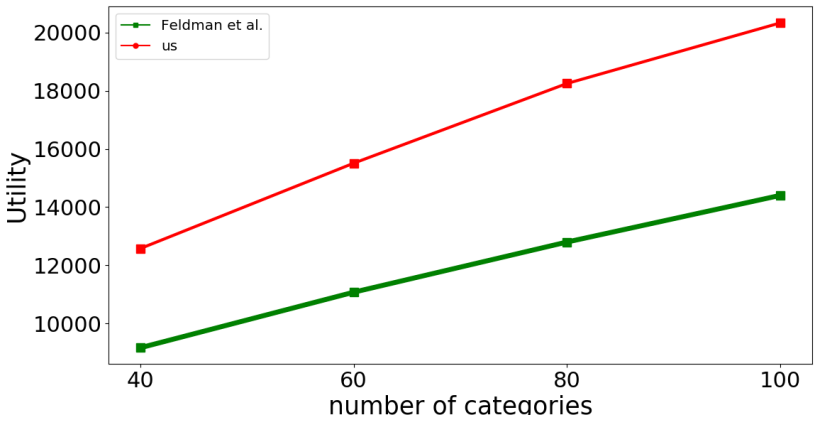

Figure 5.2: The plot is for 3-matchoid constraint, and $\alpha = 3$, $\beta = 2$

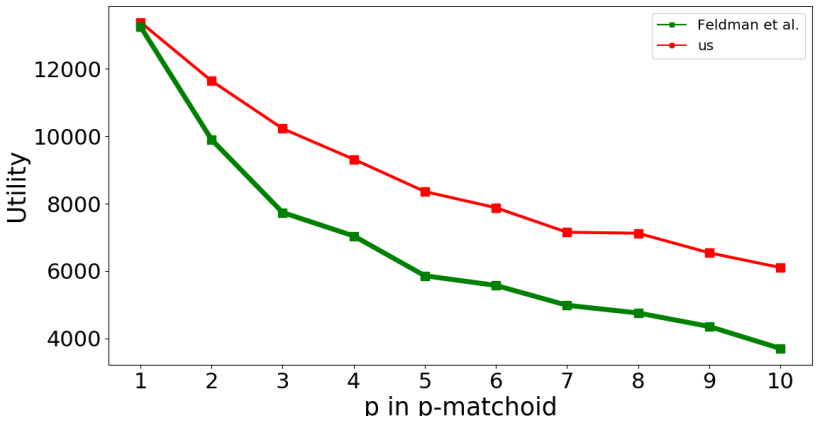

Figure 5.3: The plot is for p-matchoid constraint, for $p = 1, \cdots, 10$, and $\alpha = 3$, $\beta = 2$ and fixed $q = 30$.

Conclusion

In this thesis, we studied the submodular secretary problem. We introduce a version of this problem in which we relax the selection criteria and let the algorithm to return a subset of the shortlist as output. We further showed the link between this new model and the random order streaming model, and converted our algorithms to the random order streaming model. We improved state of the art result in this setting. Our main result is an online algorithm for 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 $\eta(k) = O(k)$. 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 $1 - 1/e$. We further generalized our results to the case of matroid and p-matchoid constraints. We design an algorithm that achieves a $\frac{1}{2}(1 - 1/e^2 - \epsilon - O(1/k))$ competitive ratio for any constant $\epsilon > 0$, using a shortlist of size $O(k)$. This is especially surprising considering that the best known competitive ratio for the matroid secretary problem is $O(\log \log k)$. We are also able to get a constant competitive algorithm using shortlist of size at most k and also a constant competitive algorithm in the preemption model. An important application of our algorithm is for the random order streaming of submodular functions. We show that our algorithm can be implemented in the streaming setting using $O(k)$ memory. It achieves a $\frac{1}{2}(1 - 1/e^2 - \epsilon - O(1/k))$ approximation. The previously best known approximation ratio for streaming submodular maximization under matroid constraint is 0.25 (adversarial order) due to [17], [12] and [11]. Moreover, we generalize our results to the case of p-matchoid constraints and give a $\frac{1}{p+1}(1 - 1/e^{p+1} - \epsilon - O(1/k))$ approximation using $O(k)$ memory, which asymptotically (as

p and k increase) approaches the best known offline guarantee $\frac{1}{p+1}$ [43]. Finally we evaluated our algorithm on real world datasets such YouTube video and Twitter streams. We empirically compared our results with state of the art results in each settings.

References

[1] M. Abam, M. Rezaei Seraji, and M Shadravan, "Online conflict-free coloring of intervals", Scientia Iranica, vol. 21, no. 6, pp. 2138–2141, 2014.

[2] S. Agrawal, M. Shadravan, and C. Stein, Submodular secretary problem with shortlists, 2018. arXiv: 1809.05082 [cs.DS].

[3] S. Agrawal, Z. Wang, and Y. Ye, "A dynamic near-optimal algorithm for online linear programming", Operations Research, vol. 62, no. 4, pp. 876–890, 2014.

[4] M. Ajtai, N. Megiddo, and O. Waarts, "Improved algorithms and analysis for secretary problems and generalizations", SIAM J. Discret. Math., vol. 14, no. 1, pp. 1–27, Jan. 2001.

[5] M. Babaioff, N. Immorlica, D. Kempe, and R. Kleinberg, "Online auctions and generalized secretary problems", SIGecom Exch., vol. 7, no. 2, 7:1–7:11, Jun. 2008.

[6] A. Badanidiyuru, B. Mirzasoleiman, A. Karbasi, and A. Krause, "Streaming submodular maximization: Massive data summarization on the fly", ser. KDD '14, New York, New York, USA: ACM, 2014, pp. 671–680.

[7] M. Bateni, M. Hajiaghayi, and M. Zadimoghaddam, "Submodular secretary problem and extensions", ACM Trans. Algorithms, vol. 9, no. 4, 32:1–32:23, Oct. 2013.

[8] N. Buchbinder and M. Feldman, "Submodular functions maximization problems", 2017.

[9] N. Buchbinder, M. Feldman, and M. Garg, "Online submodular maximization: Beating 1/2 made simple", arXiv preprint arXiv:1807.05529, 2018.

[10] N. Buchbinder, M. Feldman, J. S. Naor, and R. Schwartz, "Submodular maximization with cardinality constraints", ser. SODA '14, Portland, Oregon: Society for Industrial and Applied Mathematics, 2014, pp. 1433–1452.

[11] A. Chakrabarti and S. Kale, "Submodular maximization meets streaming: Matchings, matroids, and more", Mathematical Programming, vol. 154, no. 1, pp. 225–247, 2015.

[12] C. Chekuri, S. Gupta, and K. Quanrud, "Streaming algorithms for submodular function maximization", in Automata, Languages, and Programming, M. M. Halldórsson, K. Iwama, N. Kobayashi, and B. Speckmann, Eds., Berlin, Heidelberg: Springer Berlin Heidelberg, 2015, pp. 318–330, ISBN: 978-3-662-47672-7.

[13] N. Devanur and T. Hayes, "The adwords problem: Online keyword matching with budgeted bidders under random permutations", in ACM EC, 2009.

[14] E. B. Dynkin, "The optimum choice of the instant for stopping a Markov process", Soviet Math. Dokl, vol. 4, 1963.

[15] U. Feige, V. S. Mirrokni, and J. Vondrák, "Maximizing non-monotone submodular functions", SIAM J. Comput., vol. 40, no. 4, pp. 1133–1153, Jul. 2011.

[16] J. Feldman, M. Henzinger, N. Korula, V. Mirrokni, and C. Stein, "Online stochastic packing applied to display ad allocation", Algorithms–ESA 2010, pp. 182–194, 2010.

[17] M. Feldman, A. Karbasi, and E. Kazemi, "Do less, get more: Streaming submodular maximization with subsampling", in Advances in Neural Information Processing Systems, 2018, pp. 732–742.

[18] M. Feldman, O. Svensson, and R. Zenklusen, "A simple o (log log (rank))-competitive algorithm for the matroid secretary problem", SIAM, 2014, pp. 1189–1201.

[19] M. Feldman and R. Zenklusen, "The submodular secretary problem goes linear", ser. FOCS '15, Washington, DC, USA: IEEE Computer Society, 2015, pp. 486–505.

[20] V. Feldman and J. Vondrak, "Optimal bounds on approximation of submodular and xos functions by juntas", ser. FOCS '13, Washington, DC, USA: IEEE Computer Society, 2013, pp. 227–236.

[21] T. S. Ferguson et al., "Who solved the secretary problem?", Statistical science, vol. 4, no. 3, pp. 282–289, 1989.

[22] Y. Filmus and J. Ward, "A tight combinatorial algorithm for submodular maximization subject to a matroid constraint", 2012 Symposium on Foundations of Computer Science, 2012.

[23] Z. Friggstad, J. Könemann, Y. Kun-Ko, A. Louis, M. Shadravan, and M. Tulsiani, "Linear programming hierarchies suffice for directed steiner tree", in IPCO, Springer, 2014, pp. 285–296.

[24] Z. Friggstad, J. Könemann, and M. Shadravan, "A Logarithmic Integrality Gap Bound for Directed Steiner Tree in Quasi-bipartite Graphs ", in (SWAT 2016), ser. Leibniz International Proceedings in Informatics (LIPIcs), 2016, 3:1–3:11.

[25] R. Ghuge and V. Nagarajan, "Quasi-polynomial algorithms for submodular tree orienteering and other directed network design problems", in Symposium on Discrete Algorithms, SIAM, pp. 1039–1048.

[26] D. Gijswijt, V. Jost, and M. Queyranne, "Clique partitioning of interval graphs with submodular costs on the cliques", RAIRO-Operations Research, vol. 41, no. 3, pp. 275–287, 2007.

[27] A. Gupta, A. Roth, G. Schoenebeck, and K. Talwar, "Constrained non-monotone submodular maximization: Offline and secretary algorithms", ser. WINE'10, Stanford, CA, USA: Springer-Verlag, 2010, pp. 246–257.

[28] E. Hazan, S. Safra, and O. Schwartz, "On the complexity of approximating k-set packing", computational complexity, vol. 15, no. 1, pp. 20–39, 2006.

[29] T. Hess and S. Sabato, "The submodular secretary problem under a cardinality constraint and with limited resources", CoRR, vol. abs/1702.03989, 2017. eprint: 1702.03989.

[30] B. Kalyanasundaram and K. Pruhs, "Speed is as powerful as clairvoyance", J. ACM, vol. 47, no. 4, pp. 617–643, 2000.

[31] M. Kapralov, I. Post, and J. Vondrák, "Online submodular welfare maximization: Greedy is optimal", ser. SODA '13, New Orleans, Louisiana: Society for Industrial and Applied Mathematics, 2013, pp. 1216–1225.

[32] E. Kazemi, M. Mitrovic, M. Zadimoghaddam, S. Lattanzi, and A. Karbasi, "Submodular streaming in all its glory: Tight approximation, minimum memory and low adaptive complexity", arXiv preprint arXiv:1905.00948, 2019.

[33] D. Kempe, J. Kleinberg, and É. Tardos, "Maximizing the spread of influence through a social network", 2003, pp. 137–146.

[34] T. Kesselheim and A. Tönnis, "Submodular secretary problems: Cardinality, matching, and linear constraints", 2017.

[35] R. Kleinberg, "A multiple-choice secretary algorithm with applications to online auctions", in Proceedings of the Sixteenth Annual ACM-SIAM Symposium on Discrete Algorithms, ser. SODA '05, Vancouver, British Columbia: Society for Industrial and Applied Mathematics, 2005, pp. 630–631.

[36] N. Korula, V. Mirrokni, and M. Zadimoghaddam, "Online submodular welfare maximization: Greedy beats 1/2 in random order", ser. STOC '15, Portland, Oregon, USA: ACM, 2015, pp. 889–898.

[37] O. Lachish, "O(log log rank) competitive-ratio for the matroid secretary problem", CoRR, vol. abs/1403.7343, 2014. arXiv: 1403.7343.

[38] D. V. Lindley, "Dynamic programming and decision theory", vol. 10, no. 1, pp. 39–51, 1961.

[39] A. McGregor and H. T. Vu, "Better streaming algorithms for the maximum coverage problem", in 20th International Conference on Database Theory, 2017.

[40] B. Mirzasoleiman, A. Badanidiyuru, A. Karbasi, J. Vondrák, and A. Krause, "Lazier than lazy greedy", in Proceedings of the Twenty-Ninth Conference on Artificial Intelligence, ser. AAAI'15, Austin, Texas: AAAI Press, 2015, pp. 1812–1818.

[41] M. Moser, D. P. Jokanovic, and N. Shiratori, "An algorithm for the multidimensional multiple-choice knapsack problem", IEICE transactions on fundamentals of electronics, vol. 80, no. 3, pp. 582–589, 1997.

[42] G. L. Nemhauser and L. A. Wolsey, "Best algorithms for approximating the maximum of a submodular set function", Mathematics of operations research, vol. 3, no. 3, pp. 177–188, 1978.

[43] G. L. Nemhauser, L. A. Wolsey, and M. L. Fisher, "An analysis of approximations for maximizing submodular set functions—i", Mathematical programming, vol. 14, no. 1, pp. 265–294, 1978.

[44] A. Norouzi-Fard, J. Tarnawski, S. Mitrovic, A. Zandieh, A. Mousavifar, and O. Svensson, "Beyond 1/2-approximation for submodular maximization on massive data streams", in Proceedings of the 35th International Conference on Machine Learning, vol. 80, PMLR, 2018, pp. 3829–3838.

[45] C. A. Phillips, C. Stein, E. Torng, and J. Wein, "Optimal time-critical scheduling via resource augmentation", Algorithmica, vol. 32, pp. 163–200, 2001.

[46] B. Sankaran, M. Ghazvininejad, X. He, D. Kale, and L. Cohen, "Learning and optimization with submodular functions", arXiv preprint arXiv:1505.01576, 2015.

[47] J. A. Soto, "Matroid secretary problem in the random-assignment model", vol. 42, no. 1, pp. 178–211, 2013.

[48] R. J. Vanderbei, "The optimal choice of a subset of a population", Math of OR, vol. 5, no. 4, pp. 481–486, 1980.

[49] J. 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, ser. STOC '08, Victoria, British Columbia, Canada: ACM, 2008, pp. 67–74.

[50] J. G. Wilson, "Optimal choice and assignment of the best m of n randomly arriving items", Stochastic Processes and their Applications, vol. 39, no. 2, pp. 325 –343, 1991.

[51] J. G. Wilson, "Optimal choice and assignment of the best m of n randomly arriving items", Stochastic processes and their applications, vol. 39, no. 2, pp. 325–343, 1991.