Improved Submodular Secretary Problem with Shortlists

Mohammad Shadravan *

October 6, 2020

Abstract

First, for the submodular $k$-secretary problem with shortlists [1], we provide a near optimal $1 - 1/e - \epsilon$ approximation using shortlist of size $O(k/\epsilon \log(1/\epsilon))$. In particular, we improve the size of shortlist used in [1] from $O(k^{2\text{poly}(1/\epsilon)})$ to $O(k/\epsilon \log(1/\epsilon))$. As a result, we provide a near optimal approximation algorithm for random-order streaming of monotone submodular functions under cardinality constraints, using memory $O(k/\epsilon)$. It exponentially improves the running time and memory of [1] in terms of $1/\epsilon$.

Next we generalize the problem to matroid constraints, which we refer to as submodular matroid secretary problem with shortlists. It is a variant of the matroid secretary problem [13], in which the algorithm is allowed to have a shortlist. The main question is how to achieve a constant competitive algorithm using a shortlist as small as possible. We design an algorithm that achieves a $\frac{1}{2}(1 - 1/e^2 - \epsilon)$ competitive ratio for any constant $\epsilon > 0$, using a shortlist of size $O(\frac{k}{\epsilon} \log(1/\epsilon))$. This is especially surprising considering that the best known competitive ratio for the matroid secretary problem is $O(\log \log k)$ [13, 20], where $k = rk(M)$. Moreover, we generalize our results to the case of $p$-matchoid constraints and give a $\frac{1}{p+1}(1 - 1/e^{p+1} - \epsilon)$ approximation using shortlist of size $O(\frac{k}{\epsilon} \log(1/\epsilon))$. It asymptotically approaches the best known offline guarantee $\frac{1}{p+1}$ [22].

Furthermore, we show that our algorithms can be implemented in the streaming setting using $O(\frac{k}{\epsilon})$ memory. For any constant $\epsilon > 0$, our algorithms achieve a $1 - 1/e - \epsilon$, a $\frac{1}{2}(1 - 1/e^2 - \epsilon)$, and a $\frac{1}{p+1}(1 - 1/e^{p+1} - \epsilon)$ approximation for random-order streaming of submodular functions, under cardinality, matroid, and $p$-matchoid constraints, respectively.

*Yale University, mohammad.shadravan@yale.edu
1 Introduction

In the Secretary problem which is a classical problem, $n$ items arrive in random order. The goal is to select the item with the highest value. Decision is made in an online manner, so once we observe one item we need to irrevocably decide whether or not to select that item. There is a simple strategy to achieve a $1/e$ competitive algorithm for this problem, which is the best possible. Dynkin [10].

Many variants and generalizations of the secretary problem have been studied in the literature, see e.g., [2, 23, 25, 27, 19, 4]. Kleinberg [19], Babaioff et al. [4] introduced a multiple choice secretary problem, where the goal is to select $k$ items in a randomly ordered input so as to maximize the sum of their values; They provide a algorithm that asymptotically approaches the optimal solution. This problem has been further generalized to the case of submodular functions [6, 15], in which the value of the selected items is evaluated by a monotone submodular function. The algorithm can select at most $k$ items $a_1, \ldots, a_k$, in an online manner, from a randomly ordered sequence of $n$ items. The goal is to maximize $f(\{a_1, \ldots, a_k\})$. The algorithm has a value oracle access to the function.

Kesselheim and Tönnis [18], achieve a $1/e$-competitive competitive algorithm for this problem. The offline problem, i.e., the problem of maximizing a monotone submodular function under cardinality constraint is NP-hard, The best approximation algorithm possible is a $1 − 1/e$-approximation algorithm by the greedy algorithm [21]. No online algorithm with the same guarantee is known for this problem. Thus Agrawal et al. [1], introduced a model called shortlist model which is a relaxation of the online model. They study if a near $(1 − 1/e)$ approximation is possible under this new model.

In the shortlist model, the algorithm is allowed the keep a shortlist and upon receiving one new item add it to the shortlist or not. At the end, the output of the algorithm should be a subset of this shortlist. Optimistically, the goal is to keep this shortlist as small as possible, while achieving near optimal guarantees. They provide a $1 − 1/e − \epsilon − O(1/k)$ approximation for this problem using shortlist of size $O(k 2^{poly}(1/\epsilon))$. Although the dependency on $k$ is linear but the dependency on $1/\epsilon$ is exponential. Therefore it is far from being practical.

The shortlist model has connections to another related problem is submodular random order streaming problem studied in [23]. In this problem, items from a set $U$ arrive online in random order and the algorithm aims to select a subset $S \subseteq U$, $|S| \leq k$ 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. An algorithm in one model can not directly be converted to an algorithm in the other model. However Agrawal et al. [1] show that their algorithms, can be implemented to use the same $\eta(k) = O(k 2^{poly}(1/\epsilon))$ memory buffer for the random order streaming model.

Recently streaming algorithms for maximizing a submodular function has been studied in a series of work. Badanidiyuru et al. [3], provide the first one-pass streaming algorithm for maximizing a monotone submodular function subject to a $k$-cardinality constraint. They achieve $(1/2 − \epsilon)$-approximation streaming algorithm, with a memory of size $O(\frac{1}{\epsilon} k \log k)$. Recently, Kazemi et al. [17] improved the memory buffer to $O(k/\epsilon)$.

Norouzi-Fard et al. [23] show that under some natural assumption no $1/2 + o(1)$ approximation ratio can be achieved any algorithm for streaming submodular maximization using $o(n)$ memory.

They studied the random order streaming model in order to go beyond the upperbound for the adversarial order inputs. They achieve $1/2 + 8 \times 10^{-14}$ approximation using a memory buffer of size $O(k \log k)$. Agrawal et al. [1] substantially improve their result to $1 − 1/e − \epsilon − O(1/k)$, by showing that their algorithm for the shortlist model can be converted into a random order streaming
model. Furthermore, they improve the required memory buffer (in terms of $k$) to only $O_\epsilon(k)$. But one disadvantage of their algorithm is that their dependency on $1/\epsilon$ is exponential. In this paper, we improve their algorithm and give a near optimal algorithm using memory $O(k\text{poly}(1/\epsilon))$.

In addition to the simple cardinality constraint, more general constraints have been studied in the literature. Chekuri et al. [9] give a $1/4p$ approximation algorithm for streaming monotone submodular functions maximization subject to $p$-matchoid constraints. $p$-matchoid constraints generalize many basic combinatorial constraints such as the cardinality constraint, the intersection of $p$ matroids, and matchings in graphs and hyper-graphs. Recently, Feldman et al. [14] designed a more efficient algorithm with lower number of function evaluations achieving the same approximation $1/4p$. We show that our algorithms can be implemented in the streaming setting using $O(k\text{poly}(1/\epsilon))$ memory. For any constant $\epsilon > 0$, our algorithms achieve a $1 - 1/e - \epsilon - O(1/k)$ approximation for random-order streaming of submodular functions, under cardinality, matroid, and $p$-matchoid constraints, respectively.

Furthermore, the greedy algorithm yields a ratio of $1/(p + 1)$ for $p$-independent systems [22]. These ratios for greedy are tight for all $p$ [16]. Therefore our results for $p$-matchoid constraints is asymptotically tight.

### The shortlist model.

In [1], a relaxation of the secretary problem is introduced where the algorithm is allowed to select a shortlist of items. After seeing the entire input, the algorithm can choose from the bigger set of items in the shortlist. This model is closely related to the random order streaming model. A comprehensive comparison between these two models can be found in [1]. The main result of [1] is an online algorithm for submodular $k$-secretary problem with shortlists that, for any constant $\epsilon > 0$, achieves a competitive ratio of $1 - 1/e - \epsilon - O(1/k)$ with shortlist of size $O(k)$.

#### 1.1 Problem definition

We are given matroid $\mathcal{M} = (\mathcal{N}, \mathcal{I})$, with $rk(\mathcal{M}) = k$. Items from a set $\mathcal{U} = \{a_1, a_2, \ldots, a_n\}$ 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

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

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

$$\text{OPT} := f(S^*)$$

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 $A^* \subseteq A$, and $A^* \in \mathcal{I}$.

The problem definition for $p$-matchoid constraint is similar, but $S$ needs to be an independent set in all the matroids $\mathcal{M}_i$, for $i \in [q]$. 

3
1.2 Our Results

**Theorem 1.** For any constant $\epsilon > 0$, there exists an online algorithm (Algorithm 1) for the submodular $k$-secretary problem with shortlists that achieves a competitive ratio of $1 - \frac{1}{e} - \epsilon$, with shortlist of size $\eta_k(k)$. The running time of this algorithm is $O_\epsilon(n)$.

Throughout the paper $\eta_k(k) = O(k\text{poly}(1/\epsilon))$ and the hidden constant in $O(.)$ notation is $O(\text{poly}(1/\epsilon))$. This is an exponential speed-up of the results provided in the previous work [1].

**Theorem 3.** For any constant $\epsilon > 0$, there exists an online algorithm (Algorithm 4) for the submodular matroid secretary problem with shortlists that achieves a competitive ratio of $\frac{1}{2}(1 - \frac{1}{e} - \epsilon)$, with shortlist of size $\eta_k(k)$. The running time of this algorithm is $O_\epsilon(nk)$.

This result is especially surprising considering that the best known competitive ratio for the matroid secretary problem is $\Omega(1/\log \log k)$. It implies a constant competitive algorithm using shortlist of size at most $k$ and also a constant competitive algorithm in the preemption model.

Furthermore, for a more general constraint, namely $p$-matchoid constraints

**Theorem 5.** 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 $\frac{1}{p+1}(1 - \frac{1}{e^{p+1}} - \epsilon)$, with shortlist of size $\eta_k(k)$. The running time of this online algorithm is $O(n\kappa^p)$, where $\kappa = \max_{i \in [p]} r(M_i)$.

The proposed algorithm also has implications for another important problem of submodular function maximization under random order streaming model. Furthermore, the algorithm can be readily parallelized among multiple (as many as processors).

**Theorem 2.** For any constant $\epsilon \in (0, 1)$, there exists an algorithm for the submodular random order streaming problem with matroid constraints that achieves $1 - \frac{1}{e} - \epsilon$ approximation algorithm, while using a memory buffer of size at most $\eta_k(k) = O_\epsilon(k)$.

**Theorem 4.** For any constant $\epsilon \in (0, 1)$, there exists an algorithm for the submodular random order streaming problem with matroid constraints that achieves $\frac{1}{2}(1 - \frac{1}{e} - \epsilon)$ approximation algorithm, while using a memory buffer of size at most $\eta_k(k) = O_\epsilon(k)$.

**Theorem 6.** For any constant $\epsilon > 0$, there exists an algorithm for the submodular random order streaming problem with $p$-matchoid constraints that achieves $\frac{1}{p+1}(1 - \frac{1}{e^{p+1}} - \epsilon)$ approximation, while using a memory buffer of size at most $\eta_k(k) = O_\epsilon(k)$.

1.3 Related Work

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 welfare maximizing online mechanism design for domains in which the sets of simultaneously satisfiable agents form a matroid [8].

The existence of a constant competitive algorithm is a long-standing open problem. It has been shown that for some special cases of the matroid secretary problem, $O(1)$-competitive algorithms exists. But for general case the problem is still open. Lachish [20] provides the first $\Omega(1/\log \log (k))$-competitive algorithm (the hidden constant is $2^{2^{34}}$). Feldman et al. [13] give a simpler order-oblivious $1/(2560(\log \log (4k) + 5))$-competitive algorithm. For the preemption model, which is relaxation of the online model that we can substitute one item, Buchbinder et al. [8] present a randomized $0.0893$-competitive algorithm for cardinality constraints using $O(k)$ memory.
\section{Preliminaries}

The following properties of submodular functions are well known (e.g., see \cite{8,11,12}).

\textbf{Definition 1.} Given a monotone submodular function $f$, and subsets $A, B$ in the domain of $f$, we use $\Delta_f(A|B)$ to denote $f(A \cup B) - f(B)$.

\textbf{Lemma 1.} 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 $B$, $\Delta_f(A|B) \leq \sum_{a \in A \setminus B} \Delta_f(a|B)$

\textbf{Lemma 2.} Denote by $A(p)$ a random subset of $A$ where each element has probability at least $p$ to appear in $A$ (not necessarily independently). Then $E[f(A(p))] \geq (1-p)f(\emptyset) + (p)f(A)$

\textbf{Lemma 3.} (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 $\mu = E(X) = \sum_{i=1}^{N} p_i$. 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 $\delta \in (0,1)$.

\textbf{Definition 2.} (\textit{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) $\emptyset \in \mathcal{I}$, (ii) If $A \subseteq B \subseteq \mathcal{N}$, and $B \in \mathcal{I}$, then $A \in \mathcal{I}$, (iii) If $A, B \in \mathcal{I}$ and $|A| < |B|$, then there is an element $b \in B \setminus A$ such that $A + b \in \mathcal{I}$. In a matroid $\mathcal{M} = (\mathcal{N}, \mathcal{I})$, $\mathcal{N}$ 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 $\mathcal{M}$ (denote it by $rk(\mathcal{M})$).

\textbf{Definition 3.} (\textit{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 at most $p$ matroids.

\textbf{Lemma 4.} For any matroid $\mathcal{M}$, with $rk(\mathcal{M}) = k$. Every independent set $I \in \mathcal{I}$, with $|I| < k$ can be extended to a base $I' \supset I$, with $|I'| = k$.

\textbf{Lemma 5.} (\textit{Brualdi \cite{12}}) If $A, B$ are any two bases of matroid $M$ then there exists a bijection $\pi$ from $A$ to $B$, fixing $A \cap B$, such that $A - x + \pi(x) \in M$ for all $x \in A$.

In \cite{11}, a $(n,m)$-ball-bin random set is defined as follows. 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 $j$-th bin. They call the resulting $X_j$’s a $(n,m)$-ball-bin random set. They use these variables to define $\alpha, \beta$-windows as follows.

\textbf{Definition 4} ((\alpha,\beta) windows \cite{11}). 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 $k/\alpha$ windows, where window $i$ consists of $\alpha \beta$ consecutive slots, in the same manner as we assigned slots.

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.

Additionally, 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 $w$ that appear after slot $s'$.
3 Cardinality Constraints

In this section, we focus on the cardinality constraints, namely submodular $k$-secretary problem with shortlists. Agrawal et al. [1], provide a near optimal approximation algorithms for this problem using shortlist of size $O_\epsilon(k)$, where the hidden constant is $O(2^{\text{poly}(1/\epsilon)})$. Although the running time of their algorithm is linear in $n$, but the large hidden constant that exponentially depends on $1/\epsilon$ makes this algorithm far from practical. In this section we propose a new algorithm that improves the dependency on $1/\epsilon$. We achieve an improved approximation ratio $1 - 1/\epsilon - \epsilon$ using shortlist of size $O_\epsilon(k)$. But the the dependency of our algorithm on $1/\epsilon$ is $O(\frac{1}{\epsilon} \log(1/\epsilon))$.

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

We make some changes to the algorithm and analysis of Agrawal et al. [1]. The main modification is in the way the algorithm selects elements inside a window. Similar to [1], the building block of the algorithm are $(\alpha, \beta)$-windows defined in [1]. But the algorithm does not need to choose the best $\alpha$-subsequence $\tau^*$, and return the $\gamma(\tau^*)$ defined on that subsequence among $\left(\frac{\alpha \beta}{\alpha}\right)$ many subsequences. This number of selections in a window is the reason for having a hidden constant in the $O_\epsilon(k)$ that exponentially depends on $1/\epsilon$. We alleviate the selection method in a window by keeping track of $\alpha$ subsets. We reduce the total number of selected items in a window to $\alpha^2 \beta \log(1/\epsilon)$ and totally to $k\alpha\beta \log(1/\epsilon)$ (In the last section we even further improve this bound to make it independent of $\alpha$).

3.1 Algorithm description (cardinality constraint)

The algorithm divides the input into $(\alpha, \beta)$-windows, $1, \cdots, W$. The output of the algorithm is $S$. It is initially an empty set, and it will be incremented by adding $\alpha$ items in each window. Let’s denote by $S_1, \cdots, w$, the set $S$ by the end of window $w$. Additionally, the algorithm keeps track of a subset of selected items $R_i$, that we call it shortlist. It is initially an empty set, and it grows at the end of each window by adding a selected subset of that window to it. The shortlist is the set of items that might be selected later on by the algorithm and be added to the set $S$. Any other item that is not selected as part of the shortlist will be discarded immediately. Let’s denote by $R_1, \cdots, w$, the shortlist defined on the first $w$ windows. Throughout the paper, if the subscript of $S$ and $R$ is not stated explicitly, we mean $S_1, \cdots, w-1$ and $R_1, \cdots, w-1$ respectively.

In each window $w$, the algorithm keeps track of $\alpha$ subsets $H_1, \cdots, H_\alpha$, with $H_i$ is either $\emptyset$ or $|H_i| = i$. In each slot $s$ in window $w$, the algorithm tends to select $\alpha$ elements $m_i$, where each $m_i$ corresponds to set $H_i$. The element $m_i$ is the element with maximum marginal gain with respect to $S \cup H_i$.

$$m_i \leftarrow \arg \max_{x \in s_j \cup R} \Delta(x | S \cup H_i)$$  \hspace{1cm} (1)

Each maximum element can be found in an online manner by the online max algorithm (Algorithm 1 in [1]) using shortlist of size $O(\log(1/\epsilon))$. At the end of slot $s$, we add $\{m_i\}^{\alpha}_{i=1}$ to the shortlist $R$. Moreover, if $\Delta(H_{i+1}|S) < \Delta(H_i + m_i | S)$, we update

$$H_{i+1} \leftarrow H_i + m_i$$  \hspace{1cm} (2)
Algorithm 1  Cardinality-Constraint

1: Inputs: submodular function $f$, parameter $\epsilon \in (0, 1]$, and set $S$ and $R$.
2: Initialize $H_\ell \leftarrow \emptyset, \forall 0 \leq \ell \leq \alpha$
3: for every slot $s$ in window $w$ do
4:   for $1 \leq \ell \leq \alpha$, do
5:     $R' \leftarrow \text{Sample}(R, 1/(k\beta))$ \{sample a set of size $|R|/(k\beta)$ from $R$\}
6:     call the online max algorithm (Algorithm 1 in [1]) to compute, with probability $\epsilon/2$:
7:     $m_\ell \leftarrow \arg \max_{x \in s \cup R'} \Delta(x|S \cup H_{\ell-1})$.
8:     $M_\ell \leftarrow \text{The shortlist returned by the above online max algorithm for slot } s$ and set $H_{\ell-1}$.
9:     if $\Delta(H_\ell|S) < \Delta(H_{\ell-1} + m_\ell|S)$ then
10:        $H_\ell \leftarrow H_{\ell-1} + m_\ell$
11:     end if
12: end for
13: end for
14: return $H_\alpha$

Algorithm 2  Submodular Secretary with Shortlists

1: Inputs: number of items $n$, submodular function $f$, parameter $\epsilon \in (0, 1]$. 
2: Initialize: $S \leftarrow \emptyset$, $R \leftarrow \emptyset$, constants $\alpha \geq 1$, $\beta \geq 1$ which depend on the constant $\epsilon$.
3: Divide indices $\{1, \ldots, n\}$ into $(\alpha, \beta)$ windows.
4: for window $w = 1, \ldots, W = k/\alpha$ do
5:     $S_w \leftarrow \text{Cardinality-Constraint}(S, w, R)$
6:     $S \leftarrow S \cup S_w$
7: end for
8: return $S \cap R$

3.2 Analysis of the algorithm: cardinality constraint

In this section, we prove Theorem 1. First of all, we can bound the size of shortlist:

Lemma 6. The size of the shortlist $R$ that Algorithm 2 uses is at most $4k\alpha\beta\log(2/\epsilon)$.

Proof. There are total of $\alpha\beta(k/\alpha) = k\beta$ slots. In each slot, we run $\alpha$ online max algorithms, each add elements of $M_i$ with size $4 \log(2/\epsilon)$ to the shortlist $R$. Thus, the algorithm add $(k\beta)(4 \log(2/\epsilon)\alpha)$ items to the shortlist $R$. \hfill $\square$

Now we give an overall overview of the analysis of the algorithm.

Proof overview. We first lower bound $\mathbb{E}[f(S_1, \ldots, W)]$, and then we lower bound $\mathbb{E}[f(S \cap R)]$. In particular we prove competitive ratio $1 - 1/e - \epsilon$ for Algorithm 2 by choosing large enough parameters $\alpha, \beta$ that are depending on $1/\epsilon$. Similar to [1], a crucial idea is to show that given the history of the selection made by the algorithm in windows $1, \ldots, w - 1$, the probability that any of the $k$ items in the optimal solution $S^*$ appears either in $w$ or in the shortlist $R$ is at least $\frac{\alpha}{k}$. Additionally, the elements of $S^*$ are distributed independently and uniformly at random in the $\alpha\beta$ slots of $w$. Since we have modified the algorithm, the structure of the elements that get selected in windows $1, \ldots, w - 1$ is different from the structure of selected elements in Algorithm 2 in [1],
namely \( T_{1,\ldots,w-1} \) (refer to Definition 3 in \([1]\)). But still we are able to prove the aforementioned property. The main reason is that under new selection criteria in our Algorithm \([2]\) removing one item that is not selected by the algorithm would not change the output of the \( \arg\max \) in line 5 of Algorithm \([2]\). Hence if we remove one of the items not selected by the algorithm, all the subsets \( H_i \) in a window remain unchanged and consequently \( S_w \) and \( S \) remain unchanged. Therefore, we can still prove similar properties proven for \((\alpha,\beta)\)-windows in \([1]\).

By choosing \( \alpha \) and \( \beta \) large enough, it is easy to see that there are roughly \( \alpha \) elements of \( S^* \) in a window \( w \), and they appear in different slots w.h.p. The algorithm in \([1]\), tries to identify the slots in \( w \) containing elements of \( S^* \). In particular, it is looking for an \( \alpha \)-subsequence of slots \( \tilde{\tau}_w \) containing elements of \( S^* \cap w \). Because the algorithm is not aware of the optimal solution and the fact that elements arrive in an online manner, it is not possible to predict those slots. The idea in \([1]\), is to try all \( \alpha \)-subsequences \( \tau \) of slots in \( w \) and choose the one with highest marginal gain (tries as many as \( \binom{n+\beta}{\alpha} \), subsequences). Moreover, knowing the slots in \( w \) containing the \( S^* \) is not enough to know which element in those slots are part of \( S^* \). Thus in each slot they choose the element with the highest marginal gain with respect to the elements selected in the previous slots (refer to the definition of \( \gamma \) function in \([1]\), eq. (1)). Consequently, they argue that the marginal gain of the selected item in each slot of \( \tilde{\tau}_w \) is more than the marginal gain of the element of \( S^* \) in the same slot.

Here, in this paper we argue that it is not necessary to consider all \( \alpha \)-subsequences in window \( w \). Consider \( \alpha \)-subsequence \( \tilde{\tau}_w = \{s_1,\ldots,s_t\} \), containing \( S^* \cap w \). The algorithm in \([1]\) greedily chooses the set \( \gamma(\tilde{\tau}_w) = \{i_1,\ldots,i_t\} \) containing one element from each slot \( i_j \). Then, they lower bound the marginal gain of \( i_j \) with respect to previously selected elements \( S \cup \{i_1,\ldots,i_{j-1}\} \). More precisely, using the aforementioned crucial property stated above, in their Lemma 11, they show that for all \( j = 1,\ldots,t \),

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

(3)

Now consider subsequence \( \tilde{\tau}_w = \{s_1,\ldots,s_t\} \). Modify the method in \([1]\) for selecting corresponding elements in this subsequence, namely \( \gamma(\tau) \) as follows. Suppose that after selecting \( i_j \) in slot \( s_j \), the algorithm substitutes the current \( j \) elements selected so far from slots \( s_1,\ldots,s_j \) with a subset \( C_j \) of \( w \), whose marginal gain w.r.t. \( S \) is larger (the elements of \( C_j \) do not need to be from slots in \( \tilde{\tau}_w \), they should be from the shortlist). Thus, for all \( j = 1,\ldots,t \),

\[
\Delta(C_j|S) \geq \Delta(C_{j-1} + i_j|S).
\]

(4)

where

\[
i_j := \arg\max_{x \in s_j \cup R} \Delta(x|S \cup C_{j-1}).
\]

Given that the subset \( C_j \) selected by the algorithm is part of the shortlist, and conditional on the history of the selection made by algorithm, i.e., \( T \), we can prove an equation similar to eq. (3) for subsets \( C_j \).

\[
\mathbb{E}[\Delta_f(i_j|S \cup C_{j-1})|T] \geq \frac{1}{k} (f(S^*) - f(S \cup C_{j-1})).
\]

Consequently, we can show

\[
\mathbb{E}[f(S^*) - f(S_{1,\ldots,w-1} \cup C_j)|T] \geq \frac{1}{e} (f(S^*) - f(S_{1,\ldots,w-1} \cup C_{j-1})).
\]

(5)

\[
\mathbb{E}[f(S^*) - f(S_{1,\ldots,w-1} \cup C_j)|T] \leq \mathbb{E}[f(S^*) - f(S_{1,\ldots,w-1} \cup C_{j-1} \cup \{i_j\})|T]
\]

(6)

\[
\leq \frac{1}{e} (f(S^*) - f(S_{1,\ldots,w-1} \cup C_{j-1})).
\]

(7)
Now the question is how should the algorithm create such sets $C_j$ in an online manner. Note that the algorithm is not aware of $\tilde{\tau}_w$, the slots containing $S^* \cap w$. In order to do that, the algorithm keeps track of $\alpha$ sets $H_1, \ldots, H_\alpha$. One main observation is the following:

**Lemma 7.** Suppose $\tilde{\tau}_w$ is the subsequence containing $S^* \cap w$. For $0 \leq j < \alpha$, define $C_j$ to be the latest set $H_j$ before observing slot $s_{j+1}$, and $C_\alpha$ to be the set $H_\alpha$ at the end of window $w$. Then, $\{C_j\}_{j=1}^\alpha$ satisfy equation (4) and therefore (5).

The intuition behind the proof is inductive. Suppose at the time the algorithm arrives in the slot $s_j$, we know $H_{j-1}$ satisfies eq. (4) and we set $C_{j-1} = H_{j-1}$. Then the algorithm find the element with maximum marginal gain w.r.t. $S \cup H_j$, namely $i_j$. Now from line 8 of the algorithm, either

$$\Delta(H_j|S) \geq \Delta(H_{j-1} + i_j|S)$$

or we update $H_j = H_{j-1} + i_j$. In either case and later on in the algorithm we will have

$$\Delta(H_j|S) \geq \Delta(C_{j-1} + i_j|S)$$

Thus this property holds true for the latest $H_j$ before observing slot $s_{j+1}$. Thus by setting $C_j = H_j$, before observing $s_{j+1}$, the eq. (4) holds true for $C_{j+1}$ too. Note that the algorithm is not aware of the position of slots $s_j$, and therefore the sets $\{C_j\}_{j=1}^\alpha$. That is why the algorithm keeps track of all $\alpha$ subsets $H_i$, each corresponding to subsets of size $i = 1, \ldots, \alpha$.

### 3.3 Some useful properties of $(\alpha, \beta)$ windows

We revisit the properties proven for $(\alpha, \beta)$-windows in [1]. Because of some changes made in the algorithm we need to provide new proofs for some of these properties.

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

**Definition 5.** 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 8.** 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 $(\alpha, \beta)$ windows. The proof can be found in [1].

**Definition 6.** Let’s denote by $H^s_\ell$, the set $H_\ell$ as defined in the Algorithm at the end of slot $s$ in window $w$.

Next, we make important observations about the probability of assignment of items in $S^*$ in the slots in a window $w$, given the sets $R_{1, \ldots, w-1}, S_{1, \ldots, w-1}$. For the purpose of analysis, we define the following new random variable $T_w$ that will track all the useful information from a window $w$.

**Definition 7.** For window $w \in [W]$, define

$$T_w := \{H^s_\ell | s \in w, 1 \leq \ell \leq \alpha\}, \quad (8)$$


moreover,

\[ T_{1, \ldots, w} := \bigcup_{i=1}^{w} T_i, \]

and

\[ T(w, s) := T_{1, \ldots, w-1} \cup \{ H^s_{\ell} | s \succ_w s', 1 \leq \ell \leq \alpha \}. \]  

(9)

We denote by \( s_0 \) the first slot in window \( w \). Note that \( T(w, s_0) = T_{1, \ldots, w-1} \).

Definition 8. For slot \( s \) in window \( w \) define

\[ \text{Supp}(T_w) := \bigcup_{1 \leq \ell \leq \alpha, s \in w} H^s_{\ell}, \]

also,

\[ \text{Supp}(T_{1, \ldots, w}) := \bigcup_{i=1}^{w} \text{Supp}(T_i), \]

and,

\[ \text{Supp}(T(w, s)) := \text{Supp}(T_{1, \ldots, w-1}) \cup \bigcup_{1 \leq \ell \leq \alpha, s \succ_w s'} H^s_{\ell}. \]

(Note that \( \text{Supp}(T_{1, \ldots, w}) = R_{1, \ldots, w} \)). Furthermore, define \( R(w, s) := \text{Supp}(T(w, s)) \).

Proposition 1. For slots \( s, s' \) in window \( w \), such that \( s \succ_w s' \), we have \( T(w, s') \subseteq T(w, s) \).

Lemma 9. For any window \( w \in [W] \), and slot \( s \) in \( w \), \( T_{1, \ldots, w}, w \) 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 line 5 and 8 of Algorithm \( \square \) that \( T_{1, \ldots, w}, T(w, s) \) and \( S_{1, \ldots, w} \) are independent of the ordering of items within a slot. Since each arg max in line 5 is independent of ordering elements in a slot. Now, since the assignment of items to slots are determined by the configuration \( Y \), we obtain the desired lemma statement. \( \square \)

Following the above lemma, given a configuration \( Y \), we will sometimes use the notation \( T_{1, \ldots, w}(Y), S_{1, \ldots, w}(Y), T(w, s)(Y) \) and \( H^s_{\ell}(Y) \) to make this mapping explicit.

Lemma 10. For any item \( i \in S^* \), window \( w \in \{1, \ldots, W\} \), and slot \( s \) in window \( w \), define

\[ p_{is} := \mathbb{P}(i \in s \cup \text{Supp}(T(w, s)) | T(w, s)). \]

Then,

\[ p_{is} \geq \frac{1}{k \beta}. \]  

(10)

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

We show that for any slot \( s' \), where \( s' \) appears before slot \( s \), i.e., \( s \succ s' \),

\[ \mathbb{P}(T(w, s) = T | Y_i = s') \leq \mathbb{P}(T(w, s) = T | Y_i = s). \]  

(12)
And, for any pair of slots $s', s''$ on or after slot $s$, i.e., $s' \geq s$ and $s'' \geq s$,

$$\mathbb{P}(T(w, s) = T|Y_i = s') = \mathbb{P}(T(w, s) = T|Y_i = s'').$$  \hspace{1cm} (13)

To see (12), suppose for a configuration $Y$ we have $Y_i = s'$ and $T(w, s)(Y) = T$. Since $i \notin \text{Supp}(T(w, s))$, then by definition of $T(w, s)$, we have that $i \notin H^w_{\ell}$ for slot $s'$ and any index $1 \leq \ell \leq \alpha$.

Therefore, if we remove $i$ from slots before slot $s$, i.e., $\text{Supp}(T(w, s))$ (i.e., consider another configuration where $Y_i$ is in slot $s$ or after $s$, i.e. in $\{s': s' \geq s\}$), then $T(w, s)$ would not change. This is because either $i$ is not the output of $\arg \max$ in the definition of $H^w_{\ell}$ (refer to (1), (2)) for slot $s$ and $1 \leq \ell \leq \alpha$, and therefore its removal will not change the output of $\arg \max$ and $H_{\ell}$; or $i$ is the output of $\arg \max$ for slot $s$, and some index $1 \leq \ell \leq \alpha$, but $\Delta(H_{\ell}|S) \geq \Delta(H_{\ell-1} + i|S)$. In that case, removing $i$ will not change $H_{\ell}$ either. Hence, removing $i$ will not change $T(w, s)$.

Also by adding $i$ to slot $s$, $T(w, s)$ will not change (since $T(w, s)$ does not cover $s$) Suppose configuration $Y'$ is a new configuration obtained from $Y$ by changing $Y_i$ from $s'$ to $s$. Therefore $T(w, s)(Y') = T$.

Also remember that from Lemma 8. This mapping shows that $\mathbb{P}(T(w, s) = T|Y_i = s') \leq \mathbb{P}(T(w, s) = T|Y_i = s)$. The proof for (13) is similar. The rest of the proof is by applying Bayes rule and it is similar to Lemma 7 in [1].

\begin{lemma}
For any window $w$, $i, j \in S^*$, $i \neq j$ and $s \in w$, the random variables $1(Y_i = s|T(w, s))$ and $1(Y_j = s|T(w, s))$ are independent. That is, given $T(w, s)$, items $i, j \in S^*$, $i \neq j$ appear in slot $s$ in $w$ independently.
\end{lemma}

\begin{proof}
Proof is similar to Lemma 8 in [1] and it is based on the next Lemma.
\end{proof}

\begin{lemma}
Fix a slot $s'$, $T$, and $j \notin \text{Supp}(T)$. Suppose that there exists some configuration $Y'$ such that $T(w, s')(Y') = T$ and $Y_j' = s'$. Then, given any configuration $Y''$ with $T(w, s')(Y'') = T$, we can replace $Y_j''$ with $s'$ to obtain a new configuration $Y'$ that also satisfies $T(w, s')(Y') = T$.
\end{lemma}

\begin{proof}
Suppose the slot $s'$ lies in window $w'$. If $s' \geq s$ then the statement is trivial. So suppose $s \succ s'$. Create an intermediate configuration by removing the item $j$ from $Y''$, call it $Y'$. Since $j \notin \text{Supp}(T(w, s')(Y'')) = \text{Supp}(T)$ we have $T(w, s')(Y') = T$. In fact, for every slot and $1 \leq \ell \leq \alpha$, the set $H^w_{\ell}$ for $Y''$ will be the same as that for $Y'$, i.e., $H^w_{\ell}(Y'') = H^w_{\ell}(Y')$. Now add item $j$ to slot $s'$ in $Y'$, to obtain configuration $Y'$. We claim $T(w, s')(Y') = T$. By construction of $T_1, \ldots, w$, we only need to show that $j$ will not be in $H^w_{\ell}(Y)$ for slot $s'$ and any $1 \leq \ell \leq \alpha$.

To prove by contradiction, suppose that $j \in H^w_{\ell}(Y)$, for some $1 \leq \ell \leq \alpha$.

Note that since the slots before $s'$ are the same for $Y'$ and $Y''$, we have

$$H^{s'}_{\ell-1}(Y') = H^{s'}_{\ell-1}(Y'') = H^{s'}_{\ell-1}(Y'''),$$

and

$$H^s_{\ell-1}(Y') = H^s_{\ell-1}(Y''') = H^s_{\ell-1}(Y').$$

Suppose $j$ gets selected in slot $s'$ for some index $1 \leq \ell \leq \alpha$, i.e. $j \in H^s_{\ell}(Y')$. Thus,

$$j = \arg \max_{x \in s' \cup R(w', s')} \Delta(x|S_1, \ldots, w'-1 \cup H^{s'}_{\ell-1}(Y))).$$

11
\[
\Delta(H_{t-1}^{s-1}(Y) + j|S_1,\ldots,w') > \Delta(H_{t-1}^{s-1}(Y)|S_1,\ldots,w').
\]

Hence,
\[
\argmax_{x \in S \cup R(w', s')} \Delta(x|S_1,\ldots,w' \cup H_{t-1}^{s-1}(Y')),
\]
and
\[
\Delta(H_{t-1}^{s-1}(Y') + j|S_1,\ldots,w' - 1) > \Delta(H_{t-1}^{s-1}(Y')|S_1,\ldots,w' - 1).
\]

Thus \( j \in H_{t}^{s}(Y') \). In other words \( j \in \text{Supp}(T) \) which is a contradiction. \( \Box \)

### 3.4 Bounding \( \mathbb{E}[f(S_1,\ldots,W)] \)

In this section, we use the observations from the previous sections to lower bound the increment \( \Delta_f(S_w|S_1,\ldots,w-1) = f(S_1,\ldots,w-1 \cup S_w) - f(S_1,\ldots,w-1) \) in every window.

First we create a random subsequence \( \tilde{\tau}_w \) of slots in window \( w \) as follows:

**Definition 9** (\( Z_s \) and \( \tilde{\tau}_w \)). For every slot \( s \) create set \( Z_s \subseteq S^* \) as follows: add every item from \( i \in S^* \cap s \) independently with probability \( \frac{1}{k \beta} \) to \( Z_s \) (where \( p_{i,a} \) is defined in eq. 17). Then, for every item \( i \in S^* \cap \text{Supp}(T(w,s)) \), with probability \( \frac{1}{k \beta} \), add \( i \) to \( Z_s \). Furthermore, define subsequence \( \tilde{\tau}_w \) as the sequence of slots in a window \( w \) with \( Z_s \neq \emptyset \).

**Lemma 13.** For any slot \( s \) in a window \( w \in [W] \), given \( T(w,s) \), all \( i, i' \in S^*, i \neq i' \) will appear in \( Z_s \) independently with probability \( \frac{1}{k \beta} \), i.e., the random variables \( 1(i \in Z_s|T(w,s)) \) are i.i.d. for all \( i \in S^* \), and
\[
\Pr(i \in Z_s|T(w,s)) = \Pr(i' \in Z_s|T(w,s)) = \frac{1}{k \beta}.
\]

**Proof.** The proof is similar to Lemma 10 in [1], and it is based on Lemma 11 which is for the new construction of \( T \) in this paper. \( \Box \)

**Lemma 14.** For \( i, i' \in S^* \), and slot \( s \) in \( w \),
\[
\Pr(i \in Z_s|T(w,s), Z_s \neq \emptyset) = \Pr(i' \in Z_s|T(w,s), Z_s \neq \emptyset) \geq \frac{1}{k}.
\]

**Proof.** The proof is similar to eq. (10) in Lemma 11 of [1], and it is based on Lemma 13, which also holds for the new construction of \( T \). \( \Box \)

**Definition 10.** Define \( m^s_t \) to be \( m_t \) as defined in Algorithm 7, for slot \( s \), which is
\[
m^s_t := \argmax_{x \in S \cup R(w, s)} \Delta(x \cup H_{t-1}^{s-1}),
\]
Also for the sequence \( \tilde{\tau}_w = (s_1,\ldots,s_t) \) defined in Definition 3, define sequence \( \mu_w = (i_1,\ldots,i_{\alpha'}) \), for \( \alpha' = \min(t, \alpha) \), where
\[
i_{j} := m^s_{j},
\]
Moreover, for \( 1 \leq j \leq \alpha' \) define
\[
C_j := H_j^{(s, j+1)} - 1,
\]
if \( j + 1 > t \), set \( s_{j + 1} := \alpha' + 1 \).

We also use the notation \( i_1, \ldots, j = (i_1, \ldots, i_j) \), for \( 1 \leq j \leq \alpha' \).

The following observations are immediate:

**Proposition 2.** For slot \( s \in w \), and \( 1 \leq \ell \leq \alpha \),
\[
\Delta(H_{\ell}^s|S) \geq \Delta(H_{\ell-1}^{s-1} + m_\ell^s|S)
\]

\[
\Delta(H_{\ell}^s|S) \geq \Delta(H_{\ell-1}^{s}|S),
\]

\[
\Delta(H_{\ell}^{s+1}|S) \geq \Delta(H_{\ell}^{s}|S),
\]

also
\[
\Delta(C_{\ell}|S) \geq \Delta(C_{\ell-1}|S),
\]

moreover,
\[
\Delta(S_w|S) \geq \Delta(C_\alpha|S).
\]

**Lemma 15.** For slot \( s \) in window \( w \), and \( 1 \leq \ell \leq \alpha' = \min(t, \alpha) \),
\[
\mathbb{E}_{Z_s}[\Delta(m_\ell^s|S_1, \ldots, w_{-1} \cup H_{\ell-1}^{s-1})|T(w, s), Z_s \neq \emptyset] \geq \frac{1}{k} (f(S^*) - f(S_{1, \ldots, w_{-1} \cup H_{\ell-1}^{s-1}})) .
\]

**Proof.** From Definition 10 \( m_\ell^s \) is chosen greedily to maximize the increment
\[
\arg\max_{x \in s \cup R(w, s)} \Delta(x|S \cup H_{\ell-1}^{s-1}),
\]

So \( m_\ell^s \) belongs to \( s \cup R(w, s) \supseteq Z_s \). Therefore, we can lower bound the marginal gain of \( m_\ell^s \) w.r.t. previously selected items \( S_1, \ldots, w_{-1} \cup H_{\ell-1}^{s-1} \) by the marginal gain of a randomly picked item \( i \) from \( Z_s \) as follows.
\[
\mathbb{E}_{Z_s}[\Delta(m_\ell^s|S \cup H_{\ell-1}^{s-1})|T(w, s), Z_s \neq \emptyset] \geq \frac{1}{k} \sum_{i \in S^*} \Delta(i|S \cup H_{\ell-1}^{s-1})
\]

(using Lemma 11, monotonicity of \( f \)) \[ \geq \frac{1}{k} (f(S^*) - f(S \cup H_{\ell-1}^{s-1})) \]

\[ \square \]

**Corollary 1.** For a window \( w \), suppose the sequence \( \bar{\tau}_w = (s_1, \ldots, s_t) \) is as defined in Definition 4 and \( \mu_w = \{i_1, \ldots, i_{\alpha'}\} \) defined in Definition 11, for \( 1 \leq r \leq \alpha' = \min(t, \alpha) \),
\[
\mathbb{E}_{s, T} \left[ \mathbb{E}_{j, Z_s}[\Delta(i_j|S_1, \ldots, w_{-1} \cup C_{j-1})|T(w, s), s = s_j, j \leq r] \right] \geq \frac{1}{k} \left( \mathbb{E}_{s, T}[f(S^*) - f(S_{1, \ldots, w_{-1} \cup C_{j-1}})|T(w, s), s = s_j, j \leq r] \right).
\]
Proof. By substituting \( C_{j-1} = H_{j-1}^{(s)} \) and \( i_j = m_j^s \) in Lemma \[\text{Lemma 15}\] for slot \( s \) and \( r \leq \alpha' \), by conditioning on \((s = s_j \text{ and } j \leq r)\) and using the fact that \( Z_{s_j} \neq \emptyset \), we have

\[E_{j, Z_s}[\Delta(i_j | S_1, ..., w - 1 \cup H_{j-1}^{(s)}) | T(w, s), s = s_j, j \leq r] \geq \frac{1}{k} \left( E_j[f(S^*) - f(S_1, ..., w - 1 \cup C_{j-1}) | s = s_j, j \leq r] \right).\]

Note that by conditioning on \( s = s_j, j \leq r \), Lemma \[\text{Lemma 14}\] holds true (since it will increase the \( \Pr(i \in Z_s) \) for \( i \in S^* \setminus \text{Supp}(T) \)). Taking expectation on \( T(w, s) \) and \( s \) from both sides \((s \text{ is uniformly distributed in } w)\) implies the Corollary. \[\square\]

Now we provide a lower bound on the marginal gain of the set selected by the algorithm on window \( w \), namely \( S_w = H_\alpha \) w.r.t. \( S \) the previously selected items by the algorithm. In other words, we lowerbound the \( f \)-value of the set the algorithm keeps track of by the end of window \( w \), i.e., \( f(S \cup S_w) \).

**Lemma 16.**

\[E[f(S^*) - f(S \cup S_w) | T_1, ..., w - 1] \leq E\left[e^{-\frac{\alpha'}{k}} | T_1, ..., w - 1\right] (f(S^*) - f(S)),\]

where \( \alpha' = \min(\alpha, |\tilde{r}_w|) \).

Proof. First note that \( S_w \) is equal to the set \( H_\alpha \) at the end of window \( w \), i.e., \( S_w = H_\alpha^{(w)} \). Also note that from Proposition \[\text{Proposition 2}\] we have

\[\Delta(S_w | S) \geq \Delta(C_\alpha | S) \geq \Delta(C_{\alpha'}) | S).\]

Therefore,

\[f(S^*) - f(S \cup S_w) \leq f(S^*) - f(S \cup C_{\alpha'}).\]

Let \( \pi_0 = f(S^*) - f(S) \), and for \( 1 \leq j \leq \alpha' \),

\[\pi_j := f(S^*) - f(S \cup C_j), \quad (23)\]

Then, subtracting and adding \( f(S^*) \) from the left hand side of the previous lemma, and taking expectation on \( s, T(w, s), Z \) and \( j \), we get (subscripts of expectations are removed for simplicity)

\[-E_{s, T}[E_j[Z_s][\pi_j - \pi_{j-1} | T(w, s), s = s_{j+1}, j \leq r]] - E[Z_s][\pi_j - \pi_{j-1} | T(w, s), s = s_{j+1}, j \leq r] \]

\[= E[f(S \cup C_j) - f(S \cup C_{j-1}) | T(w, s), s = s_{j+1}, j \leq r] \]

\[= E[f(S \cup H_j^{(s)}) - f(S \cup C_{j-1}) | T(w, s), s = s_{j+1}, j \leq r] \]

\[\geq E[f(S \cup H_j^{(s)}) - f(S \cup C_{j-1}) | T(w, s), s = s_{j+1}, j \leq r] \]

\[\geq E[f(S \cup H_{j-1}^{(s)} \cup \{i_j\}) - f(S \cup C_{j-1}) | T(w, s), s = s_{j+1}, j \leq r] \]

\[\geq E[f(S \cup C_{j-1} \cup \{i_j\}) - f(S \cup C_{j-1}) | T(w, s), s = s_{j+1}, j \leq r] \]

\[\geq E[\Delta(i_j | S \cup C_{j-1}) | T(w, s), s = s_{j+1}, j \leq r] \]

\[\geq \frac{1}{k} E[\pi_{j-1} | T(w, s), s = s_{j+1}, j \leq r] \]

\[= \frac{1}{k} E[\pi_j | T(w, s), s = s_{j+1}, j \leq r - 1] \]
which implies
\[ \mathbb{E}[\pi_j | T(w, s), s = s_{j+1}, j \leq r] \leq \left(1 - \frac{1}{k}\right) \mathbb{E}[\pi_j | T(w, s), s = s_{j+1}, j \leq r-1] \leq \left(1 - \frac{1}{k}\right)^r \pi_0. \]

By martingale stopping theorem, this implies:
\[ \mathbb{E}[\pi_j | T(w, s), s = s_{j+1}, j \leq t] \leq \mathbb{E}\left[\left(1 - \frac{1}{k}\right)^t | T_1, \ldots, w-1\right] \pi_0 \leq \mathbb{E}\left[e^{-t/k} | T_1, \ldots, w-1\right] \pi_0. \]

where stopping time \( t = \alpha' \). \( (t = \alpha' \leq \alpha \) is bounded, therefore, martingale stopping theorem can be applied). 

\[ \blacksquare \]

Using concentration inequalities similar to Lemma 14 of [1], we can bound the size of \(|\tilde{\tau}_w|\) and therefore \(\alpha'\), which implies the following lemma. (because of the new construction of \(T\) we can provide a simpler proof with slightly better bound)

**Lemma 17** (Lemma 14 in [1]). For any real \(\delta' \in (0, 1)\), if parameters \(k, \alpha, \beta\) satisfy \(k \geq \alpha \beta, \beta \geq \frac{1}{\delta'}, \alpha \geq 8\beta^2 \log(1/\delta')\), then given any \(T_1, \ldots, w-1 = T\), with probability at least \(1 - \delta' e^{-\alpha/k}\),
\[ |\tilde{\tau}_w| \geq (1 - \delta')\alpha \]
and therefore w.p. at least \(1 - \delta' e^{-\alpha/k}\), we have \(\alpha' \geq (1 - \delta')\alpha\).

**Lemma 18.** For any real \(\delta' \in (0, 1)\), if parameters \(k, \alpha, \beta\) satisfy \(k \geq \alpha \beta, \beta \geq \frac{1}{\delta'}, \alpha \geq 8\beta^2 \log(1/\delta')\), then
\[ \mathbb{E}[OPT - f(S_{1,\ldots,w}) | T_{1,\ldots,w-1}] \leq (1 - \delta')e^{-\alpha/k} (OPT - f(S_{1,\ldots,w-1})) . \]

Now, similar to [1], by multiplying the inequality Lemma 25 from \(w = 1, \ldots, W\), where \(W = k/\alpha\), we get

**Proposition 3.** For any real \(\delta' \in (0, 1)\), if parameters \(k, \alpha, \beta\) satisfy \(k \geq \alpha \beta, \beta \geq \frac{1}{\delta'}, \alpha \geq 8\beta^2 \log(1/\delta')\), then \(S_{1,\ldots,w}\) tracked by Algorithm 2 satisfies
\[ \mathbb{E}[f(S_{1,\ldots,w})] \geq (1 - \delta')(1 - 1/e)OPT. \]

### 3.5 Bounding \(\mathbb{E}[f(A^\ast)]/OPT\)

In this section, we compare \(f\)-value of \(S_{1,\ldots,w}\) to \(f\)-value of the output of the Algorithm 2 namely \(S_{1,\ldots,w} \cap R\). The difference between the two sets is that \(S \cap R\), the output of the algorithm, only contains element of \(S\) that are in the Shortlist \(R\). An element of \(S\) being missed only if in the arg max in Algorithm 2 that uses the Online Max Algorithm (Algorithm 1 in [1]) does not return the exact max element as part of the shortlist that it returns. From Proposition 3 in [1], the Online Max Algorithm returns a shortlist of size \(O(\log \frac{1}{\delta})\) containing the maximum element w.p. \(\delta = \epsilon/2\).

**Proposition 4.** For any \(\delta \in (0, 1)\), slot \(s\) in window \(w\), and \(1 \leq \ell \leq \alpha, M_{\ell}\) the output of Online Max Algorithm (Algorithm 1 in [1]) in line 5 of Algorithm 4 contains
\[ m_{\ell} \leftarrow \arg \max_{x \in S \cup H_{\ell-1}} \Delta(x | S \cup H_{\ell-1}), \]
with probability \((1 - \epsilon/2)\). In other words, given configuration \(Y\),
\[ \Pr(m_{\ell} \in M_{\ell} | Y, \alpha \in S) \geq 1 - \epsilon/2 \]

15
Proposition 5. The expected $f$-value of output of Algorithm 2 is at least
\[ \mathbb{E}[f(S_1, \ldots, W \cap R)] \geq (1 - \frac{\delta}{2})\mathbb{E}[f(S_1, \ldots, W)]. \]

Proof. From the previous lemma, given any configuration $Y$, we have that each item of $S_1, \ldots, W$ is in $A$ with probability at least $1 - \delta$, where $\delta = \epsilon/2$. Therefore using Lemma 2 the expected value of $f(S_1, \ldots, W \cap R)$ is at least $(1 - \delta)\mathbb{E}[F(S_1, \ldots, W)]$. \qed

Proof of Theorem 1. Now, we can show that Algorithm 2 provides the results claimed in Theorem 1 for appropriate settings of $\alpha, \beta$ in terms of $\epsilon$. Specifically for $\delta' = \epsilon/4$, set $\alpha, \beta$ as smallest integers satisfying $\beta \geq \frac{1}{8}$; $\alpha \geq 8\beta^2 \log(1/\delta')$. Then, using Proposition 3 and Proposition B.1 for $k \geq \alpha\beta$ we obtain:
\[ \mathbb{E}[f(A^*)] \geq (1 - \frac{c}{2})(1 - \delta')(1 - 1/e)\text{OPT} \geq (1 - \epsilon)(1 - 1/e)\text{OPT}. \]

3.6 Streaming: Cardinality

In this section, we show that the algorithm can be implemented in the streaming setting, and we compute the memory required for Algorithm 2. Note that the algorithm designed in [1], for submodular $k$-secretary problem with shortlists needed to be modified slightly in order to make it memory efficient. The complicated part was regarding storing $\alpha$-subsequences efficiently, without requiring to store the entire elements in a window. Fortunately, in this paper our main algorithm for submodular $k$-secretary problem with shortlists is readily memory efficient, and it does not need any adjustment. It is mainly because we have already simplified the procedure responsible for selecting items within a window by employing a hierarchy of subsets $H_1, \ldots, H_\alpha$. Moreover, updating $\{H_\ell\}$ in line 7 of the algorithm for a slot $s$ is pretty straightforward and it only needs access to the sets $\{H_\ell\}$ computed in the previous slot. Additionally, each arg max, can be computed in an online manner using the online max algorithm. It requires memory of size $O(\log 1/\delta)$. All in all, in each iteration of the algorithm, we need to keep track of the following subsets: $S, R, \{H_\ell\}_{\ell=1}^\alpha$ and the shortlists that each of the $\alpha$ arg max keeps track of. Note that w.p. $1 - \delta$, one element of $H_\ell$ does not get selected by the online max algorithm. But Algorithm [1] still needs to keep track of those items separately for computations in the next slots of the same window. Thus we can upper bound the memory usage of the algorithm by $|S| + |R| \leq k + 4k\alpha\beta$. Because there are total of $\alpha\beta(k/\alpha) = k\beta$ slots. In each slot, we run $\alpha$ online max algorithms, each add elements of $M_\ell$ with size $4\log(2/\epsilon)$ to the shortlist $R$. But at the end of the slot we only need to keep the actual maximum element. So we can throw away the rest of the items in each $M_\ell$. Thus, the algorithm needs memory buffer of size $4k\alpha\beta = O_\epsilon(k)$.

Now, let’s bound the number of objective function evaluations for each arriving item. For each new item, it will be involved in computing the arg max in line 5 of the algorithm for $1 \leq \ell \leq \alpha$. We need to compute $\Delta(x|S \cup H_{\ell-1})$ for the new item $x$. However, the arg max is taken over $R \cup s$. Thus in the beginning of each slot we need to compute the marginal gain $\Delta(x|S \cup H_\ell)$ for all the items in $R$, which requires total of $\alpha|R| \leq 4k\alpha\beta$ evaluations. Since the arg max over $R$ is computed only once in the beginning of the slot, the total update time for all the items is bounded by $\frac{1}{\alpha\beta}k\alpha\beta \times k\beta + \alpha \times n = O_\epsilon(n)$. Therefore, the amortized update time for each item is $O_\epsilon(1)$. Furthermore note that our algorithm [1] can be run in parallel, so the computation for each arriving item can be divided between up to $\alpha$ processors. Therefore the total number of evaluation for each processor would be $n + k\beta$. 

16
Theorem 2. For any constant \( \epsilon \in (0, 1) \), there exists an algorithm for the submodular random order streaming problem with matroid constraints that achieves \( 1 - \frac{1}{e} - \epsilon \) approximation algorithm, while using a memory buffer of size at most \( \eta(k) = O(\epsilon(k)) \).

3.7 Improving the number of queries

In this section we show that we can reduce the total number of queries required in each round from \( \alpha \) to \( O(\sqrt{\alpha \log(1/\epsilon)}) = \tilde{O}(1/\epsilon) \). It is done by a slight modification of the algorithm in which in the inner loop for slot \( s \), instead of going over \( 0 \leq \ell \leq \alpha \), we only consider \( s/\beta - \sqrt{s/\beta \log(1/\epsilon)} \leq \ell \leq s/\beta + \sqrt{s/\beta \log(1/\epsilon)} \). By using a Chernoff bound we can show that w.p. at least \( 1 - \epsilon \), the total number of slots \( s' \geq s \) with \( Z_{s'} \neq \emptyset \), is between \( s/\beta - \sqrt{s/\beta \log(1/\epsilon)} \leq \ell \leq s/\beta + \sqrt{s/\beta \log(1/\epsilon)} \).

In other words, w.p. \( (1 - \epsilon) \), we have \( s = s_{j+1} \) for some \( j \) in the above range. Therefore we miss each element of the final solution w.p. \( 1 - \epsilon \), which appears in the approximation factor by using Lemma 2 (note that in this lemma we do not need sampling independently).

3.8 Improving the size of shortlist and memory

In this section, we show how we can reduce the size of shortlist from \( k\alpha \beta \) to only \( k\beta = O(k/\epsilon) \). We modify the algorithm in the way that instead of selecting one items with respect to each layer \( \ell \), the algorithm selects just one item in each slot \( s \) (or \( \log(1/\epsilon) \) many in the shortlist model), whose expected marginal gain with respect to previous layers \( S \cup H' \) is maximized. The expectation can be computed as the number element of \( S^* \) in \( \cup_{s' \leq s} Z_{s'} \) is coming from a binomial distribution with known parameters (depending on \( k, \alpha \) and position of slot \( s \) in the window). (Note that the expectation is for a fixed \( s \) and over all configurations of the random order input; and it is not conditioned on \( T \) that might affect the distribution over layers.). Thus we pick \( H_\ell \) with corresponding probability and multiply it to its marginal gain to compute the expected value. At the end of a slot we compare the selected element with all layers. If its addition to one layer improves the next layer, we modify the next layer. Conditioned on \( Z_{s} \neq \emptyset \) we can lower bound the expected marginal gain similar to \( \ref{lem:chernoff} \) (but the expectation should be on both \( T \) and \( Z \)). By a recursive formula over the slots in a window, we can lower bound the expected marginal gain in that window, which similar to the original proof we can get

\[
\mathbb{E}[OPT - f(S_1, \ldots, s)|T_1, \ldots, w-1] \leq (1 - \epsilon)e^{-\alpha/k}(OPT - f(S_1, \ldots, w-1)) .
\]

Therefore we can achieve the same approximation guarantee with at most one selection per slot. Furthermore previous section we can estimate each expected value with only a few sample in the interval of layers described in the previous section.

4 Matroid Constraints

In this section, we focus on the matroid constraints. We study the submodular matroid secretary problem with shortlists as defined in Section \ref{sec:matroid}.

We propose a new algorithm that improves the approximation ratio, and improves the dependency of size of the shortlist on \( 1/\epsilon \). Our algorithm achieves an approximation ratio \( \frac{1}{2}(1-1/e^2-\epsilon) \) using shortlist of But the the dependency on \( 1/\epsilon \) is \( O(poly(1/\epsilon)) \). Note that in this section \( k := rk(M) \) is the rank of the given matroid \( M \).
The algorithm is based on the algorithm for the cardinality constraints described in Section 3. The building block of our algorithm is again \((\alpha, \beta)\)-windows (refer to [3]). We divide the input into \((\alpha, \beta)\)-windows. We make some modifications to the Algorithm 1 Algorithm 2 and the underlying procedure that it calls, i.e., the online max algorithm (Algorithm 1 in [1]). The main difficulty in designing algorithms for the submodular matroid secretary problem with shortlists in comparison with the simpler submodular \(k\)-secretary problem with shortlists is that the algorithm needs to make sure the set of elements that are going to be returned as the output of the algorithm is an independent set. For the cardinality constraints, the algorithm could add up to \(k\) items to the set of current solution \(S\) without worrying about independence of the new set. Whereas for the matroid constraints we need to remove some of the items from the current solution \(S\), in order to make it independent. In the nutshell, the main difference of the algorithm in this section and Section 3 is the way that the new algorithm deals with these removals. In addition to oracle access to the submodular function \(f\), we assume access to an independence oracle. The independence oracle can verify, in \(O(1)\), whether or not a set is an independent set of the given matroid. Our algorithms are intuitive.

First we define functions \(g\) and \(\theta\) in eq. (24), (25): \(g(e, S)\) is counterpart of \(\Delta(e, S)\) but in matroid setting. In other words, \(g\) maximizes the marginal gain of \(S + e\), after removing possibly one element \(e'\) (selected by \(\theta\)) to make \(S + e - e'\) an independent set. The we use a slight modification of the online max algorithm, for the following problem (Secretary Problem with Replacement): we are given an (independent) set of a matroid and we want to add one item to this set, from a pool of items that are arriving in an online manner, and keep it an independent set by possibly removing some other item from the set. The goal is to maximize the \(f\)-value of the new set.

Then we extend this idea from adding one item to adding multiple items to a given independent set. More precisely, in each window \(w\) (defined similar to the one in Section 3), in addition to set \(S_w\) that is going to be added to the current solution \(S\), we also remove a corresponding set from \(S\) to make it an independent set of the matroid. A crucial lemma in the analysis of the algorithm is Brualdi lemma (refer to Lemma 5). This lemma gives a bijection between two bases of a matroid. We employ the Brualdi Lemma in our Lemma 3, in which we use the bijection provided by Brualdi Lemma to lower bound the \(f\)-value of the remaining set after removing one item by online max algorithm. Intuitively, we prove the marginal gain of the new set, after adding a new item \(a\) and possibly removing some other item to make the set independent, is at least as much as when we remove the corresponding element of \(a\) from the bijection provided by Brualdi Lemma, \(\pi(a)\). Then we argue that \(\pi(a)\) is distributed almost uniformly among elements of the current solution \(S\), thus by Lemma 2 we can lower bound the \(f\)-value of the remaining set.

4.1 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 \(\mathcal{M} = (\mathcal{N}, \mathcal{I})\) 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:
Definition 11. For an independent set $S \in \mathcal{I}$, and $e \in \mathcal{N}$ define
\[ g(e, S) := f(S + e - \theta(e, S)) - f(S), \tag{24} \]
where $\theta$ is defined as:
\[ \theta(e, S) := \arg \max_{e' \in S} \{ f(S + e - e') | S + e - e' \in \mathcal{I} \}. \tag{25} \]

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. We employ the oneline max algorithm, Algorithm 1, in [1] to find:
\[ m \leftarrow \arg \max_{x \in U} g(x, S). \]

Lemma 19 (refer to Proposition 3 in [1]). The online max algorithm, 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, using shortlist of size $O(\log 1/\delta)$.

Algorithm 3 Matroid-Constraint
1: Inputs: submodular function $f$, parameter $\epsilon \in (0, 1]$, and set $S$.
2: Initialize $H_\ell \leftarrow \emptyset, \bar{H}_\ell = \emptyset, \forall 0 \leq \ell \leq \alpha$
3: for every slot $s$ in window $w$ do
4: for $1 \leq \ell \leq \alpha$, concurrently do
5: call the online max algorithm (Algorithm 1 in [1]) to compute, with probability $\epsilon/2$:
6: \[ m_\ell \leftarrow \arg \max_{x \in s \cup R} g(x, SH(w, \ell, s)). \]
7: $o_\ell := \theta(m_\ell, SH(w, \ell, s))$.
8: $M_\ell \leftarrow$ The shortlist returned by the above online max algorithm for slot $s$ and set $H_{\ell-1}$.
9: if $f(SH(w, \ell + 1, s + 1)) < f(SH(w, \ell, s) + m_\ell - o_\ell)$ then
10: \[ H_\ell \leftarrow H_{\ell-1} + m_\ell \]
11: \[ \bar{H}_\ell \leftarrow \bar{H}_{\ell-1} + o_\ell \]
12: \[ R \leftarrow R + (\{m_\ell\} \cap M_\ell) \]
13: end if
14: end for
15: end for
16: return $H_\alpha, \bar{H}_\alpha$

Overview of the main algorithm (Algorithm 4). Similar to the algorithm in Section 3, we divide the input into sequential blocks that we refer to as $(\alpha, \beta)$ windows. (Refer to Definition 4). Intuitively, for large enough $\alpha$ and $\beta$, roughly $\alpha$ items from the optimal set $S^*$ are likely to lie in each of these windows, and further, it is unlikely that two items from $S^*$ appear in the same slot. The algorithm can focus on identifying a constant number (roughly $\alpha$) of optimal items from each of these windows, with at most one item coming from each of the $\alpha \beta$ slots in a window. At the end of window, the algorithm chooses a subset of size $\alpha$, $S_w$, to be added to the current solution $S$. Furthermore, the algorithm removes a subset of $S$ of size at most $\alpha$ from $S$, corresponding to $S_w$, in order to make it an independent set of the matroid.
Algorithm 4  Submodular Matroid Secretary with Shortlists
1: Inputs: number of items \( n \), submodular function \( f \), parameter \( \epsilon \in (0,1] \).
2: Initialize: \( S \leftarrow \emptyset, R \leftarrow \emptyset \), constants \( \alpha \geq 1, \beta \geq 1 \) which depend on the constant \( \epsilon \).
3: Divide indices \( \{1, \ldots, n\} \) into \((\alpha, \beta)\) windows.
4: for window \( w = 1, \ldots, W = k/\alpha \) do
5: \( S_w, \bar{S}_w \leftarrow \text{Cardinality-Constraint}(S,w) \)
6: \( S \leftarrow S + S_w - \bar{S}_w \)
7: end for
8: return \( S \cap R \)

For matroid constraints, in contrast with the cardinality constraints, adding items from a new window to the current solution \( S \) could make it a non-independent set of matroid \( M \). In order to make the new set independent we have to remove some items from \( S \). The removed item corresponding to \( e \) will be \( \theta(e,S) \) as defined in (25). We need to take care of all the removals for newly selected items in a window. Therefore we adapt new notations for the algorithm with matroid constraints:

Notations. Throughout the algorithm, we keep track of \( S_1, \ldots, w \), the current solution from window \( 1, \ldots, w \), and \( R_1, \ldots, w-1 \) the set of the elements that the algorithm keeps in window \( 1, \ldots, w-1 \) as shortlist. Throughout this section, if the subscript of \( S \) and \( R \) is not stated explicitly, we mean \( S_1, \ldots, w-1 \) and \( R_1, \ldots, w-1 \) respectively.

Definition 12. For a slot \( s \in \{1, \ldots, \alpha \beta \} \) in window \( w \). Let’s denote by \( H^s_\ell \), the set \( H_\ell \) as defined in the Algorithm 2 at the end of slot \( s \). Similarly, define \( \bar{H}^s_\ell \) to be the set \( \bar{H}_\ell \) at the end of slot \( s \). (also initialize \( H^0_0 = \bar{H}^0_0 = \emptyset \))

Definition 13. For slot \( s \) in window \( w \), and \( 1 \leq \ell \leq \alpha \), define
\[
SH(w, \ell, s) := (S_1, \ldots, w-1 \cup H^{s-1}_{\ell-1}) \setminus \bar{H}^{s-1}_{\ell-1}.
\] (26)

Definition 14. Define \( m^s_\ell \) to be \( m_\ell \) as defined in Algorithm 7 for slot \( s \), which is
\[
m^s_\ell := \arg \max_{x \in s \cup R(w,s)} g(x, SH(w, \ell, s)),
\] (27)
\[
r^s_\ell := \theta(m_\ell, SH(w, \ell, s)).
\] (28)

Also for the sequence \( \tilde{\tau}_w = (s_1, \ldots, s_t) \) defined in Definition 9, define sequence \( = (i_1, \ldots, i_{\alpha'}) \), and \( \nu_w := (q_1, \ldots, q_{\alpha'}) \) for \( \alpha' = \min(t,\alpha) \), where
\[
i_j := m^s_j,
\] (29)
\[
q_j := r^s_j.
\] (30)

Moreover, for \( 1 \leq j \leq \alpha' \) define
\[
C_j := H^{s_j+1-1}_j
\] (31)
and
\[
\bar{C}_j := \bar{H}^{s_j+1-1}_j
\] (32)

If \( j + 1 > \alpha' \), set \( s_{j+1} := \alpha \beta + 1 \). We also use the notation \( i_1, \ldots, j = (i_1, \ldots, i_j) \), for \( 1 \leq j \leq \alpha' \).
**Definition 15.**

\[ SC(w, j) := (S_1, \ldots, w_{j-1} \cup C_{j-1}) \setminus \bar{C}_{j-1} \]  

(33)

The following observations are immediate:

**Proposition 6.** For slot \( s \in w \), and \( 1 \leq \ell \leq \alpha \),

\[ f(SH(w, \ell, s)) \geq f(SH(w, \ell - 1, s - 1) + m_s^\ell), \]  

(34)

\[ f(SH(w, \ell, s)) \geq f(SH(w, \ell - 1, s)), \]  

(35)

\[ f(SH(w, \ell, s + 1)) \geq f(SH(w, \ell, s)), \]  

(36)

\[ f(SC(w, \ell)) \geq f(SC(w, \ell - 1)), \]  

(37)

\[ f(S + S_w - \bar{S}_w) \geq f(SC(w, \alpha)). \]  

(38)

**Proposition 7.** For each window \( w \in [W] \), \( S_1, \ldots, w \) is an independent set of \( \mathcal{M} \). Similarly, \( SH(w, \ell, s) \) is also an independent set \( \mathcal{M} \).

We define \( T_w, T_1, \ldots, w, T(w, s) \) and \( Supp(T_w), Supp(T_1, \ldots, w), Supp(T(w, s)) \) and \( R_1, \ldots, w, R(w, s) \) similar to Definition 7, 8, in Section 3 but we define it based on the new definition of \( H_\ell^s \) as defined above in Definition 12. To avoid confusion we redefine it here:

**Definition 16.** For window \( w \in [W] \), define

\[ T_w := \{ H_\ell^s | s \in w, 1 \leq \ell \leq \alpha \}, \]  

(39)

moreover,

\[ T_1, \ldots, w := \bigcup_{i=1}^w T_i, \]

and

\[ T(w, s) := T_1, \ldots, w_{s-1} \cup \{ H_{\ell}^s | s > w \}_{s'}, 1 \leq \ell \leq \alpha \}. \]  

(40)

**Definition 17.** For slot \( s \) in window \( w \) define

\[ Supp(T_w) := \bigcup_{1 \leq \ell \leq \alpha, s \in w} H_\ell^s, \]

also,

\[ Supp(T_1, \ldots, w) := \bigcup_{i=1}^w Supp(T_i), \]

and,

\[ Supp(T(w, s)) := Supp(T_1, \ldots, w_{s-1}) \cup \bigcup_{1 \leq \ell \leq \alpha; s > w \_{s'}} H_{\ell}^{s'}. \]

(Note that \( Supp(T_1, \ldots, w) = R_1, \ldots, w \)). Furthermore, define \( R(w, s) := Supp(T(w, s)) \).
Moreover, as described in Algorithm 4 define
\[ S_w := H_\alpha^\beta, \]  
(41)
and
\[ \bar{S}_w := \overline{H}_\alpha^\beta, \]  
(42)

Lemma 20. The size of the shortlist \( R_1, \ldots, W \) that Algorithm 4 uses is at most \( 4k\alpha\beta \log(2/\epsilon) \).

Proof. Similar to the proof of Lemma 6

In the next section, we will show that
\[ \mathbb{E}[f(S \cap R)] \geq \frac{1}{2} (1 - \frac{1}{2^d} - \epsilon) f(S^*) \]
to provide a bound on the competitive ratio of Algorithm 4 for submodular matroid secretary problem with shortlists.

4.2 Analysis of the algorithms (Matroids)

In this section we show that for any \( \epsilon \in (0, 1) \), Algorithm 2 with an appropriate choice of constants \( \alpha, \beta \), achieves the competitive ratio claimed in Theorem ?? for the submodular matroid secretary problem with shortlists.

First, we show the existence of a random subsequence of slots \( \tilde{r}_w \) of window \( w \) such that we can lower bound \( f(S + S_w - \bar{S}_w) - f(S) \) in terms of \( \text{OPT} - 2f(S) \) in every window. \( T \) as defined in 7, roughly speaking captures the selections the algorithm has made in the previous windows. In the following lemmas suppose the sequence \( \tilde{r}_w = (s_1, \ldots, s_t) \), and \( Z_{s_1}, \ldots, Z_{s_{s_j-1}} \) defined as in Definition 9. Intuitively, conditioned on \( T \), different elements of \( S^* \) have different probability of appearing in a slot \( s \) in \( w \). By subsampling sets \( \{Z_s\} \), make these probabilities even (Note that \( T, \tilde{r}, \) and \( Z_s \) are for the purpose of analysis. The algorithm does not keep track of these variables). We will use Lemma 13, which also holds true for the new definition of \( T \).

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 21. For slot \( s \) in a window \( w \), and \( 1 \leq \ell \leq \alpha' = \min(t, \alpha) \),
\[ \mathbb{E}[\Delta(m_s^\ell|SH(w, \ell, s))|T(w, s), Z_s \neq \emptyset] \geq \frac{1}{k} (f(S^*) - f(SH(w, \ell, s))) \]

Proof. From Definition 10, \( m_s^\ell \) is chosen greedily to maximize the increment
\[ \arg \max_{x \in S \cup R(w, s)} \Delta(x|SH(w, \ell, s)), \]
and so \( m_s^\ell \) belongs to \( s \cup R(w, s) \supseteq Z_s \). Therefore, we can lower bound the marginal gain of \( m_s^\ell \) w.r.t. previously selected items \( SH(w, \ell, s) \) by the marginal gain of a randomly picked item \( i \) from \( Z_s \) as follows.

\[ \mathbb{E}[\Delta(m_s^\ell|SH(w, \ell, s))|T(w, s), Z_s \neq \emptyset] \geq \frac{1}{k} \sum_{i \in S^*} \Delta(i|SH(w, \ell, s)) \]
(\text{using Lemma 10, monotonicity of } f) \geq \frac{1}{k} (f(S^*) - f(SH(w, \ell, s))) \]

\[ \square \]
Corollary 2. Given the sequence $\bar{\tau}_w = (s_1, \ldots, s_t)$ defined in Definition 9 and $\mu_w = \{i_1, \ldots, i_{\alpha'}\}$ defined in Definition 10 for $1 \leq j \leq \alpha' = \min(t, \alpha)$,

\[ E[\Delta(i_j|SC(w, j))]T(w, s), s = s_j] \geq \frac{1}{k}E[f(S*) - f(SC(w, j))]T(w, s), s = s_{j-1}] . \]

Now we use the Brualdi lemma (refer to Lemma 5), to create a bijection $\pi$ between a base of matroid containing the current solution $SH(w, \ell, s)$ and the optimal solution. Then for an element $a$ of the optimal solution in slot $Z_a$, if we remove its corresponding element $\pi(a)$ from the current solution, we can still lower bound the value of the remaining set.

Lemma 22. Let $S'$ be the extension of $SH(w, \ell, s)$ to a base of $\mathcal{M}$ (refer to Lemma 4). Let $\pi$ be the bijection from Brualdi lemma (refer to Lemma 2) from $S^*$ to $S'$. Then,

\[ E[f(SH(w, \ell, s) - \pi(a))T(w, s), a \in S^* \cap Z_s] \geq (1 - \frac{1}{k})f(SH(w, \ell, s)) . \]

Proof. Since $\pi$ is a bijection from $S^*$ to $S'$, from Brualdi’s lemma (Lemma 5), $SH(w, \ell, s) - \pi(a) + a \in T$, for all $a \in S^*$. Recall the definition of $Z_{s_j}$. Suppose $a$ is a randomly picked item from $S^* \cap Z_{s_j}$. Since $Z_{s_j} \neq \emptyset$, using Lemma 13 conditioned on $T(w, s)$, the element $a$ can be equally any element of $S^*$ with probability $1/k$. Therefore, $\pi(a)$ would be any of $SH(w, \ell, s)$ with probability at most $1/k$, i.e.,

\[ Pr(\pi(a) = e|T(w, s)) \leq 1/k, \text{ for } e \in SH(w, \ell, s), \]

Now based on the definition of $\pi$ and Lemma 2 the lemma follows.

Corollary 3. Let $S'$ be the extension of $SC(w, j)$ to a base of $\mathcal{M}$ (refer to Lemma 4). Let $\pi$ be the bijection from Brualdi lemma (refer to Lemma 2) from $S^*$ to $S'$. Then, for all $j = 1, \ldots, \alpha'$,

\[ E[f(SC(w, j) - \pi(a))T(w, s), s = s_j, a \in S^* \cap Z_{s_j}] \geq (1 - \frac{1}{k})f(SC(w, j)) . \]

Lemma 23. For all $j = 1, \ldots, \alpha'$,

\[ E[f(SC(w, j) - f(SC(w, j - 1))]T(w, s_j)] \geq \frac{1}{k}E[f(S^*) - 2f(SC(w, j - 1))]T(w, s_{j-1})] . \]

Proof. In the Algorithm 1 at the end of window $w$, we set $S = S + S_w - \bar{S}_w$. Suppose $a \in s_j \cap S^*$. Moreover, let $S'$ be the extension of $SC(w, j - 1)$ to a base of $\mathcal{M}$, and $\pi$ be the bijection from Brualdi’s Lemma (refer to Lemma 5) 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 $s_j$) is as follows.

\[ E[f(SC(w, j)|T(w, s_j)] \]
\[ \geq E[f(SC(w, j - 1) + a - \pi(a)]T(w, s_j), a \in S^* \cap Z_{s_j}] \]
\[ \geq E[f(SC(w, j - 1) - \pi(a)]T(w, s_j), a \in S^* \cap Z_{s_j}] + \]
\[ E[\Delta(a|SC(w, j - 1) - \pi(a))]T(w, s_j), a \in S^* \cap Z_{s_j}] \]
\[ \geq E[f(SC(w, j - 1) - \pi(a)]T(w, s_j), a \in S^* \cap Z_{s_j}] \]
\[ + E[\Delta(a|SC(w, j - 1))]T(w, s_j), a \in S^* \cap Z_{s_j}] . \]
The first inequality is from the definition of function $g$ as it is defined in equation 24. The last inequality is from submodularity of $f$. Now from the last inequality and lemma 3 we have

$$\mathbb{E}[f(SC(w, j)) | T(w, s_j)] \geq (1 - \frac{1}{k})f(SC(w, j - 1)) + \frac{1}{k}(f(S^*) - f(SC(w, j - 1))) .$$

Thus,

$$\mathbb{E}[f(SC(w, j)) - f(SC(w, j - 1)) | T(w, s_j)] \geq \frac{1}{k}(f(S^*) - 2f(SC(w, j - 1))) .$$

Hence, by taking expectation on $T(w, s_j - 1)$, and by Proposition 4

$$\mathbb{E}[f(SC(w, j)) - f(SC(w, j - 1)) | T(w, s_j)] \geq \frac{1}{k}\mathbb{E}[f(S^*) - 2f(SC(w, j - 1)) | T(w, s_j - 1)] .$$

\[ \square \]

From the standard techniques for the analysis of greedy algorithm, we can show that,

**Lemma 24.**

$$\mathbb{E} [f(S^*) - 2f(S_{1,\ldots,w-1} \cup S_w)] | T] \leq \mathbb{E} \left[ e^{\frac{-2\alpha'}{k}} | T_{1,\ldots,w-1} \right] (f(S^*) - 2f(S_{1,\ldots,w-1})) .$$

**Proof.** First note that $S_w$ is equal to the set $H_\alpha$ at the end of window $w$, i.e., $S_w = H_\alpha^\alpha$. Also note that from Proposition 2 we have

$$f(S + S_w - \bar{S}_w) \geq f(SC(w, \alpha)) \geq f(SC(w, \alpha'))$$

Therefore,

$$f(S^*) - f(S + S_w - \bar{S}_w) \leq f(S^*) - f(SC(w, \alpha'))$$

Let $\pi_0 = f(S^*) - 2f(S)$, and for $1 \leq j \leq \alpha'$,

$$\pi_j := f(S^*) - 2f(SC(w, j + 1)) ,$$

(43)

Then, subtracting and adding $f(S^*)$ from the left hand side of the previous lemma, and taking expectation conditional on $T(w, s)$, we get

$${-\frac{1}{2}} \mathbb{E} [\pi_j - \pi_{j-1} | T(w, s), s = s_j]$$

(By eq. 23) $= \mathbb{E} [f(SC(w, j + 1) | T(w, s), s = s_j) - f(SC(w, j))$

(By Definition 10) $= \mathbb{E} [f(S + H_{j+1}^{-j+1} - \bar{H}_{j+1}^{-j+1}) | T(w, s), s = s_j] - f(SC(w, j))$

(By Proposition 6) $= \mathbb{E} [f(S + H_j^{-j+1} - \bar{H}_j^{-j+1}) | T(w, s), s = s_j] - f(SC(w, j))$

(By eq. 18) $\geq \mathbb{E} [f(S + H_{j+1}^{-j+1} - \bar{H}_{j+1}^{-j+1} + i_{j+1}) | T(w, s), s = s_j] - f(SC(w, j))$

(By eq. 21) $\geq \mathbb{E} [f(SC(w, j + 1) - f(SC(w, j)) | T(w, s), s = s_j]$

(By Definition of $\Delta$) $\geq \mathbb{E} [\Delta(i_j | SC(w, j)) | T(w, s), s = s_j]$

(By Corollary 2) $\geq \frac{1}{k} \mathbb{E} [\pi_{j-1} | T(w, s), s = s_{j-1}] .$
which implies
\[
E[\pi_j|T(w,s), s = s_j] \leq \left(1 - \frac{2}{k}\right) E[\pi_{j-1}|T(w,s), s = s_{j-1}] \leq \left(1 - \frac{2}{k}\right)^j \pi_0 .
\]

By martingale stopping theorem, this implies:
\[
E[\pi_t|T(w,s), s = s_t] \leq \left(1 - \frac{2}{k}\right)^t |T_1,...,w-1| \pi_0 \leq E \left[e^{-2t/k}|T_1,...,w-1| \pi_0 .
\]

where stopping time \(t = \alpha'\). (\(t = \alpha' \leq \alpha \beta\) is bounded, therefore, martingale stopping theorem can be applied).

\[\square\]

We need a bound on size of \(\bar{\tau}_w\), and \(\alpha'\). By concentration inequalities proven in Lemma 14 in [1], we can show the following Lemma.

**Lemma 25.** For any real \(\delta' \in (0, 1)\), if parameters \(k, \alpha, \beta\) satisfy \(k \geq \alpha \beta, \beta \geq \frac{1}{\delta'}, \alpha \geq 8 \beta^2 \log(1/\delta')\), then
\[
E \left[OPT - 2f(S_1,...,w)|T_1,...,w-1\right] \leq (1 - \delta')e^{-2\alpha/k} (OPT - 2f(S_1,...,w-1)) .
\]

**Theorem 3.** For any constant \(\epsilon > 0\), there exists an online algorithm (Algorithm 4) for the submodular matroid secretary problem with shortlists that achieves a competitive ratio of \(\frac{1}{2}(1 - \frac{1}{e^2} - \epsilon)\), with shortlist of size \(\eta_\epsilon(k)\). The running time of this algorithm is \(O(\epsilon \cdot nk)\).

By setting \(\alpha = \beta = 1\), we can show the following Corollary.

**Corollary 4.** For the matroid secretary problem in the preemption model, and matroid secretary problem that uses shortlist of size at most \(\eta(k)\), there is an algorithm that achieves a constant competitive ratio.

### 4.3 Streaming: matroids

In this section, we show that the algorithm for submodular matroid secretary problem with shortlists ,i.e., Algorithm 4, can be implemented in the streaming setting, and we compute the memory required for Algorithm 2, the total number of function evaluations and access to the independence oracle.

The algorithm for matroid constraints stores \(\{H_\ell\}_\ell^\alpha\) in addition to \(\{H_\ell\}_\ell^\alpha\). In each iteration of the algorithm, we need to keep track of the following subsets: \(SH(w,\ell,s), R, \{H_\ell\}_\ell^\alpha, \{H_\ell\}_\ell^\alpha\) and the shortlists that each of the \(\alpha\) arg max keeps track of. Since for matroid constraints, the function \(\theta\) returns only one item (for \(p\)-matchoid constraints it is set of size at most \(p\)), the size of \(|\bar{H}_\ell| = |H_\ell|\). Therefore, the memory buffer required for Algorithm 4 can be upper bounded by \(|SH(w,\ell,s)| + |R| + |Supp(T_w)| \leq k + 4k\alpha\beta + \alpha^2 \beta = O_\epsilon(k)\).

Now, let’s bound the number of objective function evaluations and total number of accessing the independence oracle of the matroid for each arriving item. For each new item, it will be involved in computing the arg max in line 5 of the algorithm for \(1 \leq \ell \leq \alpha\). We need to compute \(g(x|S \cup H_{\ell-1})\) for the new item \(x\). Computing \(g\) and therefore \(\theta\) requires at most \(k\) function evaluations. It is because after adding one new item \(e\) to the \(SH(w,\ell,s)\) we should find all the elements whose removal make the new set an independent set of the matroid. Thus we need to go over all the items

25
in $SH(w, \ell, s)$, which is at most $k$. Thus the total number of function evaluation is at most $k$ times of the same amount for the case of cardinality constraint. Hence it is $O_\epsilon(nk + k^3) = O_\epsilon(nk + k^3)$. The total number of access to the independence oracle is similarly $O_\epsilon(\eta) = O_\epsilon(\eta(k))$.

**Theorem 4.** For any constant $\epsilon \in (0, 1)$, there exists an algorithm for the submodular random order streaming problem with matroid constraints that achieves $\frac{1}{2}(1 - \frac{1}{e^2} - \epsilon)$ approximation algorithm, while using a memory buffer of size at most $\eta(k) = O_\epsilon(k)$.

5 **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. A formal definition of a $p$-matchoid is in [9] and in the appendix. Throughout this section, $k$ would refer to the size of the largest feasible set.

We make some modifications in the algorithm in Section 4 and the analysis provided there. The main difference in the algorithm is that we update functions $g$ and $\theta$ defined in Definition 11. Here, function $\theta$, instead of one item, might remove up to $p$ items form the current independent set $S$. Each removed item corresponds to different ground set $N_i$, in which the new item lies (based on the definition of $p$-matchoid constraints, there are at most $p$ such elements).

**Definition 18 ([24]).** For each matroid $M_\ell = (N_\ell, I_\ell)$, and $\ell \in [q]$ define:

$$\Omega_\ell(e, S) := \{e' \in S | S + e - e' \in I_\ell\}. \hspace{1cm} (44)$$

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

$$\lambda(e, S) := \prod_{i=1}^{t} \Omega_\ell_i(e, S). \hspace{1cm} (45)$$

For a combination vector $r = (r_1, \cdots, r_p) \in \lambda(e, S)$, where $r_i \in \Omega_\ell_i(e, S)$, define the union of all the components of $r$ as:

$$\mu(r) := \{r_1, \cdots, r_p\}. \hspace{1cm} (46)$$

$$g_r(e, S) := f(S + e - \mu(r)) - f(S). \hspace{1cm} (47)$$

Also define:

$$\theta(e, S) := \mu(\arg \max_{r \in \lambda(e, S)} g_r(e, S)). \hspace{1cm} (48)$$

Furthermore define,

$$g(e, S) := \max_{r \in \lambda(e, S)} g_r(e, S). \hspace{1cm} (49)$$

Now using the new definition of $g$, we employ the oneline max algorithm, to find:

$$m_\ell \leftarrow \arg \max_{x \in S \cup R} g(x, S). \hspace{1cm} (50)$$
Accordingly we will update line 5 of Algorithm 3 by this new definition of \( g \) in eq. (49). It returns element \( e \) with maximum \( g(e, S) \), and it achieves a \( 1 - \delta \) competitive ratio with shortlists of size logarithmic in \( 1/\delta \). Here, the output of \( \theta \) in Algorithm 3 is a set instead of only one item:

\[
o_{\ell} := \theta(m_{\ell}, SH(w, \ell, s)).
\]

Additionally, we make some changes in Algorithm 3. We define \( H_{\ell}^s \) and \( \bar{H}_{\ell}^s \) defined in Definition 12 using the new definition of \( g \) in eq. (49). Note that in each update a set would be added to \( \bar{H}_{\ell}^S \), whereas for the matroid constraints it was only one item.

Furthermore, we define \( C_j, \bar{C}_j, SH(w, \ell, s), SC(w, j), S_w \) and \( \bar{S}_w \) similar to their definition in Section 4, using new definition of \( g \) and \( \theta \).

**Theorem 5.** 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 \( \frac{1}{p+1}(1 - \frac{1}{e} - \epsilon - O(\frac{1}{k})) \) approximation to \( \text{OPT} \) while using a memory buffer of size at most \( \eta(k) = O(k) \). Also, the number of objective function evaluations for each item, amortized over \( n \) items, is \( O(pk + \kappa p + \frac{k^2}{n}) \), where \( \kappa = \max_{i \in [q]} \text{rk}(M_i) \).

**Proof.** The proof is based on the recursion we get in Lemma 28 in the Appendix. It is similar to proof of Lemma 16 for the matroid constraints which omitted.

\( \square \)

### 6 Streaming

In this section, we show that Algorithm ?? can be implemented in a way that it uses a memory buffer of size at most \( \eta(k) = O(k) \). For \( p \)-matchoid constraint we have the following result for the streaming setting. The proof appears in the appendix.

**Theorem 6.** For any constant \( \epsilon > 0 \), there exists an algorithm for the submodular random order streaming problem with \( p \)-matchoid constraints that achieves a competitive ratio of \( \frac{1}{p+1}(1 - \frac{1}{e} - \epsilon - O(\frac{1}{k})) \) approximation to \( \text{OPT} \) while using a memory buffer of size at most \( \eta(k) = O(k) \). Also, the number of objective function evaluations for each item, amortized over \( n \) items, is \( O(pk + \kappa p + \frac{k^2}{n}) \), where \( \kappa = \max_{i \in [q]} \text{rk}(M_i) \).

**Proof.** The proof is similar to the one for matroid constraints, Theorem 4 which is omitted.

\( \square \)

### References

[1] Shipra Agrawal, Mohammad Shadravan, and Cliff Stein. Submodular secretary problem with shortlists. In 10th Innovations in Theoretical Computer Science Conference (ITCS 2019). Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik, 2018.

[2] Miklos Ajtai, Nimrod Megiddo, and Orli Waarts. Improved algorithms and analysis for secretary problems and generalizations. *SIAM J. Discret. Math.*, 14(1):1–27, January 2001.

[3] Moshe Babaioff, Nicole Immorlica, and Robert Kleinberg. Matroids, secretary problems, and online mechanisms. In *Proceedings of the eighteenth annual ACM-SIAM symposium on Discrete algorithms*, pages 434–443. Society for Industrial and Applied Mathematics, 2007.

[4] Moshe Babaioff, Nicole Immorlica, David Kempe, and Robert Kleinberg. Online auctions and generalized secretary problems. *SIGecom Exch.*, 7(2):7:1–7:11, June 2008.
[5] Ashwinkumar Badanidiyuru, Baharan Mirzasoleiman, Amin Karbasi, and Andreas Krause. Streaming submodular maximization: Massive data summarization on the fly. In Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD ’14, pages 671–680, New York, NY, USA, 2014. ACM.

[6] Mohammadhossein Bateni, Mohammadtaghi Hajiaghayi, and Morteza Zadimoghaddam. Submodular secretary problem and extensions. ACM Trans. Algorithms, 9(4):32:1–32:23, October 2013.

[7] Richard A Brualdi. Comments on bases in dependence structures. Bulletin of the Australian Mathematical Society, 1(2):161–167, 1969.

[8] Niv Buchbinder, Moran Feldman, Joseph (Seffi) Naor, and Roy Schwartz. Submodular maximization with cardinality constraints. In Proceedings of the Twenty-fifth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA ’14, pages 1433–1452, Philadelphia, PA, USA, 2014. Society for Industrial and Applied Mathematics.

[9] Chandra Chekuri, Shalmoli Gupta, and Kent Quanrud. Streaming algorithms for submodular function maximization. In Magnús M. Halldórsson, Kazuo Iwama, Naoki Kobayashi, and Bettina Speckmann, editors, Automata, Languages, and Programming, pages 318–330, Berlin, Heidelberg, 2015. Springer Berlin Heidelberg. ISBN 978-3-662-47672-7.

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

[11] Uriel Feige, Vahab S. Mirrokni, and Jan Vondrák. Maximizing non-monotone submodular functions. SIAM J. Comput., 40(4):1133–1153, July 2011.

[12] Moran Feldman and Rico Zenklusen. The submodular secretary problem goes linear. In Proceedings of the 2015 IEEE 56th Annual Symposium on Foundations of Computer Science (FOCS), FOCS ’15, pages 486–505, Washington, DC, USA, 2015. IEEE Computer Society.

[13] Moran Feldman, Ola Svensson, and Rico Zenklusen. A simple o (log log (rank))-competitive algorithm for the matroid secretary problem. In Proceedings of the twenty-sixth annual ACM-SIAM symposium on Discrete algorithms, pages 1189–1201. SIAM, 2014.

[14] Moran Feldman, Amin Karbasi, and Ehsan Kazemi. Do less, get more: Streaming submodular maximization with subsampling, 2018.

[15] Anupam Gupta, Aaron Roth, Grant Schoenebeck, and Kunal Talwar. Constrained non-monotone submodular maximization: Offline and secretary algorithms. In Proceedings of the 6th International Conference on Internet and Network Economics, WINE’10, pages 246–257, Berlin, Heidelberg, 2010. Springer-Verlag.

[16] Elad Hazan, Shmuel Safra, and Oded Schwartz. On the complexity of approximating k-set packing. Computational Complexity, 15(1):20–39, 2006.

[17] Ehsan Kazemi, Marko Mitrovic, Morteza Zadimoghaddam, Silvio Lattanzi, and Amin Karbasi. Submodular streaming in all its glory: Tight approximation, minimum memory and low adaptive complexity, 2019.
[18] Thomas Kesselheim and Andreas Tönnis. Submodular Secretary Problems: Cardinality, Matching, and Linear Constraints. In Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques (APPROX/RANDOM 2017), Leibniz International Proceedings in Informatics (LIPIcs), pages 16:1–16:22, 2017.

[19] Robert Kleinberg. A multiple-choice secretary algorithm with applications to online auctions. In Proceedings of the Sixteenth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA ’05, pages 630–631, Philadelphia, PA, USA, 2005. Society for Industrial and Applied Mathematics.

[20] Oded Lachish. O (log log rank) competitive ratio for the matroid secretary problem. In 2014 IEEE 55th Annual Symposium on Foundations of Computer Science, pages 326–335. IEEE, 2014.

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

[22] George L Nemhauser, Laurence A Wolsey, and Marshall L Fisher. An analysis of approximations for maximizing submodular set functions—i. Mathematical programming, 14(1):265–294, 1978.

[23] Ashkan Norouzi-Fard, Jakub Tarnawski, Slobodan Mitrovic, Amir Zandieh, Aidasadat Mousavifar, and Ola Svensson. Beyond 1/2-approximation for submodular maximization on massive data streams. In Proceedings of the 35th International Conference on Machine Learning, volume 80, pages 3829–3838. PMLR, 10–15 Jul 2018.

[24] Mohammad Shadravan. Submodular matroid secretary problem with shortlists, 2020.

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

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

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

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

**Lemma 26** (Azuma-Hoeffding inequality). Suppose \( \{X_k : k = 0, 1, 2, 3, \ldots \} \) 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):

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

B Missing Proofs in Matroid Constraints (Section 4)

B.1 Proof of Theorem 3

**Proof.** Now from Lemma 25, we have, for any real \( \delta' \in (0, 1) \), if parameters \( k, \alpha, \beta \) satisfy \( k \geq \alpha \beta \), \( \beta \geq \frac{1}{\delta'} \), \( \alpha \geq 8 \beta^2 \log(1/\delta') \), then the set \( S_1, \ldots, W \) tracked by Algorithm 2 satisfies

\[
\mathbb{E}[f(S_1, \ldots, W)] \geq (1 - \delta')(1 - 1/e^2)OPT.
\]

Now, we compare \( f(S_1, \ldots, W) \) to \( f(S \cap R) \), with \( R \) being the shortlist returned by Algorithm 4. The main difference between the two sets is that in the construction of shortlist \( R \), Algorithm 4 is being used to compute the argmax in the definition of \( H_\ell \), 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 \( R \).

Similar to Lemma 16 in [1], we can show that each element in \( R \) gets selected by the algorithm with probability at least \( 1 - \delta \). More precisely, let \( A \) be the shortlist returned by Algorithm 2 and \( \epsilon \) is the parameter used to call Algorithm 4. Then, for given configuration \( Y \), for any item \( a \), we have

\[
Pr(a \in A | Y, a \in SH(w, \ell, s)) \geq 1 - \epsilon/2.
\]

Therefore using Lemma 2,

\[
\mathbb{E}[f(S \cap R)] \geq (1 - \frac{\epsilon}{2})\mathbb{E}[f(S_1, \ldots, W)].
\]

subset of shortlist \( A \) returned by Algorithm 2. Thus by setting \( \delta' = \epsilon/4 \), for \( k > \alpha \beta \), we obtain lower bound \( \frac{1}{2}(1 - 1/e^2 - \epsilon)OPT \). The running time of the algorithm is discussed in the streaming section. \( \square \)
C Missing Proofs in the $p$-matchoid Constraints Section

Now we can generalize Lemma 3 to $p$-matchoid constraints.

Suppose the sequence $\tilde{\tau}_w = (s_1, \ldots, s_t)$ defined as in Definition 9.

**Definition 19.** For slot $s$ in window $w$, and $1 \leq \ell \leq \alpha$, define
\[
SH(w, \ell, s) := (S_1, \ldots, w_{-1} \cup \tilde{H}^{s-1}_{\ell-1}) \setminus \tilde{H}^{s-1}_{\ell-1}.
\]

**Definition 20.** Define $m^s_j$ to be $m_\ell$ as defined in Algorithm 7 for slot $s$, which is
\[
m^s_j := \arg \max_{x \in S \cup R(w, s)} g(x, SH(w, \ell, s)),
\]
and
\[
r^s_j := \theta(m_\ell, SH(w, \ell, s)).
\]

Also for the sequence $\tilde{\tau}_w = (s_1, \ldots, s_t)$ defined in Definition 7, define sequence $= (i_1, \ldots, i_{\alpha'})$, and $\nu_w := (q_1, \ldots, q_{\alpha'})$ for $\alpha' = \min(t, \alpha)$, where
\[
i_j := m^s_j,
\]
and
\[
q_j := r^s_j.
\]

Moreover, for $1 \leq j \leq \alpha'$ define
\[
C_j := H^j_{s+1} - 1
\]
and
\[
C_j := H^j_{s+1} - 1
\]
If $j + 1 > \alpha'$, set $s_{j+1} := \alpha \beta + 1$. We also use the notation $i_1, \ldots, j = (i_1, \ldots, i_j)$, for $1 \leq j \leq \alpha'$.

**Definition 21.**
\[
SC(w, j) := (S_1, \ldots, w_{-1} \cup C_{j-1}) \setminus C_{j-1}.
\]

For any slot $s$ in window $w$, and element $b \in N_i$, let $S_i^j$ be the extension of $SH(w, \ell, s)$ to a base of $M_i$ (refer to Lemma 11), and $\pi_i$ be the bijection from Brualdi lemma (refer to Lemma 5) from $S^*$ to $S_i^j$. Further, let’s denote
\[
\pi(b) := \{\pi_i(b)|b \in N_i\}.
\]

**Lemma 27.** For slot $s$ in window $w$, and $\pi$ as defined in eq. (23),
\[
E[f(SH(w, \ell, s) - \pi(a))|T(w, s), a \in S^* \cap Z_a] \geq (1 - \frac{p}{k})f(SH(w, \ell, s)).
\]

**Proof.** The proof is similar to the proof of Lemma 3. For $\ell \in [q]$, since $\pi_\ell$ is a bijection from $S^* \cap N_\ell$ to $S_\ell$, we have $S(w, \ell, s) - \pi_\ell(a) + a \in M_\ell$, for all $a \in S^*$.

Recall the definition of $Z_\phi$. Suppose $a$ is a randomly picked item from $S^* \cap Z_\phi$. Since $Z_\phi \neq \emptyset$, using Lemma 13 conditioned on $T(w, s)$, the element $a$ can be equally any element of $S^*$ with probability $1/k$. Therefore, $\pi_i(a)$ would be any element of $SH(w, \ell, s)$ with probability at most $1/k$, i.e.,
\[
\Pr(\pi_i(a) = e|T(w, s)) \leq 1/k, \text{ for } e \in SH(w, \ell, s), i \in [q]
\]

31
For element \( e \in SH(w, \ell, s) \), let \( \mathcal{N}(e) \) be the set of indices \( i \) such that \( e \in \mathcal{N}_i \). Because of the \( p \)-matchoid constraint, we have \( |\mathcal{N}(e)| \leq p \). Define
\[
\pi^{-1}(e) := \{ t | t \in \mathcal{N}_i \text{ for some } i \in \mathcal{N}(e) \text{ and } \pi_i(t) = e \}.
\]
we have also \( |\pi^{-1}(e)| \leq p \). Thus, each element \( e \in SH(w, \ell, s) \) belongs to \( \pi(a) \) with probability at most \( p/k \):
\[
\Pr(e \in \pi(a)|T(w, s_j), a \in S^* \cap Z_{s_j}) = \Pr(a \in S^* \cap Z_{s_j} \cap \pi^{-1}(e)|T(w, s_j)) \leq \frac{p}{k}.
\]
Now we apply Lemma 2. It is crucial to note that in Lemma 2 each element do not necessarily need to be selected independently. Definition of \( \pi \) and lemma 2 imply the lemma. \( \Box \)

**Corollary 5.** For slot \( s \) in window \( w \), and \( \pi \) as defined in eq. (23). Then, for all \( j = 1, \ldots, \alpha' \),
\[
\mathbb{E}[f(SC(w, j) - \pi(a))|T(w, s), s = s_j, a \in S^* \cap Z_{s_j}] \geq (1 - \frac{p}{k})f(SC(w, j)).
\]

**Lemma 28.** For all \( j = 1, \ldots, \alpha' \),
\[
\mathbb{E}[f(SC(w, j)) - f(SC(w, j - 1))|T(w, s_j)] \geq \frac{1}{k} \mathbb{E}[f(S^*) - (p + 1)f(SC(w, j - 1)|T(w, s_j - 1))].
\]

**Proof.** In the Algorithm 4 at the end of window \( w \), we set \( S = S + S_w - \bar{S}_w \). Suppose \( a \in s_j \cap S^* \). Moreover, let \( S'_\ell \) be the extension of \( SC(w, j - 1) \) to an independent set in \( \mathcal{M}_\ell \), and \( \pi_\ell \) be the bijection in Brualdi lemma (refer to Lemma 5) from \( S^*_\ell \) to \( S'_\ell \). Further, let’s denote
\[
\pi(b) := \{ \pi_i(b) | b \in \mathcal{N}_i \}.
\]

Then, 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 \( s_j \)) is as follows.
\[
\mathbb{E}[f(SC(w, j)|T(w, s_j)]
\geq \mathbb{E}[f(SC(w, j - 1) + a - \pi(a))|T(w, s_j), a \in S^* \cap Z_{s_j}]
\geq \mathbb{E}[f(SC(w, j - 1) - \pi(a))|T(w, s_j), a \in S^* \cap Z_{s_j}]
\geq \mathbb{E}[\Delta(a|SC(w, j - 1) - \pi(a))|T(w, s_j), a \in S^* \cap Z_{s_j}]
\geq \mathbb{E}[f(SC(w, j - 1) - \pi(a))|T(w, s_j), a \in S^* \cap Z_{s_j}]
\geq \mathbb{E}[\Delta(a|SC(w, j - 1))|T(w, s_j), a \in S^* \cap Z_{s_j}].
\]

The first inequality is from the definition of function \( g \) as it is defined in equation 24. The last inequality is from submodularity of \( f \). Now from the last inequality and lemma 27 we have
\[
\mathbb{E}[f(SC(w, j)|T(w, s_j)] \geq (1 - \frac{p}{k})f(SC(w, j - 1)) + \mathbb{E}[\Delta(a|SC(w, j - 1))|T(w, s_j), a \in S^* \cap Z_{s_j}].
\]

Now from lemma 2 and the above inequality we can show
\[
\mathbb{E}[f(SC(w, j)|T(w, s_j)] \geq (1 - \frac{p}{k})f(SC(w, j - 1)) + \frac{1}{k}(f(S^*) - f(SC(w, j - 1)))
\]

32
Thus,
\[ \mathbb{E}[f(SC(w, j)) - f(SC(w, j - 1))|T(w, s_j)] \geq \frac{1}{k}(f(S^*) - (p + 1)f(SC(w, j - 1)) \right). \\

Hence, by taking expectation on \( T(w, s_{j-1}) \), and by Proposition \( \text{II} \)
\[ \mathbb{E}[f(SC(w, j)) - f(SC(w, j - 1))|T(w, s_j)] \geq \frac{1}{k}\mathbb{E}[f(S^*) - (p + 1)f(SC(w, j - 1)|T(w, s_{j-1})]] \right). \\
\]
This figure "Tw3gaus.png" is available in "png" format from:

http://arxiv.org/ps/2010.01901v1
This figure "correctFig.png" is available in "png" format from:

http://arxiv.org/ps/2010.01901v1
This figure "matroid.png" is available in "png" format from:

http://arxiv.org/ps/2010.01901v1
This figure "pmatchoid.png" is available in "png" format from:

http://arxiv.org/ps/2010.01901v1
This figure "pmatchoidmany.png" is available in "png" format from:

http://arxiv.org/ps/2010.01901v1
This figure "tw3uni.png" is available in "png" format from:

http://arxiv.org/ps/2010.01901v1