The Simulated Greedy Algorithm for Several Submodular Matroid Secretary Problems

Tengyu Ma[1], Bo Tang[2], and Yajun Wang[3]

1 **Princeton University***
`tengyu@cs.princeton.edu`
2 **University of Liverpool***
`tangbonk1@gmail.com`
3 **Microsoft Research Asia**
`yajunw@microsoft.com`

Abstract

We study the matroid secretary problems with submodular valuation functions. In these problems, the elements arrive in random order. When one element arrives, we have to make an immediate and irrevocable decision on whether to accept it or not. The set of accepted elements must form an *independent set* in a predefined matroid. Our objective is to maximize the value of the accepted elements. In this paper, we focus on the case that the valuation function is a non-negative and monotonically non-decreasing submodular function.

We introduce a general algorithm for such *submodular matroid secretary problems*. In particular, we obtain constant competitive algorithms for the cases of laminar matroids and transversal matroids. Our algorithms can be further applied to any independent set system defined by the intersection of a *constant* number of laminar matroids, while still achieving constant competitive ratios. Notice that laminar matroids generalize uniform matroids and partition matroids.

On the other hand, when the underlying valuation function is linear, our algorithm achieves a competitive ratio of 9.6 for laminar matroids, which significantly improves the previous result.

1998 ACM Subject Classification F.2.2 Nonnumerical Algorithms and Problems

Keywords and phrases secretary problem, submodular function, matroid, online algorithm

Digital Object Identifier 10.4230/LIPIcs.STACS.2013.478

1 Introduction

In the classical secretary problem [8, 11, 12], one interviewer is interviewing n candidates for a secretary position. The candidates arrive in an online fashion and the interviewer has to decide whether or not to hire the current candidate when he/she arrives. The goal is to hire the best secretary. It has been shown that when the candidates are arriving in random order, there exists an algorithm that hires the best candidate with probability $1/e$, where e is the base of the natural logarithm.

Recently, Babaioff et al. [3] formulated the matroid secretary problem. Instead of hiring one candidate (element), in the matroid secretary problem, we seek to select a set of elements which form an independent set in a matroid. Again, the elements arrive in random order and the weights of the elements are revealed when they arrive. When one element arrives, we have to make an immediate and irrevocable decision on whether to accept this element or not. The important constraint is that the set of accepted elements must form an independent

* This work was done when the authors were visiting Microsoft Research Asia.

set in the predefined matroid. The objective is to maximize the total weights of the selected elements. Notice that the decision on accepting a particular element will impact our ability in accepting future elements.

In the matroid secretary problem, the value of a set of elements is the summation of the weights on these elements, i.e., the valuation function is linear. In some applications, however, it is more natural to measure the quality of a set by a valuation function, which is not necessarily linear. One set of functions widely used in the optimization community are the *submodular* functions. Such functions are characterized as functions with diminishing returns. We give the formal definition in Section 2.

For example, consider the following scenario. An advertiser is targeting a few platforms to reach a good coverage of audience. However, the coverage from different platforms may overlap with each other. In this case, the performance of a particular set of platforms can only be modeled as a submodular function. Assume the advertiser has to negotiate with the platforms one by one in an online fashion and has a hard budget limit on targeting at most k platforms. This is exactly the matroid secretary problem with a submodular valuation function on a uniform matroid.

We can also consider multiple arriving advertisers, while assuming platforms are available offline. One can impose constraints both on the advertisers and platforms, e.g., each advertiser can afford k platforms, and each platform can support at most ℓ advertisers. This scenario can be modeled as an intersection of two partition matroids, with a submodular valuation function, where the objective is to maximize the value of an overall online assignment.

In this paper, we extend the matroid secretary problem to the case with submodular valuation functions. In other words, the weights are not directly associated with elements. Instead, there exists an oracle to query the value of any subset of the elements we have seen. Our objective is to accept a set of elements which are independent in a given matroid with maximum value with respect to a submodular valuation function. We refer such problems as *submodular matroid secretary problems*. We refer the original matroid secretary problems, i.e., those with linear valuation functions, as *linear matroid secretary problems*.

We use the competitive analysis to measure the performance of our algorithms following the matroid secretary problem literature. More formally, let U be the set of elements and $\mathcal{M}$ be a matroid defined on U. Before the process starts, an adversary assigns a submodular valuation function $f(\cdot) : 2^{|U|} \to \mathcal{R}^+ \cup \{0\}$, which maps any subset of U to a non-negative real number. After that, there is a random permutation applied to the elements to decide their arriving order to our online algorithm. Our algorithm can only query $f(\cdot)$ using elements that have been seen. In other words, the algorithm does not know $f(\cdot)$ before any element arrives.

Let $\mathrm{OPT}_f(\mathcal{M}) = \max_{S\in\mathcal{M}} f(S)$ be the value of the optimal independent set. The objective of the submodular matroid secretary problem is to find an algorithm Alg which maximizes the following ratio:

$$\inf_f \frac{\mathbb{E}_{\mathcal{P},\mathcal{A}}[f(\mathrm{Alg}_f(\mathcal{P},\mathcal{A}))]}{\mathrm{OPT}_f(\mathcal{M})}, \tag{1}$$

where $\mathrm{Alg}_f(\mathcal{P},\mathcal{A})$ is the solution generated by the algorithm given permutation $\mathcal{P}$ and the internal randomness $\mathcal{A}$ of the algorithm with valuation function $f(\cdot)$. The expectation is taken over all permutations and the internal randomness of the algorithm. We call the algorithm is C-competitive, i.e., with competitive ratio C, if the ratio in Eqn.(1) is at least $1/C$.

Our contributions. In this paper, we study the submodular matroid secretary problem with submodular valuation functions that are *non-negative* and *monotonically non-decreasing*.

Our contribution is two-fold. First, we develop a general *simulated greedy* algorithm, which is inspired by the algorithm for the linear matroid secretary problem with transversal matroids in [6, 17]. Our algorithm is constant competitive for the submodular matroid secretary problem with laminar matroids and transversal matroids. Our analysis can be extended to the case that the independent set is defined as the intersection of a *constant* number of laminar matroids. Notice that laminar matroids generalize uniform matroids and partition matroids. When applying to the linear matroid secretary problem on laminar matroids, our algorithm improves the competitive ratio from $\frac{16000}{3}$ [15] to 9.6. Our algorithm is also much simpler than the one in [15].

Second, our technique in analyzing submodular functions could be of independent interest. Consider our simulated greedy algorithm for the uniform matroid case with cardinality μ. We maintain two sets M and N, which are initially empty. In each time, we will select an element $e \in U \setminus (M \cup N)$ such that $f_M(e)$ is maximized until $|M| = \mu$, where $f(\cdot)$ is the valuation function. With probability p, e is placed into M. Otherwise, i.e., with probability $1-p$, e is placed into N. We develop machinery to show that $\mathbb{E}[f(N)] = \Theta(\mathbb{E}[f(M)])$, despite the fact that the elements are greedily selected with optimal marginal values against M. This fact is not intuitive though very important in our analysis. See our result in Section 4 in more details.

Related work. The secretary problem has been studied decades ago. It is first published in [12] and has been folklore even earlier [10]. Several results have appeared to generalize the classical secretary problem, while assuming that the elements arrive in random order. For example, Kleinberg [16] gave a $1 + O(1/\sqrt{k})$-competitive algorithm for selecting at most k elements to maximize the sum of the weights. Babaioff et al. [2] provided a constant competitive algorithm for the Knapsack secretary problem, in which each element has a weight and a size, and the objective is to accept a set of elements whose total size is at most a given integer such that the total weight is maximized.

Babaioff et al. [3] systematically introduced the *matroid secretary problem.* The objective is to maximize the total weight of the selected elements S, which form an independent set in a given matroid. They gave an $O(\log r)$-competitive algorithm for a general matroid, i.e., the expected total weight of the elements in S is $O(1/\log r)$ of the optimal solution, where r is the rank of the matroid. The competitive ratio has been recently improved to $O(\sqrt{\log r})$ by Chakraborty et al. [5]. However, the conjecture that the matroid secretary problem with a general matroid allows a constant competitive algorithm is still widely open, while constant competitive algorithms have been found for various matroids: uniform/partition matroids [2, 16], truncated partition matroids [3], graphical matroids [1, 17], transversal matroids [6, 17], laminar matroids [15], and regular and decomposable matroids [7]. For general matroids, Soto [19] developed a constant-competitive algorithm in *random assignment model*, i.e., the weights of the elements are assigned uniformly at random. This result can be extended to the case where the elements arrive in an adversarial order [13].

Gupta et al. [14] studied the *non-monotone* submodular matroid maximization problem for both offline and online (secretary) versions. For the online (secretary) version, they provided a $O(\log r)$-competitive algorithm for general matroids and a constant competitive algorithm for uniform matroids (algorithms achieving constant competitive ratios are obtained independently by Bateni et al.[4]) and partition matroids. Feldman et al. [9] developed a simpler algorithm with a better competitive ratio for partition matroids for *monotonically non-decreasing* submodular functions.

Structure. In Section 2, we present some preliminaries and our algorithm. We then analyze a simple stochastic process in Section 3, which serves as a building block for later analysis. In Section 4, we analyze the algorithm for the cases of laminar matroids and the intersection of constant number of laminar matroids. Due to space limitation, some of the proofs and the analysis for the transversal matroid case are deferred to the full version [18].

2 Preliminaries

2.1 Matroids

In the matroid secretary problem, the set of accepted elements must form an independent set defined by a given matroid.

▶ **Definition 1** (Matroids). Let $U \neq \emptyset$ be the ground set and $\mathcal{I}$ be a set of subsets of U. The system $\mathcal{M} = (U, \mathcal{I})$ is a matroid with independent sets $\mathcal{I}$ if:

1. If $A \subseteq B \subseteq U$ and $B \in \mathcal{I}$, then $A \in \mathcal{I}$.
2. If $A, B \in \mathcal{I}$ and $|A| < |B|$, there exists an element $x \in B \setminus A$ such that $A \cup \{x\} \in \mathcal{I}$.

In this paper, we work with the following two matroids.

▶ **Definition 2** (Laminar matroids). Let $U \neq \emptyset$ be the ground set. Let $\mathcal{F} = \{B_1, \ldots, B_\ell\}$ be a family of subsets over U. $\mathcal{F}$ is a laminar family, if for any B_i, B_j such that $|B_i| \leq |B_j|$, either $B_i \cap B_j = \emptyset$ or $B_i \subseteq B_j$. Each set $B_i \in \mathcal{F}$ is associated with capacity $\mu(B_i)$. The laminar family $\mathcal{F}$ and $\mu(\cdot)$ define a matroid $\mathcal{M} = (U, \mathcal{I})$, such that any set $T \subseteq U$ is independent if for all $1 \leq i \leq \ell$, $|T \cap B_i| \leq \mu(B_i)$.

In particular, each B_i defines a capacity constraint on the independent sets and a set is independent if it satisfies all such constraints. For simplicity, we assume all B_is are distinct and $\mu(B_i) < \mu(B_j)$ if $B_i \subset B_j$. Otherwise, the capacity constraint in B_i is redundant.

▶ **Definition 3** (Transversal matroids). Let $G = (L, R, E)$ be an undirected bipartite graph with left nodes L, right nodes R and edges E. In the transversal matroid defined by G, the ground set is L and a set of left nodes $S \subseteq L$ is independent if there exists a matching in G such that the set of left nodes in the matching is S.

2.2 Submodular functions

In this paper, we assume the quality of the solution is measured by a submodular function. Notice that throughout this paper, we only work with *non-negative* and *monotonically non-decreasing* submodular functions.

▶ **Definition 4.** Let U be the ground set. Let $f(\cdot) : 2^{|U|} \to \mathcal{R}$ be a function mapping any subset of U to a real number. $f(\cdot)$ is a submodular function if:

$$\forall S, T \subseteq U, \;\; f(S) + f(T) \geq f(S \cup T) + f(S \cap T).$$

For simplicity, for any set $S \subseteq U$, we define its marginal function value $f_S(\cdot)$ as follows. For any $T \subseteq U$, $f_S(T) = f(S \cup T) - f(S)$. For singletons, we also write $f_S(e) = f_S(\{e\})$. It is not difficult to see that $f_S(\cdot)$ is submodular if $f(\cdot)$ is submodular.

2.3 The simulated greedy algorithm

Our general algorithm is based on the greedy algorithm, as in Algorithm 1.

Algorithm 1: GREEDY

```
Input: Set $H \subseteq U$ of matroid $(U, \mathcal{I})$ and function $f(\cdot)$
Output: A set of elements $T \subseteq H$ and $T \in \mathcal{I}$
$T \leftarrow \emptyset$;
while $\exists\, e^* = \text{argmax}_{e \in H} \{f_T(e) \mid M \cup \{e\} \in \mathcal{I}\}$ do
    $T \leftarrow T \cup \{e\}$; $H \leftarrow H \setminus \{e\}$;
end
return $T$;
```

Algorithm 2: ONLINE

```
Input: Matroid $(U, \mathcal{I})$ and function $f(\cdot)$
Output: The set of selected elements
        ALG
$M, N, \text{ALG} \leftarrow \emptyset$;
$m \leftarrow Binom(|U|, p)$;
Observe the first $m$ elements denoted by
$H$;
$M \leftarrow \text{GREEDY}(H)$;
for any subsequent element $e$ do
    if $\text{GREEDY}(H \cup \{e\}) \neq \text{GREEDY}(H)$
    then
        $N \leftarrow N \cup \{e\}$;
        if $\text{ALG} \cup \{e\} \in \mathcal{I}$ then
            Accept $e$ and
            $\text{ALG} \leftarrow \text{ALG} \cup \{e\}$;
        end
    end
end
```

Algorithm 3: SIMULATE

```
Input: Matroid $(U, \mathcal{I})$ and function $f(\cdot)$
Output: The set of selected elements $S$
$H, M, N, S \leftarrow \emptyset$;
for each element $e$ do
    Flip a coin with probability $p$ of head;
    if head, $H \leftarrow H \cup \{e\}$;
end
while $\exists\, e^* = \text{argmax}_{e \in U \setminus \{M \cup N\}} \{f_M(e) \mid$
$M \cup \{e\} \in \mathcal{I}\}$ do
    if $e \in H$ then $M \leftarrow M \cup \{e\}$;
    else $N \leftarrow N \cup \{e\}$;
end
Prune $N$ to produce a set of elements $S \in \mathcal{I}$;
```

Our *simulated greedy* algorithm ONLINE works as follows. (We will discuss the name of *simulated greedy* in a minute.) We observe the first m elements H without any selection, where m is sampled from Binomial distribution $Binom(n, p)$ for some chosen probability p. Then we compute the greedy solution GREEDY(H). After that, for any subsequent element e, we test that whether the greedy solution will change if e is added to H hypothetically. If so, we mark e as a candidate and place it in N. Furthermore, if $\text{ALG} \cup \{e\}$ is independent in $\mathcal{I}$ for candidate e and current ALG, we accept e into ALG. (Both N and ALG are initially empty.) The final ALG will be the output of our algorithm. Observe that maintaining set N is not necessary because N only collects elements that has passed the greedy check and might be accepted potentially. However, we keep the notation in the algorithm because it corresponds to the same N in SIMULATE, which is heavily used throughout the analysis.

As we mentioned earlier, ONLINE is a generalization of the algorithms in [6, 17]. In particular, it has been observed that a *simulated* random algorithm in Algorithm 3 can be used in analyzing the performance of ONLINE. (We name ONLINE as a *simulated greedy* algorithm because of the corresponding greedy algorithm which simulates the online version.)

More specifically, SIMULATE works as follows. We maintain two sets M and N which are initially empty. In each step, we select an element $e \in U \setminus (M \cup N)$ such that $f_M(e)$ is maximized and $M \cup \{e\} \in \mathcal{I}$. (If no such element exists, SIMULATE terminates.) Then we toss a biased random coin with probability p to be head, which is the same probability in sampling m in ONLINE. If the coin is head, e is placed into M. Otherwise, e is placed into

N. Since N may not be an independent set in $\mathcal{I}$ after SIMULATE terminates, we prune N to produce $S \subset N$ such that $S \in \mathcal{I}$. The actual pruning step might be different in different application settings.

SIMULATE is useful in analyzing the performance of ONLINE with random arriving elements, because, as the naming suggests, both M and N have the same joint distribution in the two algorithms. This connection is extensively discussed in [6, 17]. For completeness, we provide a proof in the full version [18]. We will guarantee that S in SIMULATE is stochastically dominated by ALG in ONLINE. Since we assume $f(\cdot)$ is non-decreasing, in analyzing the performance of ONLINE, we can focus on S in SIMULATE.

▶ **Lemma 5.** *The sets of elements of H, M and N by* SIMULATE *have the same joint distribution as the H, M and N generated by* ONLINE *with a random permutation of the elements in U.*

3 A simple stochastic process

In this section, we study a simple stochastic process which serves as a building block of our analysis. We will apply this process to either the entire ground set U or some subsets of the elements in U. Therefore, although we use the same notation for M and N in this section, they can be viewed as the intersections between the set of elements that are under consideration and the actual global M and N generated by the algorithm.

The simple stochastic process is defined by an underlying Bernoulli process, with an infinite sequence of independent and identical random variables $X_t \in \{0, 1\}$ for $t \geq 1$. Each variable X_t is a Bernoulli random variable with probability p to be 1.

Our stochastic process is parametrized by a constant $\mu \geq 1$. We maintain two sets M and N, which are initially empty, as follows. Starting from $t = 1$, if $X_t = 1$, we place t into M; otherwise, t is placed in N. The process immediately terminates after $|M| = \mu$.

We associate a non-negative weight w_t to every time stamp t. In particular w_t is a mapping from the previous $t-1$ random variables to a non-negative real number. (w_1 is constant by definition. If the process has been terminated before time t, we set $w_t = 0$.) For any set $T \subseteq \mathbb{N}$, we define the weight as,

$$w(T) = \sum_{t \in T} w_t(X_1, X_2, \ldots, X_{t-1}). \tag{2}$$

Define $w(\emptyset) = 0$. The following proposition shows that the total weights of M and N are close to each other.

▶ Proposition 6. $\mathbb{E}[w(M)] = \frac{p}{1-p}\mathbb{E}[w(N)]$.

Notice that after the process terminates, we have $|M| = \mu$. On the other hand, the size of N might be very large. Our analysis will be based on Ns that are with size at most μ. We produce an independent set S from N by a pruning process as follows.

Pruning. More formally, to address the issue of too large Ns, we define $S = N$ if $|N| \leq \mu$ and $S = \emptyset$ otherwise. Clearly, we have $S \subseteq N$ and $w(S) \leq w(N)$.

We want to show that $w(S)$ is close to $w(N)$ in expectation. However , it is not possible for arbitrary set of $\{w_t\}$. In what follows, we impose a "decreasing weight" condition on $\{w_t\}$, which always holds in our applications. This condition is crucial in building the connection between $w(S)$ and $w(N)$.

▶ **Definition 7** (Decreasing weight mappings). The set of mappings $\{w_t\}$ forms a sequence of decreasing weight mappings if for any $i < j$ and $x_1, x_2, \ldots, x_{i-1}, x_i, \ldots, x_{j-1}$ we have:

$$w_i(x_1, \ldots, x_{i-1}) \geq w_j(x_1, \ldots, x_{i-1}, \ldots, x_{j-1}).$$

▶ Proposition 8. Let $\beta = 2e(1-p)$. If $\{w_t\}$ forms a sequence of decreasing weight mappings, we have

$$\mathbb{E}[w(N)] - \mathbb{E}[w(S)] \le \frac{(\mu + 1 - \mu\beta)\beta^{\mu}}{(1-\beta)^2} \cdot \mathbb{E}[w(S)] \le \frac{(\mu + 1 - \mu\beta)\beta^{\mu}}{(1-\beta)^2} \cdot \mathbb{E}[w(N)]$$

If $\mu = 1$, it can be improved to

$$\mathbb{E}[w(N)] - \mathbb{E}[w(S)] \le \frac{1-p^2}{p^2}\mathbb{E}[w(S)] \le \frac{1-p^2}{p^2}\mathbb{E}[w(N)].$$

We briefly discuss the intuition behind this statement. Our objective is to show that the weight pruned from N to S is small. The random process indicates that the probability for having a large N is exponentially decreasing on its size, e.g., by the Chernoff bound. Therefore, the probability mass of N that is pruned is small. In terms of weight, on the other hand, those larger Ns do have greater weights.

The condition of the decreasing weight mappings comes to rescue. In particular, in this case, the weight of N grows roughly "linear" to its size. As the probability decreases exponentially with the size of N, the total weight pruned can still be bounded as the summation of a geometric sequence for those large Ns. The complete proof can be found in [18].

4 Laminar Matroid

In this section, we study the performance of our simulated greedy algorithm SIMULATE for the submodular matroid secretary problem with a laminar matroid. We first show that the entire process of SIMULATE can be casted as a simple stochastic process as discussed in the previous section. After that, we inspect the pruning stage in details. In particular, for each B_i in the laminar matroid, we study a simple stochastic process restricted on the elements in B_i. The loss on the entire pruning steps can be divided into losses on the B_is, which can be bounded by Proposition 8.

Let μ be the rank of the laminar matroid. Essentially, SIMULATE will select (at most) μ elements. We cast the SIMULATE process to the simple stochastic process with μ as follows.

In the t-th round, when the first $t-1$ random coins are tossed, the current element e in the greedy order is uniquely defined, as well as the current M and N. We define the weight $w_t = f_{M_e}(e)$ where M_e is the current elements in M.

Remark. We make two remarks regarding the connection between the two stochastic processes. First, the original simple stochastic process terminates when $|M| = \mu$. SIMULATE might terminate earlier because of the limit on the number of elements. In such cases, we assume the availability of an *infinite* number of dummy elements, with zero weights, which will eventually fill up M. In particular, when any of these dummy element arrives at time t, $w_t = 0$ with respect to the previous random outcomes. Notice that these dummy elements will enlarge the size of N without increasing the weights of N and S. So all conclusions we draw in last section still hold. Second, M (as well as N and S) in the simple stochastic process consists of time stamps, while in all processes we study later M consists of real elements. Nevertheless, for every real element $e \in M$, we define $w(e) = w_t$ where t is the time e appears in the greedy order of SIMULATE. Both w_t and $w(e)$ are random variables. We have $w(M) = \sum_{e \in M} w(e)$.

We extend the $w(\cdot)$ to elements besides those in M. In particular, $w(e) = f_{M_e}(e)$ for $e \in M \cup N$, i.e., e appears in the greedy order of SIMULATE, where M_e is the current set of elements in M when e appears. If $e \notin M \cup N$, set $w(e) = 0$. Notice that $w(M) = f(M)$

by definition. Furthermore, each element in the offline optimal solution has probability p in H, i.e., a head coin is associated with it. By submodularity of $f(\cdot)$, the expected value of the optimal solution in H is at least $p \cdot \mathrm{OPT}$. On the other hand, the greedy algorithm is a 2-approximation with a matroid constraint when the valuation function is monotone and submodular. Together with Proposition 6, we have

▶ **Lemma 9.**

$$\mathbb{E}[f(M)] = \mathbb{E}[w(M)] = \frac{p}{1-p}\mathbb{E}[w(N)] \geq \frac{p}{2} \cdot \mathrm{OPT}$$

Pruning. Notice that although M is independent, N might not be independent. We obtain S by pruning N as follows.

$$S = N \setminus \left(\bigcup_{B \in \mathcal{F}} \mathbf{1}_{|N \cap B| > \mu(B)} \cdot (N \cap B) \right), \tag{3}$$

where $\mathbf{1}_{cond} \cdot (N \cap B) = N \cap B$ if *cond* is true and empty otherwise. In other words, if one constraint B_i is violated in N, we remove all elements in B_i from N. Clearly, S is independent. Furthermore, since ALG will be the greedy independent set of N for a random order, it is straightforward to show that $S \subseteq \mathrm{ALG}$.

Therefore, it is sufficient to bound $\mathbb{E}[f(S)]$. To do that, we first provide a lower bound for $\mathbb{E}[w(S)]$. After that, we bound $\mathbb{E}[f(S)]$ in terms of $\mathbb{E}[w(S)]$.

Roadmap. Here we briefly outline our strategy in getting the two pieces of results. To measure $\mathbb{E}[w(S)]$, we estimate the weight loss due to the pruning in Eqn.(3). For each constraint B_i, we cast the stochastic process in SIMULATE in processing elements in B_i into a simple stochastic process with $\mu(B_i)$. By invoking Proposition 8, the weight loss $w(N \cap B_i) - w(S \cap B_i)$ is $2^{O(\mu(B_i))} \cdot w(N \cap B_i)$, which is charged to all elements in B_i proportionally to $\mathbf{1}_{e \in N} w(e)$ for all $e \in B_i$. The catch here is, for each element $e \in U$, the set of $\{B_i\}$ containing e has a strictly increasing $\{\mu(B_i)\}$ sequence. Therefore, the charges on e form a geometric sequence which in total will not exceed a constant fraction of $\mathbf{1}_{e \in N} \cdot w(e)$. Since $w(N) = \sum_{e \in N} w(e)$, the total weight loss is a constant fraction.

The second piece of ingredient is to make a connection between $\mathbb{E}[f(S)]$ and $\mathbb{E}[w(S)]$. For simplicity, let us consider $\mathbb{E}[f(N)]$ and $\mathbb{E}[f(M)]$ instead to convey the idea. Recall that $w(N) = \sum_{e \in N} f_{M_e}(e)$, where M_e is the set of elements in M when e arrives. Therefore, it is not intuitive why $\mathbb{E}[f(N)]$ should be large in the first place. To elaborate, we consider function $F = f(M) + 2f(N) - f(M \cup N)$ during the execution of the algorithm, which is a lower bound of $2f(N)$. We can view $f(M) + f(N) - f(M \cup N)$ as the *intersection* between M and N, e.g., if $f(\cdot)$ is modeling a set cover. During the execution of the algorithm, when e arrives, we have two cases: (1) $f_{M_e}(e) \approx f_{N_e}(e)$, where M_e and N_e are the current set of M and N respectively. F will grow nicely proportional to $f_{M_e}(e)$ in this case. (2) $f_{M_e}(e) \gg f_{N_e}(e)$. Notice e is placed into M with probability p, in which case F grows proportional to $f_{M_e}(e)$ as well. This is because $f_{M_e \cup N_e}(e) \leq f_{N_e}(e) \ll f_{M_e}(e)$ due to the submodularity of $f(\cdot)$. Therefore, F grows in both cases in expectation, which gives a lower bound for $\mathbb{E}[f(N)]$ with respect to $\mathbb{E}[f(M)]$. The analysis in bounded $f(S)$ is more complicated. Though the underlying idea is identical. We formally implement these two ideas in Lemma 10 and Lemma 11.

▶ **Lemma 10.** *Let $\beta = 2e(1-p)$. We have*

$$\mathbb{E}[w(S)] \geq (1 - \frac{2\beta}{(1-\beta)^3})\mathbb{E}[w(N)].$$

Proof. Since for a fixed set of random outcomes, $w(\cdot)$ is a linear function. By Eqn.(3), we have that

$$\mathbb{E}[w(N)] \le \mathbb{E}[w(S)] + \sum_{B\in\mathcal{F}} \mathbb{E}[w(\mathbf{1}_{|N\cap B|>\mu(B)} \cdot (N\cap B))].$$

Now we focus on the term $\mathbb{E}[w(\mathbf{1}_{|N\cap B|>\mu(B)} \cdot (N\cap B))]$ and the simulated greedy algorithm on elements in B, i.e., a particular constraint in $\mathcal{F}$. We isolate B in the process by rearranging the randomness as follows. First, for each element in $U\setminus B$, we assign an independent random coin to it, i.e., if this element appears in the algorithm, its random coin will be tossed. For a fixed outcome of all random coins outside of B, the simulated greedy algorithm is a *simple stochastic process* for the elements in B. The only difference, however, is the process may terminate before $|M\cap B| = \mu(B)$. This can be easily remedied by appending dummy elements as before. Recall that $\beta = 2e(1-p)$. By Proposition 8, we have:

$$\mathbb{E}[\mathbf{1}_{|N\cap B|>\mu(B)} \cdot w(N\cap B)] \le \frac{(\mu(B)+1-\mu(B)\beta)\beta^{\mu(B)}}{(1-\beta)^2} \cdot \mathbb{E}[w(N\cap B)]. \tag{4}$$

It follows that

$$\begin{aligned}
&\mathbb{E}[w(N)] \\
&\le \mathbb{E}[w(S)] + \sum_{B\in\mathcal{F}} \frac{(\mu(B)+1-\mu(B)\beta)\beta^{\mu(B)}}{(1-\beta)^2} \cdot \mathbb{E}[w(N\cap B)] \\
&= \mathbb{E}[w(S)] + \frac{1}{(1-\beta)^2} \sum_{B\in\mathcal{F}} \sum_{e\in U} \mathbb{E}[(\mu(B)+1-\mu(B)\beta)\beta^{\mu(B)} \cdot w(e)\mathbf{1}_{e\in B} \cdot \mathbf{1}_{e\in N}] \\
&= \mathbb{E}[w(S)] + \frac{1}{(1-\beta)^2} \sum_{e\in U} \mathbb{E}\left[w(e)\mathbf{1}_{e\in N} \left(\sum_{B\in\mathcal{F}} (\mu(B)+1-\mu(B)\beta)\beta^{\mu(B)} \cdot \mathbf{1}_{e\in B}\right)\right] \\
&\le \mathbb{E}[w(S)] + \frac{1}{(1-\beta)^2} \sum_{e\in U} \mathbb{E}[w(e)\mathbf{1}_{e\in N}] \left(\sum_{i\ge 1} (i+1-i\beta)\beta^i\right) \qquad (5) \\
&= \mathbb{E}[w(S)] + \frac{2\beta}{(1-\beta)^3} \mathbb{E}[w(N)]
\end{aligned}$$

Eqn.(5) uses the fact that the set of constrains $\{B_i\}$ containing an element e has a strictly increasing sequence of $\{\mu(B_i)\}$. ◀

We then bound $\mathbb{E}[f(S)]$ as follows. For an element e, let N_e be the set of elements in N when e appears in SIMULATE. We define $g(e) = f_{N_e}(e)$ if $e \in M \cup N$ and $g(e) = 0$ otherwise. [1]

▶ **Lemma 11.** *For any $t > 0$, let $\theta = 1 + \frac{(1-p)t}{p}$. We have*

$$\mathbb{E}[f(S)] \ge (\frac{1}{\theta} - \frac{(1-\beta)^3}{t((1-\beta)^3 - 2\beta)})\mathbb{E}[w(S)]$$

[1] We define $g(e)$ based on N_e instead of S_e, i.e., the current set of elements in S, because S_e is still a random set even all the randomness before e's arrival is fixed.

Proof. Let $g(S) = \sum_{e \in S} g(e)$. Since $S \subseteq N$, we have $f(S) \geq g(S)$ by the submodularity of $f(\cdot)$. We inspect the function $F(S, M, N) = t \cdot g(S) + f(M) - f(M \cup N)$. By the monotonicity of f, $f(S) \geq g(S) \geq F(S, M, N)/t$.

Define $\Delta_e = F(S'_e, M'_e, N'_e) - F(S_e, M_e, N_e)$ where M'_e (resp. N'_e and S'_e) is the set M (resp. N and S) after we process element e. If $e \notin M \cup N$, define $\Delta_e = 0$. Therefore, $F(S, M, N) = \sum_{e \in U} \Delta_e$. Let $\mathcal{R}_e$ be the sub-σ-algebra encoding all randomness up to the time e is picked in SIMULATE. Notice that M_e and N_e are $\mathcal{R}_e$ measurable. We have $\Pr[e \in M \mid \mathcal{R}_e] = p$ and $\Pr[e \in N \mid \mathcal{R}_e] = 1 - p$.

$$\begin{aligned}
\mathbb{E}[\Delta_e \mid \mathcal{R}_e] &= t \cdot (\mathbb{E}[g(S') - g(S) \mid \mathcal{R}_e]) + (\mathbb{E}[f(M') - f(M) \mid \mathcal{R}_e]) \\
&\quad - (\mathbb{E}[f(M' \cup N') - f(M \cup N) \mid \mathcal{R}_e]) \\
&= t \cdot \Pr[e \in S \mid \mathcal{R}_e] f_{N_e}(e) + \Pr[e \in M \mid \mathcal{R}_e] f_{M_e}(e) - f_{M_e \cup N_e}(e)
\end{aligned}$$

Then we bound $\mathbb{E}[\Delta_e \mid \mathcal{R}_e]$ by case analysis. Notice that $\Pr[e \in M \mid \mathcal{R}_e] + \Pr[e \in N \mid \mathcal{R}_e] = 1$ and $\Pr[e \in N \mid \mathcal{R}_e] \geq \Pr[e \in S \mid \mathcal{R}_e]$.
Case 1: $f_{M_e}(e) \geq \theta \cdot f_{N_e}(e)$.

$$\begin{aligned}
\mathbb{E}[\Delta_e \mid \mathcal{R}_e] &\geq \Pr[e \in M \mid \mathcal{R}_e](f_{M_e}(e) - f_{M_e \cup N_e}(e)) - \Pr[e \in N \mid \mathcal{R}_e] f_{M_e \cup N_e}(e) \\
&\geq \frac{p}{1-p}(1 - \frac{1}{\theta}) \Pr[e \in S \mid \mathcal{R}_e] f_{M_e}(e) - \Pr[e \in N \mid \mathcal{R}_e] f_{M_e}(e)
\end{aligned}$$

Case 2: $f_{M_e}(e) < \theta \cdot f_{N_e}(e)$.

$$\begin{aligned}
\mathbb{E}[\Delta_e \mid \mathcal{R}_e] &\geq t \cdot \Pr[e \in S \mid \mathcal{R}_e] f_{N_e}(e) - \Pr[e \in N \mid \mathcal{R}_e] f_{M_e \cup N_e}(e) \\
&\geq \frac{t}{\theta} \Pr[e \in S \mid \mathcal{R}_e] f_{M_e}(e) - \Pr[e \in N \mid \mathcal{R}_e] f_{M_e}(e)
\end{aligned}$$

By definition of θ, we have $\frac{p}{1-p}(1 - \frac{1}{\theta}) = t/\theta$. So

$$\mathbb{E}[\Delta_e \mid \mathcal{R}_e] \geq \frac{t}{\theta} \Pr[e \in S \mid \mathcal{R}_e] f_{M_e}(e) - \Pr[e \in N \mid \mathcal{R}_e] f_{M_e}(e)$$

Therefore

$$\begin{aligned}
t \cdot \mathbb{E}[f(S)] \geq \mathbb{E}[F(S, M, N)] &= \sum_e \mathbb{E}_{\mathcal{R}_e}[\mathbb{E}[\Delta_e \mid \mathcal{R}_e]] \\
&\geq \sum_{e \in U} \mathbb{E}_{\mathcal{R}_e}[\frac{t}{\theta} \Pr[e \in S \mid \mathcal{R}_e] f_{M_e}(e) - \Pr[e \in N \mid \mathcal{R}_e] f_{M_e}(e)] \\
&= \frac{t}{\theta} \mathbb{E}[w(S)] - \mathbb{E}[w(N)] \qquad (6) \\
&\geq \frac{t}{\theta} \mathbb{E}[w(S)] - \frac{(1-\beta)^3}{(1-\beta)^3 - 2\beta} \mathbb{E}[w(S)]
\end{aligned}$$

The last inequality is by Lemma 10. So $\mathbb{E}[f(S)] \geq (\frac{1}{\theta} - \frac{(1-\beta)^3}{t((1-\beta)^3 - 2\beta)}) \mathbb{E}[w(S)]$ ◀

Combining all the results together, we have an algorithm with competitive ratio at most 211 with $p = 0.9794$ and $t = 10.1415$.

▶ **Theorem 12.** *There is an online algorithm with competitive ratio at most 211 for the submodular matroid secretary problem with laminar matroids.*

5 Conclusion

In this paper, we develop a general algorithm for the submodular matroid secretary problems. In particular, we obtain constant competitive algorithms for laminar matroids and transversal matroids. Our algorithm can also handle the intersection of a constant number of laminar matroids, which makes our algorithm more applicable. We state the results for transversal matroids and intersection of matroids, and defer their proofs in the full version [18], where we also analyze the algorithm for the linear matroid secretary problem with laminar matroids.

▶ **Theorem 13.** *There is an online algorithm with competitive ratio at most 95 for the submodular matroid secretary problem with transversal matroids.*

▶ **Theorem 14.** *For any constant k, there is an online algorithm with competitive ratio at most $\frac{1000k(k+1)}{9}$ for the submodular matroid secretary problem with the intersection of k laminar matroids.*

▶ **Theorem 15.** *Algorithm 2 is a 9.6-competitive algorithm for the linear matroid secretary problem with laminar matroids.*

However, our algorithm does not work on general matroid case. Consider the following simple example on graphical matroids. There is a single heavy edge (u, v) in the graph. There is a large number of nodes $K = \{u_1, u_2, \ldots, u_n\}$ and edges $\{(u, u_i), (u_i, v) \mid u_i \in K\}$. The weight on each such edge is very small. It is easy to verify that the probability that our algorithm will accept (u, v) is exponentially small on n. Nevertheless, our algorithm can handle graphical matroids using the same decomposition technique [1], i.e., by reducing the problem to a partition matroid, which is randomly selected from two constructed partition matroids. On the other hand, it would be interesting to characterize the independent set constraints for which our algorithm framework is constant competitive.

In the distinction between the submodular case and linear case in matroid secretary problem, we still cannot adapt the recent $O(\sqrt{\log r})$ competitive algorithm in [5] as well as the constant competitive algorithm for the random assignment model in [19] previously on the linear case. It would be interesting to close this gap. Finally, it is still widely open whether the matroid secretary problem permits constant competitive algorithms for general matroids.

References

1 Moshe Babaioff, Michael Dinitz, Anupam Gupta, Nicole Immorlica, and Kunal Talwar. Secretary problems: weights and discounts. In *SODA*, pages 1245–1254, 2009.

2 Moshe Babaioff, Nicole Immorlica, David Kempe, and Robert Kleinberg. A knapsack secretary problem with applications. In *APPROX/RANDOM*, pages 16–28, 2007.

3 Moshe Babaioff, Nicole Immorlica, and Robert Kleinberg. Matroids, secretary problems, and online mechanisms. In *SODA*, pages 434–443, 2007.

4 Mohammad Hossein Bateni, MohammadTaghi Hajiaghayi, and Morteza Zadimoghaddam. The submodular secretary problem and its extensions. In *APPROX*, pages 39–52, 2010.

5 Sourav Chakraborty and Oded Lachish. Improved competitive ratio for the matroid secretary problem. In *SODA*, pages 1702–1712, 2012.

6 Nedialko B. Dimitrov and C. Greg Plaxton. Competitive weighted matching in transversal matroids. In *ICALP*, pages 397–408, 2008.

7 Michael Dinitz and Guy Kortsarz. Matroid secretary for regular and decomposable matroids. *CoRR*, abs/1207.5146, 2012.

8 E. B. Dynkin. Optimal choice of the stopping moment of a markov process. *Dokl.Akad.Nauk SSSR*, 150:238–240, 1963.

9 Moran Feldman, Joseph (Seffi) Naor, and Roy Schwartz. Improved competitive ratios for submodular secretary problems (extended abstract). In *APPROX*, pages 218–229, 2011.

10 T. S. Ferguson. Who solved the secretary problem? *Statistical science*, pages 282–289, 1989.

11 PR Freeman. The secretary problem and its extensions: A review. *International Statistical Review*, pages 189–206, 1983.

12 M. Gardner. Mathematical games column. *Scientific American*, 35, 1960.

13 Shayan Oveis Gharan and Jan Vondrák. On variants of the matroid secretary problem. In *ESA*, pages 335–346, 2011.

14 A. Gupta, A. Roth, G. Schoenebeck, and K. Talwar. Constrained non-monotone submodular maximization: Offline and secretary algorithms. In *WINE*, pages 246–257, 2010.

15 Sungjin Im and Yajun Wang. Secretary problems: Laminar matroid and interval scheduling. In *SODA*, 2011.

16 Robert Kleinberg. A multiple-choice secretary algorithm with applications to online auctions. In *SODA*, pages 630–631, Philadelphia, PA, USA, 2005.

17 Nitish Korula and Martin Pál. Algorithms for secretary problems on graphs and hypergraphs. In *ICALP (2)*, pages 508–520, 2009.

18 T. Ma, B. Tang, and Y. Wang. The simulated greedy algorithm for several submodular matroid secretary problems. *arXiv preprint arXiv:1107.2188*, 2011.

19 Jose Soto. Matroid secretary problem in the random assignment model. In *SODA*, pages 1275–1284, 2011.